

ABSTRACT

Title of dissertation: COMBINING LINGUISTIC AND
 MACHINE LEARNING TECHNIQUES
 FOR WORD ALIGNMENT IMPROVEMENT

Necip Fazıl Ayan, Doctor of Philosophy, 2005

Dissertation directed by: Professor Bonnie J. Dorr
 Department of Computer Science

Alignment of words, i.e., detection of corresponding units between two sentences that are translations of each other, has been shown to be crucial for the success of many NLP applications such as statistical machine translation (MT), construction of bilingual lexicons, word-sense disambiguation, and projection of resources between languages. With the availability of large parallel texts, statistical word alignment systems have proven to be quite successful on many language pairs. However, these systems are still faced with several challenges due to the complexity of the word alignment problem, lack of enough training data, difficulty learning statistics correctly, translation divergences, and lack of a means for incremental incorporation of linguistic knowledge.

This thesis presents two new frameworks to improve existing word alignments using supervised learning techniques. In the first framework, two rule-based approaches are introduced. The first approach, Divergence Unraveling for Statistical MT (DUSTer), specifically targets translation divergences and corrects the

alignment links related to them using a set of manually-crafted, linguistically-motivated rules. In the second approach, Alignment Link Projection (ALP), the rules are generated automatically by adapting transformation-based error-driven learning to the word alignment problem. By conditioning the rules on initial alignment and linguistic properties of the words, ALP manages to categorize the errors of the initial system and correct them.

The second framework, Multi-Align, is an alignment combination framework based on classifier ensembles. The thesis presents a neural-network based implementation of Multi-Align, called NeurAlign. By treating individual alignments as classifiers, NeurAlign builds an additional model to learn how to combine the input alignments effectively.

The evaluations show that the proposed techniques yield significant improvements (up to 40% relative error reduction) over existing word alignment systems on four different language pairs, even with limited manually annotated data. Moreover, all three systems allow an easy integration of linguistic knowledge into statistical models without the need for large modifications to existing systems. Finally, the improvements are analyzed using various measures, including the impact of improved word alignments in an external application—phrase-based MT.

COMBINING LINGUISTIC AND MACHINE LEARNING TECHNIQUES
FOR WORD ALIGNMENT IMPROVEMENT

by

Necip Fazıl Ayan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Advisory Commmittee:

Professor Bonnie J. Dorr, Chair
Professor Rebecca Green
Professor Rebecca Hwa
Professor Philip Resnik
Professor Amy Weinberg

© Copyright by

Necip Fazıl Ayan

2005

DEDICATION

This thesis is dedicated to my parents, Muradiye and Cevdet Ayan, for having a dream for me and doing everything they can to make that dream come true, to my wife, Burcu Karagöl-Ayan, for loving and supporting me for the last 12 years and making me a better person, and to my son, Alperen Ayan, for bringing a new meaning to my life.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Dr. Bonnie Dorr, for always pushing me hard, challenging my work, teaching invaluable lessons, being available all the time, and most importantly helping me become a better researcher. I'm grateful to her for her understanding, patience and support in bad times over the last five years.

I would also like to thank Dr. Christof Monz for his guidance and help while developing the algorithms in this thesis.

I thank all my committee members, Dr. Rebecca Green, Dr. Rebecca Hwa, Dr. Philip Resnik and Dr. Amy Weinberg for sparing their invaluable time to read my dissertation and making very helpful comments.

I thank Adam Lopez for providing me with his word alignment system and making several insightful comments on my work, Nitin Madnani for pointing out several resources that made my life easier while implementing the algorithms in this thesis, and Dr. Nizar Habash for his help in early stages of my work. I should also express my gratitude to Dr. Franz Och and Dr. Philipp Köhn for making GIZA++ and Pharaoh public, and allowing me to use them in my experiments.

I would like to thank Muhammet Pakdil for listening to me day and night, supporting me through all the good and bad times, and being a true friend. I also

thank my friends, Füsün Yaman, Evren Şirin, Mustafa Murat Tıkır, Okan Kolak, Kemal Akkaya, Çağdaş Dirik and several others, for making my life at College Park easier and enjoyable. I thank Karagöl family, who accepted me as their own son or brother, for their moral support and encouragement.

I thank all the organizations and individuals who made this research possible by financially supporting me.

Most of all, I thank my family for helping me to be where I am right now. I owe a great debt to my parents, who always believed in me, supported me and did everything possible to make sure I get a good education. I thank my wife, Burcu, for letting me become a part of her life, being on my side through all the difficult times in my graduate life, and making me a better person. This thesis would not have materialized without her constant love, trust, support, and encouragement. Finally, I thank my son Alperen—the best thing that ever happened to me in my life—for filling my heart with joy and happiness, and being an inspiration to me. This thesis is dedicated to them.

TABLE OF CONTENTS

List of Tables	viii
List of Figures	xii
1 Introduction	1
1.1 Motivation	4
1.2 Techniques for Improving Existing Word Alignments	8
1.2.1 Improving One Word-Alignment System	9
1.2.2 Combining Multiple Alignments	16
1.3 Contributions	20
1.4 Thesis Layout & Brief Overview of Chapters	21
2 Related Work	24
2.1 Sentence-Level Alignment	26
2.2 Word Level Alignment	28
2.2.1 Heuristic Methods	31
2.2.2 Statistical Methods	35
2.3 Phrase Level Alignment	54
2.4 Combining Word Alignments	58
2.5 Applications	62
2.6 Evaluation of Word Alignments	63
2.7 Discussion	68
3 DUSter: Divergence Unraveling for Statistical MT	74
3.1 Translation Divergences	76
3.2 Relating Alignment Errors to Divergence Types	80
3.3 DUSter: System Description	84
3.3.1 Parameters	86
3.3.2 Universal Rule Set	88
3.3.3 Setting Up DUSter for a New Language	94
3.3.4 Improving Alignments Using DUSter	98
3.4 Evaluation	101
3.4.1 Settings	101
3.4.2 Results	102
3.5 Summary	103

4	Alignment Link Projection (ALP)	105
4.1	Transformation-based Error-driven Learning	108
4.2	Notation	111
4.3	Description and Parameters of ALP	113
4.3.1	Initial Alignment	114
4.3.2	TBL Templates	115
4.3.3	Instantiation of Templates	122
4.3.4	Best Rule Selection	125
4.4	Evaluation Data and Settings	125
4.5	Experimental Results	128
4.5.1	Results for Different Initial Alignments	130
4.5.2	Results for Different Sets of Templates	134
4.5.3	Using Different Types of Instantiation	136
4.5.4	Using Different Methods for Best Rule Selection	137
4.5.5	Effects of Training Size for Input Aligners	138
4.5.6	Analysis of ALP Rules	139
4.6	Summary	142
5	Multi-Align: A Framework for Combining Multiple Alignments	144
5.1	Classifier Ensembles	147
5.1.1	Why Ensembles Work	148
5.1.2	When Ensembles Work	149
5.1.3	Creating Ensemble Members	153
5.1.4	Combining Ensemble Members	155
5.1.5	Ensembles in NLP	157
5.2	Multi-Align	158
5.3	Preliminary Study: Alignment Combination Using Weighted Sum- mation	167
5.3.1	Set of Features	168
5.3.2	Perceptron Learning	169
5.3.3	Using Perceptrons for Alignment Combination	172
5.4	Evaluation Data and Settings	173
5.5	Experimental Results	175
5.5.1	Effects of Feature Selection for Link Classes	175
5.5.2	Effects of Different Feature Functions	177
5.5.3	Effects of Training Size for Input Aligners	178
5.6	Summary	179
6	NeurAlign	181
6.1	Neural Networks	183
6.2	NeurAlign in Multi-Align Framework	187
6.3	NeurAlign Approach	189
6.3.1	Extracting Features	190

6.3.2	Learning A Classifier	192
6.4	Evaluation	197
6.4.1	Training and Evaluation Data	197
6.4.2	Neural Network Settings	200
6.4.3	Description of Classification Data	201
6.5	Experimental Results	203
6.5.1	Effects of Using Different Features for NeurAlign ₁	204
6.5.2	Effects of Partitioning Data (NeurAlign ₁ vs. NeurAlign ₂)	206
6.5.3	Effects of Number of Alignment Systems	210
6.5.4	Effects of Training Size for NeurAlign	212
6.5.5	Effects of Training Size for Input Aligners	213
6.5.6	Stability of Results	215
6.5.7	Experiments on Languages With Scarce Resources	217
6.6	Summary	219
7	Analysis of Alignments and MT Evaluation	221
7.1	Analysis of Improvements in Alignments	223
7.1.1	Precision vs. Recall	223
7.1.2	Comparison of Number of Alignment Links	227
7.1.3	Comparison of Fertilities	230
7.1.4	Resolution of Ambiguous Links	233
7.2	Phrase-based Machine Translation	240
7.3	MT Evaluation: Results and Analysis	245
7.3.1	Phrase Table Analysis	248
7.3.2	MT Evaluation using BLEU	249
7.3.3	Parameters	251
7.4	Summary	253
8	Conclusions	254
8.1	Contributions	255
8.2	Limitations	257
8.3	Future Work	258
A	DUSTer Parameters	263
B	DUSTer Universal Rules for English-Spanish	297
C	ALP Rules (on English-Spanish)	303
D	ALP Rules (on English-Chinese)	306

LIST OF TABLES

2.1	Summary of IBM Models	38
3.1	Human and GIZA++ Alignments Between an English and Spanish Sentence	81
3.2	Number and Percentage of Alignment Errors by GIZA++ (on English-Spanish)	82
3.3	Semantic and Syntactic Categories of Words from Missing, Added and Replaced Links (on English-Spanish)	83
3.4	DUSTer Parameters in English	87
3.5	Set of Universal Rules in English-Spanish	93
3.6	Times Needed To Prepare DUSTer for 3 Languages	97
3.7	GIZA++ and DUSTer Results (on English-Spanish)	102
4.1	Templates for Expanding the Alignment A According to a Validation Alignment V	117
4.2	Templates for Deleting Spurious Links in a Given Alignment A . . .	118
4.3	Number of Words with at Least One Correct Alignment Link (on English-Spanish)	120
4.4	Templates for Handling Multi-Word Correspondences in a Given Alignment A	121
4.5	Templates for Correcting One-to-One Correspondences in a Given Alignment A	122

4.6	GIZA++ Results (on English-Spanish and English-Chinese)	127
4.7	ALP Results Using Different Initial Alignments (on English-Spanish)	130
4.8	ALP Results Using Different Initial Alignments (on English-Chinese)	131
4.9	ALP Results Using GIZA++(int) as Initial Alignment (on English-Spanish and English-Chinese)	134
4.10	ALP Results Using Different Template Instantiations (on English-Spanish and English-Chinese)	136
4.11	ALP Results Using Different Rule Selection Methods (on English-Spanish and English-Chinese)	137
4.12	ALP Results Using Different Initial Alignments When GIZA++ is Trained on More Data (on English-Chinese)	138
5.1	Multi-Align: Effects of Feature Selection for Link Classes (on English-Chinese)	176
5.2	Multi-Align: Effects of Different Feature Functions (on English-Chinese)	177
5.3	Multi-Align: Effects of Using More Training Data for Initial Alignments (on English-Chinese)	179
6.1	Distribution of Instances According to GIZA++ Outputs (on English-Spanish)	193
6.2	Error Rates According to POS Tags for GIZA++($e \rightarrow s$) (on English-Spanish)	195
6.3	GIZA++ Results (on English-Spanish, English-Chinese, English-Arabic and English-Romanian)	200
6.4	SAHMM Results (on English-Chinese)	200
6.5	NeurAlign ₁ : Effects of Feature Set for Combining Alignments (on English-Spanish)	205
6.6	NeurAlign ₁ : Effects of Feature Selection for Partitioning (on English-Spanish)	207

6.7	NeurAlign ₂ : Effects of Feature Set for Combining Alignments (on English-Spanish)	207
6.8	NeurAlign: Effects of Feature Selection for Partitioning Using GIZA++ and SAHMM Alignments (on English-Chinese)	209
6.9	NeurAlign: Effects of Using a Higher Number of Input Alignments (on English-Chinese)	211
6.10	NeurAlign: Effects of Training Size (on English-Chinese)	213
6.11	NeurAlign: Effects of Using More Data to Train Input Alignments (on English-Chinese)	214
6.12	Stability of NeurAlign (on English-Chinese)	216
6.13	GIZA++ vs. NeurAlign (on English-Arabic)	217
6.14	GIZA++ vs. NeurAlign (on English-Romanian)	218
7.1	Relative Improvements in AER by ALP and NeurAlign over 5 Different Alignments (in Percentages)	222
7.2	Number of Alignment Links for Different Alignments (on English-Chinese)	228
7.3	Percentage of English Words with Different Fertilities (on English-Chinese)	231
7.4	Percentage of Chinese Words with Different Fertilities (on English-Chinese)	232
7.5	Resolution of Ambiguous Cases Among 2 Alignments (on English-Chinese)	235
7.6	Resolution of Ambiguous Cases Among 4 Alignments (on English-Chinese)	237
7.7	Upper Bounds (Assuming Perfect Resolution of Ambiguous Cases) and NeurAlign Results for 3 Sets of Input Alignments (on English-Chinese)	238
7.8	Size of Phrase Tables Generated by Different Alignments (on English-Chinese)	249

7.9	Evaluation of Pharaoh with Different Initial Alignments (on English-Chinese)	250
7.10	MT Parameters (on English-Chinese)	252

LIST OF FIGURES

1.1	DUSTer: Example Rule Application	11
1.2	Alignment Link Projection (ALP)	14
1.3	Multi-Align Framework for Combining Multiple Alignments	17
2.1	Word Alignment Example	29
3.1	Graphical Illustration of Three Types of Alignment Errors	82
3.2	Components of DUSTer	85
3.3	Universal Rules in BNF Format	89
3.4	Universal Rule Application Example	92
3.5	Example Sentences, Dependency Tree and Initial Alignment	98
3.6	DUSTer’s Inferred Alignments from Initial GIZA++ Output	100
4.1	TBL Architecture	108
4.2	Pseudo-code for Alignment Link Projection	112
4.3	Graphical Representation of a Template	115
4.4	Illustration of Deletion Templates	119
4.5	Graphical Illustration of Two Multi-word Correction Templates	122
5.1	Multi-Align Framework for Combining Multiple Alignments	159
5.2	Representation of Grow-diag-final Method in Multi-Align Framework	164

5.3	A Perceptron Network with R Input Units and S Output Units . .	170
6.1	Multilayer Perceptron Overview	184
6.2	NeurAlign in Multi-Align Framework	188
6.3	An Example of Transforming Alignments into Classification Data .	192
6.4	NeurAlign ₁ —Alignment Combination Using All Data At Once . . .	194
6.5	NeurAlign ₂ —Alignment Combination with Partitioning	196
7.1	Precision and Recall for Initial Alignments and ALP	224
7.2	Precision and Recall for Initial Alignments and NeurAlign	226
7.3	Examples of Valid and Invalid Phrase Pairs	243

Chapter 1

Introduction

Parallel texts are texts accompanied by their translation in one or more languages. The first few attempts at using parallel texts, such as machine translation (MT) in the late Fifties, did not succeed because of limited storage and computing capacities of computers, along with the difficulty of creating electronic texts. However, the incredibly rapid growth of the Web, and development of several techniques to collect data from the Internet have led to a huge increase in parallel resources in electronic form (Resnik, 1998; Koehn, 2002; Resnik and Smith, 2003). Since then, parallel texts have become one of the most commonly used resources in natural language processing (NLP), especially in statistical MT.

The most critical task in parallel text processing is *alignment*, i.e., detection of corresponding units between two texts of different languages. Given parallel texts in two different languages, alignment can be extracted at the level of sections, paragraphs, sentences, phrases, collocations, or words. Although the alignment between other units is also an important problem, this thesis will focus on the

alignment of words.

Alignment of words, i.e., detection of corresponding words between two sentences that are translations of each other, is usually an intermediate step of statistical MT that has been shown to be crucial for the success of statistical MT systems (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2003; Koehn et al., 2003), and for many other NLP applications such as construction of bilingual lexicons (Gale and Church, 1991b; Melamed, 1997c), automatic generation of transfer rules for machine translations (Menezes and Richardson, 2001; Carbonell et al., 2002), word-sense disambiguation (Brown et al., 1991a; Gale et al., 1992; Chang et al., 1996; Diab and Resnik, 2002), projection of resources (such as morphological analyzers, part-of-speech taggers, and parsers) from a resource-rich language into other resource-poor languages (Yarowsky et al., 2001; Hwa et al., 2002), and cross-language information retrieval (Fluhr, 1995; Oard and Dorr, 1996).

The word alignment problem is difficult because of the following reasons:

1. Words are not always aligned one-to-one because some languages are more verbose than others. Moreover, nearly 50% of the tokens in any text are function words, which frequently translate into an affix, positional information, part of expressions, phrases, or even nothing at all. There is even less of a one-to-one correspondence between function words than for content words.
2. Some languages make morphological distinctions that are absent in the other. German, for example, makes a number of case distinctions, especially in

adjectives, that are not reflected in the morphology of English.

3. For various reasons, such as language typology, style, and cultural differences, a translator does not always translate literally on a word-by-word basis. Most of the time, words are added or deleted.
4. Idiomatic expressions and phrases in two different languages usually contain words that are not translations of each other.

Given two sentences $\mathbf{e} = e_1, \dots, e_i, \dots, e_I$ and $\mathbf{f} = f_1, \dots, f_j, \dots, f_J$ that are translations of each other, there are $I \times J$ different connections that can be drawn between \mathbf{e} and \mathbf{f} because each of the J words in \mathbf{f} can be connected to any of the I words in \mathbf{e} . Since an alignment is determined by the connections that it contains, and since a subset of the possible connections can be chosen in $2^{I \times J}$ ways, there are $2^{I \times J}$ possible alignments. Different word alignment models reduce the search space by making further assumptions on types of connections between the words. For example, in IBM Models (Brown et al., 1993), each source word f_j can align to exactly one target word e_i , or the null word. Alternatively, the target words can link to any number of source words. As a result, the number of possible alignments is $O(I^J)$ but the complexity is still exponential.

Since Brown et al. (1993) proposed the widely-used IBM models, several researchers have developed various word alignment systems that are based on different models, such as hidden Markov models (HMM), maximum entropy models, log-linear combinations, and similarity-based heuristic methods. Taking advan-

tage of large parallel texts, statistical systems have been shown to outperform their counterparts significantly. Because of their recent success and their ability to handle different languages easier than linguistically-motivated techniques, most of the researchers working on word alignment shifted their focus to statistical methods in the recent years.

In this thesis, instead of building a new word alignment model from scratch, two frameworks are presented to improve existing statistical word alignment systems. One goal is to enrich statistical models with linguistic knowledge. Another goal is to enable the use of different word alignments without heavy modifications to their alignment modeling. The rest of this chapter describes the motivation behind the alignment frameworks presented in the later chapters. Two different approaches to improving word alignments are presented. Finally, the contributions of this thesis are highlighted and the organization of the remainder of this thesis is outlined.

1.1 Motivation

Current statistic alignment systems are faced with five important challenges:

1. Complexity of the word alignment problem,
2. Lack of enough training data (in the face of limited linguistic resources),
3. Learning statistics correctly,

4. Translation divergences, and
5. Lack of a means for incremental incorporation of linguistic knowledge.

The rest of this section discusses each of these 5 challenges in detail.

The first issue is the complexity of the word alignment problem. As mentioned above, the word alignment problem is an exponential problem. Like all other machine learning systems, statistical systems need to make certain assumptions, or *biases*, about the hypothesis to be learned from the training data to reduce the search space of hypotheses. Such biases enable the learning algorithm to perform well in some domains, but not in others. As a result, all systems tend to make similar errors on unseen data because either the model is deficient when handling certain cases (because of its biases) or the generalization learned by the system on training data is not reflected in unseen data.

The second hurdle for statistical systems is the lack of *enough* linguistically annotated training data for capturing generalizations. This has led to the tendency for the development of statistical systems that use large parallel texts without any linguistic knowledge about the languages. However, it is unclear how one determines how much data is sufficient for different language pairs. For example, in some initial experiments presented in this thesis, a training set consisting of 47K sentence English-Spanish pairs yields lower error rates than a training set consisting of 107K English-Chinese sentence pairs using the same word alignment system. The recent trend in statistical systems is to incorporate all the data that

is available and let the system take care of redundant or noisy data. Ultimately, with billions and billions of sentences, this is equivalent to memorizing all possible word (or phrase) correspondences so that there is nothing left unseen for any given test set. In practice, this is nearly impossible to achieve, at least for a majority of language pairs. That is, there will always be language pairs where there is only a limited amount of data. Moreover, statistical systems are still susceptible to their biases in modeling alignments; therefore, it is highly unlikely that they will produce 100% correct alignments even with infinite data.

The third problem is related to infrequent words and frequently-seen function words. The rareness of some words—coupled with the too-frequent occurrence of other words in the training data—makes it very difficult to choose what statistics to use (Dunning, 1993; Moore, 2004). Moreover, most expressions in the languages are only “semi-frozen” and can still undergo a number of linguistic operations (such as inflection, insertion of adjectives and adverbs, conversion to the passive voice, etc.). For example, one of the major problems with the IBM models is their tendency to align rare words to several words on the other side in an attempt to reduce the number of unaligned words. The problem with the function words is more dramatic: Statistical systems are often unable to pick up the correct word correspondences for function words because function words occur in every sentence several times and statistical models are based on co-occurrence statistics. Experiments on English-Spanish data that are presented in Chapter 3 support this claim. The alignment of function words might not be important in certain applica-

tions (e.g, bilingual lexicon construction), but they are vital in machine translation because they usually translate into an affix, positional information, or a sub-part of an expression or phrase.

The fourth problem is the handling of *translation divergences*, i.e., structural differences between languages. Divergences between languages occur when the underlying meaning of a set of words in one language is distributed over a set of words with different syntactic and lexical semantic properties in the other language—e.g., the English verb *fear* corresponds to *tener miedo de* (*have fear of*) in Spanish. The most common types of alignment errors related to divergences occur when a word in one language is translated into a phrasal structure with the addition or deletion of function words (i.e., conflational, inflational and structural divergences) or into words that have different parts of speech (i.e., categorial divergence). This thesis demonstrates empirically that the existence of translationally divergent word pairs is the most significant factor contributing to statistical alignment errors.

The last problem is the difficulty of incremental incorporation of linguistic knowledge into statistical systems. The most common method for injecting linguistic knowledge into the alignment process is to build a new alignment model to account for additional linguistic knowledge in the form of features (or feature functions) (Toutanova et al., 2002; Cherry and Lin, 2003; Liu et al., 2005). In a sense, this is equivalent to discarding already existing systems and building a new system from scratch. There are two problems with this ‘build-everything-from-scratch’ approach: First, the resulting system may lose valuable information

that is inherent in the architecture of previous systems, even when the new system produces better alignments. Second, it is quite difficult to assess the usefulness of different pieces of linguistic knowledge.

In addressing the five challenges above, this thesis demonstrates that it is thus possible to improve on word alignments generated by existing word alignment systems. The core of the work presented in this thesis is the construction of automatic techniques for: (1) detection of common errors produced by existing word alignment systems and (2) correction of those errors to obtain better word alignments.

1.2 Techniques for Improving Existing Word Alignments

This thesis presents two different frameworks to improve existing word alignments. Both of them are based on the assumption that existing alignment systems succeed at capturing some word correspondences but fail at handling certain phenomena, resulting in similar alignment errors. Both approaches attempt to identify alignment links where the input systems perform well, and places where the systems make repeated errors. This is achieved by employing additional linguistic features, such as part-of-speech (POS) tags, dependency relations and semantic-based word classes, or alignment-based features that are extracted from the outputs of input alignments.

The difference between the two frameworks is that one starts with only one

alignment and categorizes the frequently occurring errors systematically, whereas the other makes use of multiple alignments. In the first framework, either new alignment links are added based on already existing links or some of the links that share a common linguistic or alignment-based property are deleted. In contrast, the second framework involves cross-validation of different word alignments. This is achieved by identifying places where the alignments agree or disagree and making a decision about which alignment link to pick in each case according to the features of the words that are involved.

1.2.1 Improving One Word-Alignment System

In the first framework, two new rule-based systems are presented. *Divergence Unraveling for Statistical MT (DUSTer)*, is a system that combines linguistic and statistical knowledge to resolve structural differences between languages during the process of alignment. The goal is to identify places where two sentences are divergent, and correct the alignment links related to them using a set of manually crafted, linguistically motivated rules.

Previous work on divergences (Dorr et al., 2002) showed that at least 10% of the sentence pairs in Arabic/English and Spanish/English are divergent in some way. Moreover, it has been demonstrated that different divergence types frequently co-occur. For instance, for the following two sentences in English and Spanish,

Different politicians please Maria.

Maria tiene gustos de políticos diferentes.

there are four co-occurring divergences: Categorical (*please* is a verb in English but *gusto* is a noun in Spanish), conflational (*please* is conflated to *tener gusto*), thematic (*Maria* and *politicians* switch thematic-to-syntactic realization order), and structural (*politicians* is an argument in English but an oblique in Spanish). Thus, it is important to handle divergences concurrently rather than tackling each divergence one at a time and to pay attention to distinctions in both content and structure.

A preliminary study on analysis of alignment errors made by a statistical system demonstrated that missing alignment links are the most frequent alignment errors introduced by GIZA++, accounting for nearly 80% of the errors. These are precisely the cases that correspond to the divergence classes specified above, most notably *conflational* and *structural* divergences, where a single word in one language corresponds to multiple words in the other language.

The key idea behind DUSter is to relate one or more linguistically-motivated categories—specifically, POS tags and semantic word classes— associated with the English input words to those of the foreign language (FL); the resulting *match sets* are used to infer corrected alignment links. DUSter achieves this by employing a set of manually crafted rules.

Consider the following example:

English: *John fears Mary*

Spanish: *John tiene miedo de Mary*

(Gloss): *John has fear of Mary*

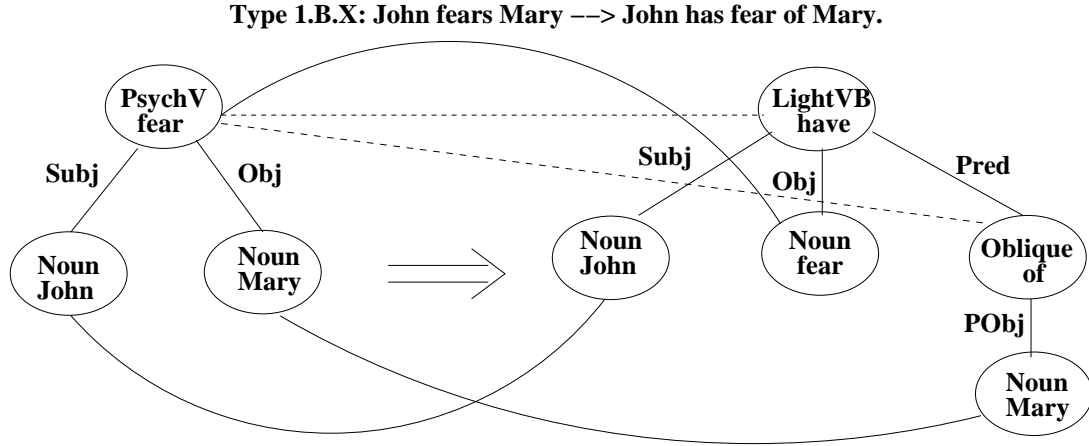


Figure 1.1: DUSTER: Example Rule Application

The rule below (also presented in Figure 1.1 graphically) corresponds to the mapping between ‘*j fears k*’ in English and ‘*j has fear of k*’ in Spanish.

```

1.B.X [ English{2 1 3} Spanish{2 1 3 4 5} ]

[PsychV<1,i,CatVar:V_N,Verb> [Noun<2,j,Subj>] [Noun<3,k,Obj>]] <-->

[LightVB<1,Verb,C:i> [Noun<2,j,Subj>] [Noun<3,i,Obj>]

[Oblique<4,Pred,Prep,C:i> [Noun<5,k,PObj>]]]

```

Note that, in addition to the simple indices, *i*, *j*, etc., *conflated indices* (*C:i*, *C:j*, etc.) are used to refer to semantically light words that co-occur with high-content words but are generally unaligned in the initial alignments. Nodes marked *C:i* are taken to be related structurally to a (single) high-content node marked *i*. Nodes marked *C:i* inherit their alignment links from the node marked *i* during alignment correction. For example, the conflated index *C:i* on the right-hand-side

(RHS) of rule 1.B.X associates two semantically light nodes—node 1 (LightVB) and node 4 (Oblique)—with node 3 (Obj), which has the same alignment index (i) as node 1 in the left-hand-side (LHS) of the rule. In this case, the initial alignment associated with these two nodes (node 3 in RHS and node 1 in LHS) is copied to the two $C:i$ nodes in RHS. That is, the conflated indices provide the appropriate mechanism to copy an alignment link from a single (high-content) word pair (i.e., *fear,miedo(fear)*) to one or more light-content words in the FL (i.e., *fear,tener(have)* and *fear,de(of)*).

Employing such a set of linguistically motivated rules yields promising results but the improvement over the initial alignment is relatively low, primarily because the coverage of the rules is not adequate to capture common errors made by the initial alignment system. Specifically, the hand-constructed rules are both too general and not general enough:

1. The rules are too general in that they are not tailored to accommodate the specific characteristics of individual alignment systems; thus, common errors made by the initial alignment system are not necessarily addressed by the rules.
2. The rules are too specific in that they are designed to handle a small set of divergence examples that are not necessarily represented by the cases that arise in most data sets.

To overcome these problems, this thesis presents a second rule-based ap-

proach where rules are tailored to the initial alignment and are generated automatically using machine learning techniques. This new rule-based approach, *Alignment Link Projection (ALP)*, reduces the number of alignment errors by categorizing the errors made by an initial alignment system. ALP starts with an initial alignment and then fills out (i.e., *projects*) new word-level alignment relations (i.e., *links*) from existing alignment relations. ALP then deletes certain alignment links associated with common errors, thus improving precision and recall.

ALP adapts transformation-based error-driven learning (TBL) (Brill, 1993) to the problem of word alignment. Following the TBL formalism, ALP attempts to find an ordered list of transformation rules (within a pre-specified search space) to improve a baseline annotation. The transformation rules decompose the search space into a set of consecutive words (windows) within which alignment links are added to, or deleted from, the initial alignment. This window-based approach exploits the clustering tendency of alignment links, i.e., when there is a link between two words, there is frequently another link in close proximity.

As shown in Figure 1.2, ALP starts with an unannotated parallel corpus (which might be enriched with linguistic features such as POS tags), a ground-truth alignment for this corpus, and a set of rule templates. The first step is word-level alignment of the corpus using an initial annotator, which is usually an existing word alignment system, to obtain a word-aligned (annotated) corpus. Next, for each rule template, ALP goes over the corpus and finds the words that satisfy the condition of the template. If the template is applicable to the words in question,

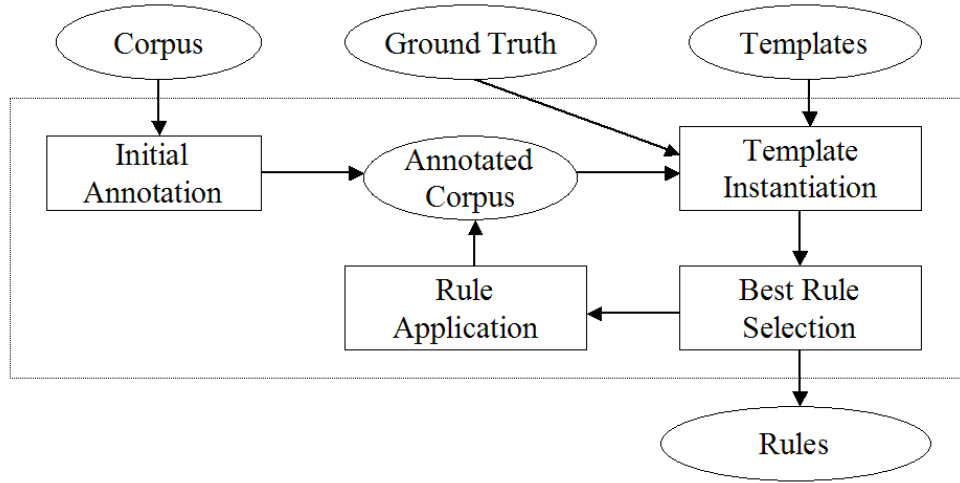


Figure 1.2: Alignment Link Projection (ALP)

a new rule is generated by instantiating the constituents of the template with features of the words. As part of the template instantiation process, ALP records whether the action taken by the rule results in a correct or incorrect alignment link by comparing it against the ground-truth.

After all the templates are instantiated with all possible values, ALP decides which rule results in the best score by applying the rule to a copy of the current state of the annotated corpus, and evaluating against the ground truth. If the best rule results in a higher score than the previous alignment, ALP updates the final state of the annotated corpus by applying the rule. This process of instantiating templates, finding the best rule, and comparing its application against the ground truth is repeated until the best rule found in the current iteration yields a lower score than the previous alignment, or when the maximum number of rule applications is satisfied.

The initial templates consider consecutive words (of size 1, 2 or 3) in both languages. The condition portion of a rule template tests for the existence of an alignment link between two words. The action portion involves the addition or deletion of an alignment link. For example, assuming an alignment link between the words e_i and f_j is represented by (i, j) , the rule template,

Condition: $(NULL, j), (i, j + 1)$

Rewrite rule: add (i, j)

is applicable only when a word (e_i) in one language is aligned to the second word (f_{j+1}) of a phrase (f_j, f_{j+1}) in the other language, and the first word (f_j) of the phrase is unaligned in the initial alignment. The action taken by this rule template is to add a link between e_i and f_j .

ALP employs three different sets of templates to project new alignment links or delete existing links in a given alignment:

1. Expansion of the initial alignment according to another alignment,
2. Deletion of spurious alignment links, and
3. Correction of multi-word (one-to-many or many-to-one) correspondences.

ALP requires only a small amount of manually aligned data for extracting transformation rules—a major strength of the system. Any existing word-alignment system may be used for the initial annotation. Note that these initial aligners are treated as black boxes in the ALP framework. The instantiation of the

templates is based on linguistic features of words such as POS tags, dependency relations, and semantic-based word classes, rather than the words themselves. Using these features is what sets ALP apart from existing combination approaches, such as the *refined method* (Och and Ney, 2000b; Koehn et al., 2003). The selection of best rules may be accomplished using two different metrics: The accuracy of the rule or the overall impact of the application of the rule on the entire data.

As shown in Chapter 4, ALP provides significant improvements over various initial aligners and, moreover, the improvements are not specific to any language pair. Alignment link projection approach yields a significantly lower alignment error rate than that of the best performing alignment system (22.6% relative reduction on English- Spanish data and 23.2% relative reduction on English-Chinese data).

1.2.2 Combining Multiple Alignments

The second alignment framework combines existing word alignments into an improved alignment by taking advantage of the merits of different systems rather than building a new system from scratch. This method relies on the concept of classifier ensembles and attempts to reduce the number of alignment errors by cross validating a given alignment system against another one. Classifier ensembles have been shown to perform better than individual systems in several machine learning applications, but have never been used on word alignments.

At the core of the multiple-alignment framework is an engine called *Multi-*

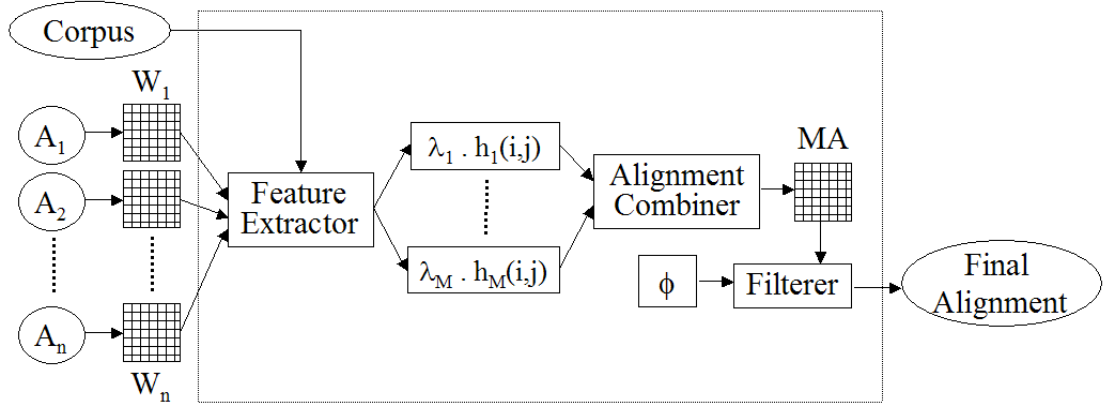


Figure 1.3: Multi-Align Framework for Combining Multiple Alignments

Align, which treats each alignment system as a classifier, and then the alignment combination problem is transformed into a classifier ensemble problem. Multi-Align combines any number of word alignment systems, regardless of the assumptions the individual aligners make or the resources they employ. One of the motivations behind the Multi-Align is to be able to incorporate linguistic knowledge into the alignment modeling. As described in Section 2.2, different pieces of linguistic knowledge, such as POS tags and dependency trees, have been shown to improve word alignments. Multi-Align provides a mechanism for combining linguistically-informed alignment approaches with statistical aligners without the need for complex modifications to existing systems.

Figure 1.3 illustrates the Multi-Align design. In this framework, first, n different word-alignment systems, A_1, \dots, A_n , generate word alignments between a given English sentence and a FL sentence. Then a *Feature Extractor* takes the output of these alignment systems and the parallel corpus (which might be enriched

with linguistic features) and extracts a set of feature functions based on linguistic properties of the words and the input alignments. Each feature function h_m is associated with a model parameter λ_m . Next, an *Alignment Combiner* uses this information to generate a single word-alignment matrix based on the extracted feature functions and the model parameters associated with them. The contribution of each feature function to a particular alignment link is proportional to the model parameter associated with that feature. In a final step, a *Filterer* filters the alignment links according to their confidence in the final alignment matrix. The decision to include a particular alignment link in the final alignment is based on a confidence threshold ϕ .

Parameters of Multi-Align can be set manually or learned via machine learning algorithms. To illustrate the generality of the framework, these parameters are set manually to implement three alignment combination approaches: intersection, union, and a refined alignment approach called grow-diag-final (Koehn et al., 2003).

Later, two different methods are presented for learning the model parameters automatically using machine-learning techniques. The first method is a combination module based on weighted summation of the model parameters and feature functions. A simple but effective method based on (single-layer) perceptrons (Rosenblatt, 1958) is used to learn the model parameters and the confidence threshold. Perceptrons enable word-alignment improvements, but they are only capable of solving problems where a linear solution exists. Unfortunately, word alignment

is not a linearly separable problem.

To overcome this problem, the Multi-Align framework has been implemented as a neural-network based classifier-ensemble approach called *NeurAlign*. Neural nets with two or more layers and non-linear activation functions are capable of learning any function of the feature space with arbitrarily small error and they have been shown to be effective for combining classifiers (Hansen and Salamon, 1990). Neural nets have been shown to be effective especially with (1) high-dimensional input vectors, (2) relatively sparse data, and (3) noisy data with high within-class variability, all of which apply to the word alignment problem.

Results presented in Chapter 6 indicate that, even with only two input alignments, *NeurAlign* yields a significant 28-39% relative error reduction over the best of the input alignment systems and a significant 20-34% relative error reduction over the best known alignment combination technique on English-Spanish and English-Chinese data. Also, using as many input alignments as possible produces improved word alignments.

Multi-Align is a general enough framework to be easily tunable to applications where alignments are utilized. Users are able to tune parameters to control the effects of input alignment systems or additional resources. Moreover, Multi-Align is an easy-to-adapt, robust framework that takes advantage of already existing word alignment systems.

The impact of improved word alignments on machine translation is also investigated in this thesis. As shown in Chapter 7, improved word alignments lead

to a better MT system, based on automated MT evaluation metrics.

1.3 Contributions

This thesis yields the following contributions:

- Error analysis on word alignments based on linguistic properties of the words, such as POS tags, dependency relations, and semantic-based classes.
- Introduction and implementation of a word alignment improvement module that targets translation divergences.
- A novel word alignment correction system that characterizes alignment errors systematically using machine learning, and improves word alignments by correcting frequently occurring errors.
- The first use of classifier ensembles on word alignment problem and introduction of a new framework for combining different word alignments, which might take as many aligners as possible as input, regardless of their underlying model and resources they employ.
- The easy integration of linguistic knowledge into statistical models without the need for large modifications to existing word alignment systems.
- The implementation of two neural-network based alignment combination algorithms to improve word alignments within the combination framework.

- The analysis of alignments that elucidates the sources of improvements by one alignment over another.
- The investigation of the impact of improved word alignment on machine translation (specifically on a phrase-based MT).

Although this thesis focuses on improving word alignments, it also contributes to other fields in Computer Science by:

1. Demonstrating that error-driven learning techniques are useful for improving existing systems and that they can be used easily to eliminate the need for handling each system separately.
2. Introducing an application of classifier ensembles in a new field and demonstrating that classifier ensembles perform better than individual classifiers in this new field.
3. Demonstrating that neural networks can be successfully and easily applied to a new domain.
4. Demonstrating that linguistic knowledge can be easily integrated into existing systems without changing the internals of existing systems.

1.4 Thesis Layout & Brief Overview of Chapters

This thesis includes 8 chapters as briefly described below:

- Chapter 2 presents earlier word-alignment work. The well-known IBM models and HMM-based alignment systems are described, as well as different techniques to extend or combine these models. Evaluation of word alignments is also discussed.
- Chapter 3 presents a novel method, DUSter, to identify translation divergences, i.e., structural differences, between languages, and to correct word alignments related to these divergences using linguistically motivated rules.
- Chapter 4 presents a new technique, ALP, to systematically categorize the errors made by an initial alignment system, and to correct those errors for an improved word alignment. The technique of transformation-based error-driven learning is adapted to the problem of learning word alignments, and then the learned patterns are used to project new alignment links from existing links.
- Chapter 5 first presents a brief survey on ensembles of classifiers, which is the core of the work presented in this thesis, and its application to NLP problems. A new and general framework, Multi-Align, is introduced that combines outputs of different word alignment systems. A simple but effective combination method based on linear weighting of the outputs of individual aligners is described.
- Chapter 6 presents a new method, NeurAlign, to combine different word alignments using an additional more complex model. A brief overview of

neural networks—the core of the combination technique—is given. An investigation into how neural networks can be used effectively for generating an ensemble of word alignments is presented.

- Chapter 7 presents an analysis of improvements using various measures, and investigates the effects of improved word alignments in the context of another application, specifically phrase-based machine translation.
- Chapter 8 concludes with overall observations and conclusions, limitations of the methods, and future directions.

Chapter 2

Related Work

Parallel texts are texts accompanied by their translation in one or more languages. History abounds with parallel texts, such as contracts, treaties, sacred writings, literature, dating from just about every period and involving nearly every pair of languages. In many cases, the parallelism was only virtual (in today's terms) because the texts and their translations were not written on the same physical medium. The first known source of parallel texts is the Rosetta stone, which conveyed the honors presented to King Ptolemy V by the temples of Egypt, in two languages (Greek and Egyptian) and three writing systems (Véronis, 2000).

The first few attempts at using parallel texts, such as machine translation in the late Fifties, were limited by storage and computing capacities of computers, along with the difficulty of creating electronic texts. Between 1970 and 1990, two groups (Bell Communications Research, and the IBM T. J. Watson Research Center) collected a French-English corpus containing over fifty million words taken from transcriptions of debates in the Canadian Parliament, which is known as

“The Hansards.” In addition to a verbatim record of the proceedings and its translation, the Hansards include session numbers, names of speakers, time stamps, question numbers, and indications of the original language in which each speech was delivered. Consequently, this corpus became a *de facto* gold standard for developing and testing systems, opening the way for parallel text processing. The incredibly rapid growth of the Web and development of several techniques to collect data from the Internet has led to a huge increase in parallel resources in electronic form (Resnik, 1998; Koehn, 2002; Resnik and Smith, 2003). Since then, parallel texts have become one of the most commonly used resources in natural language processing.

The most critical task in parallel text processing is *alignment*, i.e., detection of corresponding units between two texts of different languages. Given parallel texts in two different languages, alignment can be extracted on a level of sections, paragraphs, sentences, phrases, collocations, or words. Other logical approaches involve aligning parse trees of a sentence and its translation (Matsumoto et al., 1993; Meyers et al., 1996), or simultaneously generating parse trees and alignment arrangements (Wu, 1995).

The rest of this chapter will describe some of the techniques that have been used to solve three alignment problems: Sentence-level alignment, word-level alignment, and phrase-level alignments. The applications that use word alignment are then described. The chapter concludes with a discussion of techniques for evaluating word alignments.

2.1 Sentence-Level Alignment

Alignment of parallel texts at the sentence level is an important problem, and it is usually the first step toward aligning parallel texts at the word level. Due to the success of sentence alignment algorithms on the Hansards corpus in early 90's, sentence alignment is considered a trivial task, but it has been later noticed that it is not an easy task because of the following reasons:

- The electronic texts may contain physical noise, such as OCR errors.
- A single sentence in one language may be translated into two or more sentences in the other language, or the content may be distributed across multiple translated sentences (Wu, 1994).
- A sentence, or even a whole passage, may be missing from one or the other of the texts.
- The texts may have presentational differences such as different places for floating figures and tables, footnotes, headers, different orders in glossaries, etc. (Véronis, 2000).

Various methods have been proposed for solving sentence alignment problems. These methods can be classified into two groups:

1. **Lexical methods:** Word correspondences in the aligned sentences are used (Catizone et al., 1989; Kay and Roscheisen, 1993). Which word in one text corresponds to which word in the other text is based on the similarity of their

distributions, where the similarity measure is usually the Dice coefficient (Dice, 1945).

2. **Statistical methods:** Only internal information given in the parallel text is used, without making any direct assumptions about the lexical content of the sentences (Gale and Church, 1991a; Brown et al., 1991b). The common strategy is to use lengths of sentences (either in number of words or number of characters) to guide the search space.

There has been additional research on enrichment of statistical methods with various linguistic resources, such as dictionaries, lists of cognates¹, etc. (Simard et al., 1992; Church, 1993; Melamed, 1996a; McEnery and Oakes, 1995). When sentence-length correlation between the languages is not high because of different character sets and grammatical and rhetorical difference of the two languages (for example, Chinese and English), lexical knowledge has been shown to be necessary for more accurate sentence alignment (Fung and Church, 1994; Wu, 1994; Haruno and Yamazaki, 1996; Davis et al., 1995; Dagan et al., 1993).

Much of the work on sentence alignment is based on alignment at the word level. For example, the Smooth Injective Map Recognizer (SIMR) first identifies likely points of correspondence between the two texts using translation lexicons or cognates, and then uses these word correspondences to align sentences (Melamed,

¹The term *cognate* refers to pairs of tokens of different languages that share *obvious* phonological or orthographic and semantic properties leading to their use in mutual translations.

1996a). Thus, the problems of word alignment and sentence alignment are interconnected.

Although sentence-level alignment is still an important issue, in this thesis, it is assumed that the parallel texts have been already aligned at the sentence level. This thesis focuses on improving word alignments.

2.2 Word Level Alignment

Word alignment is arguably a more complicated problem than sentence alignment. Figure 2.1 shows an alignment of words between the English sentence *She will fear her enemies* and the Spanish sentence *Ella tendrá miedo de sus enemigos*. The alignments may be one-to-many (*fear* is aligned with three Spanish words: *tendrá miedo de*) and many-to-one (*will fear* is aligned with the Spanish verb *tendrá*). It is often difficult to decide, even for a human, just which words in an original are responsible for a given one in a translation. Moreover, some words apparently translate morphological or syntactic phenomena rather than other words, i.e., *functional words*. However, it is relatively easy to establish correspondences between such words as proper nouns and technical terms, so that partial alignment at the word level is often possible (Véronis, 2000).

Some of the issues causing problems for word alignment are mentioned in (Kay and Roscheisen, 1993; Ker and Chang, 1997; Véronis, 2000):

- Words are not always aligned one-to-one because some languages are wordier

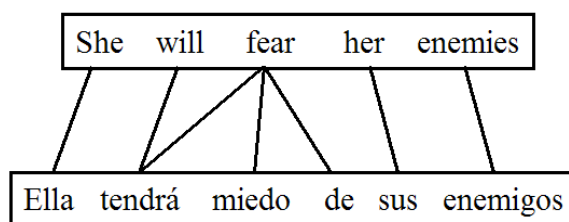


Figure 2.1: Word Alignment Example

than others.

- Some languages make morphological distinctions that are absent in the other (German, for example, makes a number of case distinctions, especially in adjectives, that are not reflected in the morphology of English.)
- Paraphrased and free translations: For various reasons, such as language typology, style, and cultural differences, a translator does not always translate literally on a word by word basis. Most of the time, words are added or deleted.
- There is a high percentage of function words (about 50% of the tokens in any text), for which there is even less of a one-to-one correspondence than for content words. Function words frequently translate into an affix, positional information, part of expressions, phrases or even nothing at all.

In addition, the same problems that arise with sentence alignment arise for word alignments because many sentence alignment methods use (often partial) word alignments as anchor points (Kay and Roscheisen, 1993; Dagan et al., 1993; Fung and Church, 1994). A variety of techniques, including bootstrapping and

relaxation, perform the two types of alignment at the same time. In fact, the *bitext-correspondence*² problem has often been viewed as just one instance of the more general *translation analysis* problem, which is known to be AI-complete (Isabelle et al., 1993). In other words, solving the bitext correspondence problem is no easier than any significant AI problem.

The first issue is robustness: As pointed out in (Church, 1993), “real texts are noisy.” For instance, parts of the source text are omitted in the target text or end up in a different order. The second issue is accuracy: even when the texts are clean, there are hard decisions to make for the alignment (Simard and Plamondon, 1998). During sentence alignment, word alignment is not the primary goal so the methods are usually error-tolerant to noise in the text. In contrast, when the primary goal is word alignment, one can no longer settle for rough and partly erroneous alignments. For this purpose, various researchers have focused directly on filtering out noise in alignments and extractions (Dagan and Church, 1994; Melamed, 1996b; Resnik and Melamed, 1997; Jones and Alexa, 1997).

Given two sentences $\mathbf{e} = e_1, \dots, e_i, \dots, e_I$ and $\mathbf{f} = f_1, \dots, f_j, \dots, f_J$ that are translations of each other, there are $I \times J$ different connections that can be drawn between \mathbf{e} and \mathbf{f} because each of the J words in \mathbf{f} can be connected to any of the I words in \mathbf{e} . Since an alignment is determined by the connections that it contains,

²A bitext correspondence is defined as a pair of tokens on two sides of the text that can be translated to each other. The unit of correspondence need not to be words but can also occur in the form of multiple words (Melamed, 1996a).

and since a subset of the possible connections can be chosen in $2^{I \times J}$ ways, there are $2^{I \times J}$ possible alignments. Different word alignment models reduce the search space by making further assumptions on types of connections between the words. For example, in IBM Model 4 (Brown et al., 1993), each source word f_j can align to exactly one target word e_i , or the null word. On the other hand, the target words can link to any number of source words. As a result, the number of possible alignments is $O(I^J)$, which is still exponential.

Word alignment techniques fall into two groups: Similarity-based heuristic methods, and statistical (or corpus-based) methods. In the rest of this section, both approaches are described in detail.

2.2.1 Heuristic Methods

The common strategy is to find similarities between words in two texts using some correlation measures based on co-occurrence counts. The key idea for robust statistical modeling is not only looking at where two words co-occur, but also looking at the places where only one of the two occurs or the places neither occurs. In one of the early studies, the correlation measure ϕ^2 was useful for this purpose (Gale and Church, 1991b).³

$$\phi^2(w_1, w_2) = \frac{(ad - bc)^2}{(a + b)(a + c)(b + c)(c + d)}$$

³According to (Melamed, 2000), the G^2 statistic suggested by (Dunning, 1993) slightly outperforms ϕ^2 .

where a is the number of places where w_1 and w_2 co-occur, b is the number of places where w_1 occurs but not w_2 , c is the number of places where w_2 occurs but not w_1 , and d is the number of places neither occurs. The ϕ^2 measure is used progressively to align words with each other. That is, on a portion of the corpus, ϕ^2 is computed for all pairs occurring in this corpus. After selecting the best pairs, the same procedure is repeated for another subset of corpus, enhancing the set of best pairs at each iteration. The final set of best pairs is used to align the words that are in the pair.

The Smooth Injective Map Recognizer (SIMR) is a greedy algorithm for finding bitext correspondence (Melamed, 1996a). SIMR produces sentence-level alignments, but it makes use of word alignment techniques as a component of the algorithm. SIMR relies on the high correlation between the lengths of mutual translations, as in (Gale and Church, 1991a; Brown et al., 1991b), and infers bitext maps from likely points of correspondence between the two texts (Church, 1993). Unlike previous methods, SIMR searches for a handful of points of correspondence at a time, by choosing only those points whose geometric arrangement most resembles the typical arrangement of true points of correspondence (TPC). This selection involves localized pattern recognition heuristics for guessing whether a given point in the bitext space is a TPC. A translation lexicon or a set of cognates can be used for this decision. Melamed claims that a set of TPCs leads to alignment more directly than a translation model or a translation lexicon because TPCs are a relation between token instances, not between token types. Another advantage

of SIMR is that it handles inversions and word-order differences. Moreover, a set of correspondence points, supplemented with sentence boundary information, can be used to express sentence correspondence.

It has been demonstrated that SIMR can easily be ported to a new language pair and is applicable to any text genre in any pair of languages (Melamed, 1997a). SIMR’s output quality is highly influenced by the coverage of the translation lexicon used to find correspondence points. In subsequent studies, Melamed extended SIMR to eliminate the problem of *indirect associations*, i.e., frequent co-occurrences of words that are not translations of each other, by taking into account several features of the tokens such as POS information, word ordering, word classes, and existence in another bilingual lexicon, using simple heuristics (Melamed, 1997b, 2000). The proposed model generates translation lexicons with precision and recall both exceeding 90%.

In a recent study (Probst and Brown, 2002), the effects of the dependence on the translation lexicon are examined. A simple bilingual lexicon is enhanced by taking advantage of inflectional and derivational morphology. Improved dictionaries were shown to yield statistically significant improvements in the quality of word alignment.

In other approaches (Simard and Plamondon, 1998), the robustness of character based methods such as *char_align* (Church, 1993) and the accuracy of lexical-based methods, such as (Chen, 1993; Dagan et al., 1993), are combined. This is a two-step strategy: first compute a bitext map, working on robustness rather than

accuracy, and second use this map to constrain the search space for the computation of sentence alignment, this time relying on a method that favors accuracy over robustness or efficiency. The initial bitext mapping is computed using a program that is called *Jacal* (Just another cognate alignment program) that is similar in flavor to Church’s *char_align* and Melamed’s SIMR (Melamed, 1996a). One difference is that, instead of using translation lexicon, *Jacal* uses isolated cognates, which results in an improvement in robustness. The results are comparable with other methods for Hansards corpus but not quite accurate for other corpora. Simard claims that this is caused by the segmentation errors in the corpus.

Similarity-based approaches on their own are usually successful for frequent words that are consistently aligned to one translation. However, words that are less frequent or exhibit diverse translations generally do not have statistically significant evidence for a confident alignment, resulting in incomplete or incorrect alignments. Ker presents a word alignment algorithm based on thesaurus classification and relies on an automatic procedure to acquire class-based alignment rules (Ker and Chang, 1997). The basic motivation is that a word’s translational deviation is mostly bound within the relevant semantic classes. The word classes are adopted from the categories in Roget’s thesaurus.⁴ Experimental results indicate that a classification based on existing thesauri is highly effective in broadening coverage (over 80%) while maintaining a high precision rate.

⁴Word classes can also be obtained by using parts-of-speech classes (Chang and Chen, 1994).

2.2.2 Statistical Methods

At the core of any statistical machine translation system is a word-level alignment model. The IBM models (Brown et al., 1993) were among the first statistical alignment approaches, in which the goal was to translate a foreign-language (FL) text into English, i.e., a FL string $\mathbf{f} = f_1, \dots, f_j, \dots, f_J$ is translated into an English string $\mathbf{e} = e_1, \dots, e_i, \dots, e_I$. Among all possible English translations, the one with the highest probability $\hat{\mathbf{e}}$ is chosen by an application of Bayes' decision rule:

$$\hat{\mathbf{e}} = \operatorname{argmax}_{\mathbf{e}} \{p(\mathbf{e}|\mathbf{f})\} = \operatorname{argmax}_{\mathbf{e}} \{p(\mathbf{e}) \cdot p(\mathbf{f}|\mathbf{e})\}$$

$P(\mathbf{e})$ is called the language model and $P(\mathbf{f}|\mathbf{e})$ is the string translation model. The *argmax* operation can be characterized as a search problem. A key issue in modeling the string translation probability $p(\mathbf{f}|\mathbf{e})$ is the definition of correspondence between the words of the English sentence and the words of the foreign language. Typically, a pairwise dependence is inferred by considering all possible pairs of words. Models describing these types of dependencies are referred to as *alignment models*.

Five different IBM-style alignment models were designed—IBM Model 1, IBM Model 2, etc. (Brown et al., 1993)—to assign a probability to each of the possible word-by-word alignments. The models use minimal linguistic information so they are applicable to any language pair as long as a sufficient parallel text is available.

The IBM models are constrained by assigning each FL word to at most one

English word. A hidden alignment variable $\mathbf{a} = a_1^J$ is introduced, where each a_j takes a value between 0 and I , where I and J are the lengths of English and FL sentence, respectively. The value of a_j represents the position of the target word e_{a_j} to which the source word f_j corresponds, i.e., if the word in position j of the FL string is connected to the word in position i of the English string, then $a_j = i$, and if it is not connected to any English word, then $a_j = 0$. Using this hidden alignment variable, the likelihood of $p(\mathbf{f}|\mathbf{e})$ is written as follows:

$$p(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

A Viterbi alignment $\hat{\mathbf{a}}$ of a specific model is an alignment that maximizes $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$:

$$\hat{\mathbf{a}} = \operatorname{argmax}_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

Model 1 assumes that alignments are independent of each other, and the distribution is uniform. In this case,

$$\begin{aligned} p(\mathbf{f}|\mathbf{e}) &= \sum_{\mathbf{a}} \prod_{j=1}^J p(a_j) p(f_j|e_{a_j}) \\ &= \frac{\epsilon}{(I+1)^J} \prod_{j=1}^J \sum_{i=0}^I p(f_j|e_i) \end{aligned}$$

It is easy to show that for Model 1, the most likely alignment \hat{a} of e and f is given by:

$$\hat{\mathbf{a}} = \operatorname{argmax}_{\mathbf{a}} \prod_{j=1}^J p(f_j|e_{a_j})$$

Since the alignments are independent of each other, the most likely alignment \hat{a} can be found by choosing the value for a_j that leads to the highest value for $p(f_j|e_{a_j})$, for each j .

Model 2 discards the uniform alignment distribution by enforcing a_j to be dependent on the position of the words, and also the lengths of the sentences.

$$p(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} \prod_{j=1}^J p(a_j|j, I, J) p(f_j|e_{a_j})$$

Models 1 and 2 capture some interesting correspondences between words, but they are usually incapable of connecting one word to multiple words successfully. Models 3 and 4 introduce an additional fertility model $p(\phi_i|e_i)$, describing the number of words aligned to the English word e_i , and a distortion model $p(d(j|i, I, J))$, which describes the movement of words. In Model 3, for instance,

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \binom{J - \phi_0}{\phi_0} \times p_0^{J-2\phi_0} p_1^{\phi_0} \times \prod_{i=1}^I \phi_i! p(\phi_i|e_i) \times \prod_{j=1}^J p(f_j|e_{a_j}) d(j|a_j, I, J)$$

with $\sum_f p(f|e) = 1$, $\sum_j d(j|a_j, I, J) = 1$, $\sum_\phi p(\phi|e) = 1$, and $p_0 + p_1 = 1$.

Model 5 was designed to address the deficiency problem, i.e., the problem of alignment to an increasingly higher number of empty words on each iteration in Models 3 and 4, and was shown to be better than previous models.⁵ However, Model 4 performs nearly as well as Model 5, with a much lower computation cost. The key ideas of each model are summarized in Table 2.1 (taken from (Och and Ney, 2000b)).

⁵The alignment to empty words arises when the model assigns a high probability on generalized strings, i.e., strings where some positions are aligned with several words and others with none.

IBM-1:	All alignments have the same probability.
IBM-2:	A zero-order alignment model $p(a_j j, I, J)$ is applied, where different alignment positions are independent from each other.
IBM-3:	An inverted zero-order alignment model $p(j a_j, I, J)$, with an additional fertility model $p(\phi e)$, which describes the number of words ϕ aligned to an English word e , is applied.
IBM-4:	An inverted first-order alignment model $p(j j')$ with an additional fertility model $p(\phi e)$ is applied.
IBM-5:	A reformulation of IBM-4 with a refined alignment model to address the deficiency problem.

Table 2.1: Summary of IBM Models

IBM models are later extended by learning word classes automatically and using those classes instead of the actual words during modeling (Och and Weber, 1998). Researchers have also investigated enhancements to the IBM-3 and IBM-4 models (Och and Ney, 2000b) for the purpose of addressing the *deficiency problem*. Another problem with the IBM models is that they consider only one direction in the translation model, i.e., $p(f_j|e_i)$. Zens et al. (2004) improved the IBM models using a symmetric translation model by applying a linear and log-linear interpolation to the probabilities in two directions. The same study also proposes a discounted smoothing technique for an improved translation model to overcome the data sparseness problem seen in highly inflected languages like German. The goal is to smooth the lexicon probabilities of the full-form words using a probability distribution that is estimated using the word base forms, i.e., *stems*.

Another improvement to IBM models is the addition of context dependencies using a maximum entropy (ME) model, which is directly integrated into the EM training (García-Varea et al., 2002). The goal is to include more dependencies,

i.e., a larger context, in the translation model rather than just using $p(f_j|e_{a_j})$. In a ME framework, the properties of e that are deemed to be useful are described in the form of feature functions $\phi_{e,k}(x, f)$, where x is the context of e . For example, the absence or existence of a specific word e'_k in the context of an English word e , which can be translated as f'_k , can be represented by the following feature function:

$$\phi_{e,k}(x, f) = \begin{cases} 1 & \text{if } f = f'_k \text{ and } e'_k \in x \\ 0 & \text{otherwise} \end{cases}$$

Assuming that $p_e(f|x)$ represents the probability of the ME model associated with e assigns to f in the context of x , and the feature functions for a specific word e are represented by $\phi_{e,k}(x, f) : k = 1, \dots, K_e$,

$$p_e(f|x) = \frac{1}{Z_{\Lambda_e}(x)} \exp \left(\sum_{k=1}^{K_e} \lambda_{e,k} \cdot \phi_{e,k}(x, f) \right)$$

Here, $\Lambda_e = \{\lambda_{e,1}, \dots, \lambda_{e,K_e}\}$ and $Z_{\Lambda_e}(x)$ is a normalization factor. The parameter values $\lambda_{e,k}$ that will maximize the likelihood for a given training corpus can be optimized using generalized iterative scaling (Darroch and Ratchiff, 1972) or improved iterative scaling (Berger et al., 1996). Although a different ME model should be learned for each target word e , this process is usually subject to a thresholding. The context of a word e is defined by three words to the left and three words to the right of e . In addition to dependence on the actual words, the approach also models dependence on the word classes. Alignment error rate is reduced with respect to IBM models.

Moore (2004) describes two limitations of Model 1 that are not deeply struc-

tural but that can be addressed merely by changing how the parameters of Model 1 are estimated. The first of these nonstructural problems is that rare words in the source language tend to act as garbage collectors (Brown et al., 1993; Och et al., 2004), aligning to too many words in the target language. The second nonstructural problem is that it seems to align too few target words to the null source word. To address the first problem of rare words aligning to too many words, at each iteration of EM, all the translation probability estimates are smoothed by adding virtual counts according to a uniform probability distribution over all target words. The second problem is addressed by adding extra null words to each source sentence and multiplying all the translation probabilities for the null word by the number of null words per sentence. Moore (2004) also proposes a heuristic model based on the log-likelihood ratio statistic recommended by Dunning (1993), which has also been shown to be useful for other applications (Melamed, 2000). The goal is to improve Model 1 alignment accuracy by starting with a better set of initial parameter estimates. These three simple modifications achieve excellent results, which are nearly comparable to those of the more complex IBM models.

These five models were re-implemented during the JHU Workshop in 1999, and the resulting package, GIZA, has become a classic in the MT literature (Al-Onaizan et al., 1999). Later, GIZA is improved by addressing several problems in the implementation and deficiencies of the models (described below). Since then, the resulting software, GIZA++ (Och, 2000), has become the state-of-the-art statistical alignment package.

An alternative and widely used alignment model is a hidden Markov model in which alignment probabilities rely on the differences in the alignment positions rather than on the absolute positions (Vogel et al., 1996). The basic motivation is the existence of strong localization effects in aligning the words in parallel texts, especially for language pairs from Indo-European languages, where words are not distributed arbitrarily over the sentence positions, but tend to form clusters. For some languages (German, for example), there is an even stronger restriction because the difference in the position index may be smaller than 3. The basic idea behind the HMM-based alignment model is that the English word that is aligned to a FL word is dependent on the English word that the previous FL is aligned with.⁶ Formally,

$$p(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} \prod_{j=1}^J p(a_j|a_{j-1})p(f_j|e_{a_j})$$

HMM-based alignment models, like IBM models, also suffer the problem of aligning each source word to at most one target word.

An alternative HMM-based alignment model is adopted that makes use of the following observation (Tillmann et al., 1997): Over large portions of the source string, the alignment is monotone. For the alignment model, the monotonicity property allows only transitions from a_{j-1} to a_j with a jump width δ . If $\delta = 0$, a target word is aligned with two or more source words. If $\delta = 1$, a single new target

⁶If f_j is aligned with e_i , then $a_j = i$. The HMM model states that a_j is dependent on the previous alignment a_{j-1} .

word is generated. If $\delta = 2$, there is a word in the target string with no aligned word in the source string.

The HMM-based alignment model has been further extended (Och and Ney, 2000a) in three different ways:

1. Use of equivalence classes over both languages for a refined alignment model.
2. Empty word handling: the HMM network is extended by I empty words, i.e., the English word e_i has a corresponding empty word e_{i+I} .
3. Smoothing: $p'(a_j|a_{j-1}, I) = \alpha \cdot \frac{1}{I} + (1 - \alpha) \cdot p(a_j|a_{j-1}, I)$

The results show that more sophisticated alignment models are crucial for reduction of word-alignment error. Consistently, the use of a first-order alignment model, modeling an empty word and fertilities result in better alignments.

Toutanova et al. (2002) reported further improvements on word alignment by extending HMMs as follows:

1. Incorporating POS tag information of the source and target languages in the translation model,
2. Approximately modeling fertility, and
3. Better modeling of the alignments to NULL target word.

HMMs provide good quality alignments, better than IBM Models 1-3 and

comparable to Model 4, despite the simplicity of the model.⁷ A log-linear combination of the HMM and IBM Model 4, which was recently introduced as Model 6, was shown to improve alignments further (Och and Ney, 2003), but at a higher computational cost.

The complexity of the word alignment problem can be reduced by enforcing syntactic bracketing constraints between two sentences using an Inversion Transduction Grammar (ITG) (Wu, 1997). Reordering of the words is modeled by binary branching trees using rules that reverse of the order of the words. The trees were generated by synchronously parsing a parallel corpus, using a simple grammar where one non-terminal symbol generates either a terminal symbol or 2 non-terminal symbols (by either keeping the order of the words same or reversing the order). Generation of a terminal symbol is guided by lexical translation probabilities at the leaves. Inversion transduction grammar cannot handle complex syntactic structures such as noun phrases or verb phrases, but they reduce the complexity of the word alignments to polynomial-time. Later, Zhao and Vogel (2003) extended bilingual bracketing approach to enforce additional constraints on the structure of one language by using English POS tags and base noun phrase boundaries. In a more recent study, ITGs were extended to condition the grammar

⁷In fact, standard implementations of IBM Model 4 (e.g., GIZA++) use HMMs to obtain initial alignments. Lopez and Resnik (2005) demonstrated that their implementations of Model 4 derived most of their performance benefits from an underlying HMM, and the improvements by Model 4 itself were relatively small compared to improvements by the HMM.

production probabilities on lexical information throughout the tree. This study demonstrated that lexicalization yields further improvements on word alignment (Zhang and Gildea, 2005).

Yamada and Knight (2001) used a similar model by transforming a source-language parse tree into a target-language string by applying stochastic operations, which capture linguistic differences such as word order and case marking, at each node. As a result, the space of possible word alignments is constrained by the structure of input trees. The input trees are generated by an existing parser, and the output of the model is a string, not a parse tree. Therefore, parsing is only needed on the channel input side. The tree transformation is achieved by three operations:

1. Reordering: intended to model translation between languages with different word orders, such as SVO-languages (English or Chinese) and SOV-languages (Japanese or Turkish).
2. Word insertion: intended to capture linguistic differences in specifying syntactic cases (for example, English and French use structural position to specify case, while Japanese and Korean use case-marker particles).
3. Translating leaf words

The model parameters are estimated using the EM algorithm, and the resulting model produces word alignments that are better than those produced by IBM Model 5. Later, Gildea (2003) extended tree-to-string alignment model by intro-

ducing a *clone* operation, which copies an entire subtree of the source language syntactic structure and moves it anywhere in the target language sentence, to handle structural divergences between languages (Dorr, 1994) and free translations in the training data. The same work also introduced a new tree-to-tree alignment model, where the space of possible alignments were constrained by parse trees on both sides rather than only on the source side. In order to provide enough flexibility, a single node on the source tree might produce two nodes on the target tree or two nodes in the source tree may be grouped together to produce a single node in the target tree. The tree-to-tree model was trained on parallel treebanks; thus, they are limited by the size of training data. Although the tree-to-tree model did not provide significant improvements over unstructured IBM models, they brought a huge savings in computational complexity. In a later study, Gildea (2004) investigated whether dependency trees were more useful than constituency trees but found, interestingly, that the latter significantly outperformed the dependency tree model.

An empirical comparison of ITGs and tree-to-string models revealed that ITGs significantly outperformed the latter (Zhang and Gildea, 2004). Using trees has proven fruitful for improving alignments but a data-derived tree structure (i.e., ITG) gives better results than projecting automatic English parser output onto the Chinese string (i.e., tree-to-string model). Zhang and Gildea (2004) claim that tree-to-string models perform poorly because of parsing errors and non-isomorphism of parse trees due to structural differences between languages.

A primary drawback of the generative alignment models is that incorporation of arbitrary (whether linguistic or not) features of the data is difficult. The most common strategy to overcome this problem is to represent words with a set of features and weights associated with them, and then model the alignment based on these features and weights. For example, in an early study, associations between two words are viewed as word alignment clues, which can be estimated from association measures on a training data (Tiedemann, 2003). Alignment clues are assumed to be independent of each other, and they can be computed based on arbitrary feature functions between the words. Possible alignment clues include co-occurrence (Dice coefficient), string similarity, and translation probabilities extracted from a pre-aligned data according to features of the words such as POS tags, phrase categories, word positions, and any other kind of contextual features. Each alignment clue $C_k(e_i, f_j)$ is associated with a probability $p(a_k)$. The overall clue is represented as $C_{all}(e_i, f_j) = p(a_{all}) = p(a_1 \cup a_2 \cup \dots \cup a_n)$. For instance, given only two clues, $C_{all}(e_i, f_j) = p(a_1) + p(a_2) - p(a_1 \cap a_2)$. The clues are assumed to be mutually independent; therefore, $C_{all}(e_i, f_j) = p(a_1) + p(a_2) - p(a_1) \cdot p(a_2)$. Once a clue confidence matrix is obtained by computing the overall clue for each pair of words, the best alignment is extracted using a dynamic programming approach.

Similarly, in a recent study, for each pair of words e_i and f_j , a confidence score $s(e_i, f_j)$ is computed using a set of features (Taskar et al., 2005). Then, based on these confidence scores, a word alignment problem is viewed as a maximum weighted bipartite matching problem, where nodes correspond to the words in the

two sentences. The best alignment is taken to be the one that maximizes the sum of edge scores in the graph satisfying some constraints, such as one-to-one alignment. The score of each link is a weighted sum of the features, and weights are learned using large-margin estimation via support vector machines on a small manually-aligned data. The feature set for a pair of words e_i and f_j includes

1. The Dice coefficient between e_i and f_j ,
2. Absolute difference between i and j ,
3. Word-similarity features such as cognateness, substring matching, etc.,
4. Word-frequency features, and
5. IBM Model 4 alignment outputs.

Moore (2005) presents another example of discriminative training. The word alignment is modeled as a weighted sum of several features, and the alignment with the highest sum is selected as the word alignment for that sentence pair. Formally,

$$\hat{a} = \underset{\mathbf{a}}{\operatorname{argmax}} \sum_{i=1}^M \lambda_i h_i(\mathbf{a}, \mathbf{e}, \mathbf{f})$$

where h_i represents a feature and λ_i corresponds to the weight associated with the feature h_i . The weights of the features are computed using a version of weighted perceptron learning (Collins, 2002). This model is very similar to the model in (Taskar et al., 2005): Instead of using dice coefficients and absolute position differences as features and using support vector machines to compute weights, Moore

(2005) uses log-likelihood ratio and non-monotonicity measures as features and averaged perceptron learning to compute weights. This discriminative model achieves results comparable to more complex IBM models. The biggest advantages of the model are its low computational complexity and ease of integrating different pieces of linguistic knowledge into alignment modeling.

As an alternative to IBM and HMM models where the alignment is a hidden variable, the alignment can be modeled directly using a different decomposition of the terms (Cherry and Lin, 2003; Ittycheriah and Roukos, 2005; Liu et al., 2005). In this framework, the alignment problem is defined as finding the alignment \mathbf{a} that maximizes $p(\mathbf{a}|\mathbf{e}, \mathbf{f})$ instead of finding Viterbi alignment in IBM models.⁸

Cherry and Lin (2003) define an alignment \mathbf{a} to be a set of t links $\{l_1, \dots, l_t\}$, where each $l_k = (e_{i_k}, f_{j_k})$ for some i_k and j_k . The consecutive subsets of \mathbf{a} is denoted by $l_i^j = \{l_i, l_{i+1}, \dots, l_j\}$. Given this notation, $p(\mathbf{a}|\mathbf{e}, \mathbf{f})$ can be decomposed as follows:

$$\begin{aligned} p(\mathbf{a}|\mathbf{e}, \mathbf{f}) &= p(l_1^t|\mathbf{e}, \mathbf{f}) \\ &= \prod_{k=1}^t p(l_k|\mathbf{e}, \mathbf{f}, l_1^{k-1}) \end{aligned}$$

Assuming the context of the k th link is $C_k = \{\mathbf{e}, \mathbf{f}, l_1^{k-1}\}$, the term in the product can be decomposed into two: 1) A link probability given a co-occurrence of two words, and 2) A context probability given a link divided by context probability

⁸IBM and HMM models search for the alignment that maximizes $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$, which is equivalent to maximizing $p(\mathbf{a}|\mathbf{e}, \mathbf{f})$.

given a co-occurrence of two words. Formally,

$$p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = \prod_{k=1}^t p(l_k|e_{i_k}, f_{j_k}) \frac{p(C_k|l_k)}{p(C_k|e_{i_k}, f_{j_k})}$$

Unfortunately, $C_k = \{\mathbf{e}, \mathbf{f}, l_1^{k-1}\}$ is too complex to estimate context probabilities directly. Suppose FT_k is a set of context-related features such that $p(l_k|C_k)$ can be approximated by $p(l_k|e_{i_k}, f_{j_k}, FT_k)$. Assuming, for all $ft \in FT_k$, ft is independent of l_k and e_{i_k}, f_{j_k} , the problem is reduced to:

$$p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = \prod_{k=1}^t \left(p(l_k|e_{i_k}, f_{j_k}) \times \prod_{ft \in FT_k} \frac{p(ft|l_k)}{p(ft|e_{i_k}, f_{j_k})} \right)$$

Unfortunately, this alignment model has very few factors to prevent undesirable alignments, such as having all source words align to the same target word. To guide the search process over all possible alignments, a one-to-one constraint, that requires each word to participate in exactly one alignment link, and a cohesion constraint, that is used to restrict possible link combinations by using a dependency tree in English, are employed. Two types of context-specific features are fed to the system: (1) Adjacency features force close-proximity source-language words to be closer in the target language; and (2) Dependency features capture regularities among grammatical relationships between languages, using a dependency tree. The biggest advantage of this model is that new information can be incorporated modularly by adding features, which is similar to the motivation behind maximum entropy (ME) models. In fact, Cherry and Lin (2003) claimed that estimates of $p(l_k|e_{i_k}, f_{j_k}, C_k)$ can be improved using the ME model.

Recently, Ittycheriah and Roukos (2005) followed this idea and proposed a ME-based alignment model, using a direct formulation of alignments, as in Cherry and Lin (2003). Similar to IBM models, it is assumed that each source word is generated by a target word; therefore, each alignment includes J links, where the value of a_j represents the position of the target word e_{a_j} to which the source word f_j corresponds. Then,

$$\begin{aligned}
p(\mathbf{a}|\mathbf{e}, \mathbf{f}) &= p(a_1^J|\mathbf{e}, \mathbf{f}) \\
&= \prod_{j=1}^J p(a_j|\mathbf{e}, \mathbf{f}, a_1^{j-1}) \\
&= \frac{1}{Z} \prod_{j=1}^J p(a_j|a_{j-1})^\alpha \times p(a_j|\mathbf{e}, \mathbf{f}, a_1^{j-1})^{1-\alpha}
\end{aligned}$$

The first term in the product corresponds to the transition model, which tends to keep alignment links close to each other. The second term is the observation model that measures the linkage of source and target words using a set of feature functions defined on the words and their context.⁹ The context of a target word e_i is defined as the words e_1, \dots, e_i , and Wordnet synsets associated with e_i . Five sets of feature functions are employed:

1. Lexical features, which are similar to IBM Model 1 probability estimates,
2. Segmentation features,
3. Wordnet features,

⁹The formulation of the observation model is similar to ME model in García-Varea et al. (2002), so it will not be repeated here.

4. Spelling correction features, and

5. Dynamic features, which are related to already established alignment links.

The values of the feature functions are extracted from manually-aligned data (nearly 10K sentence pairs). Ittycheriah and Roukos (2005) reported significant improvements over IBM models and HMM-based alignments.

Another attempt to directly model $p(\mathbf{a}|\mathbf{e}, \mathbf{f})$ also uses ME model, but differs from Ittycheriah and Roukos’s approach by modeling alignment directly as a log-linear combination of feature functions (Liu et al., 2005). In ME framework, there are M feature functions $h_m(\mathbf{a}, \mathbf{e}, \mathbf{f})$, $m = 1, \dots, M$. For each feature function, there exists a model parameter λ_m , $m = 1, \dots, M$. The direct alignment probability is given by

$$p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = \frac{\exp(\sum_{m=1}^M \lambda_m \cdot h_m(\mathbf{a}, \mathbf{e}, \mathbf{f}))}{\sum_{\mathbf{a}'} \exp(\sum_{m=1}^M \lambda_m \cdot h_m(\mathbf{a}', \mathbf{e}, \mathbf{f}))}$$

The best alignment $\hat{\mathbf{a}}$ is given by

$$\hat{\mathbf{a}} = \operatorname{argmax}_{\mathbf{a}} \left(\sum_{m=1}^M \lambda_m \cdot h_m(\mathbf{a}, \mathbf{e}, \mathbf{f}) \right)$$

In order to incorporate a new dependency that contains extra information other than the bilingual sentence pair, the model above is modified by introducing a new variable \mathbf{v} , which corresponds to set of additional dependencies:

$$p(\mathbf{a}|\mathbf{e}, \mathbf{f}, \mathbf{v}) = \frac{\exp(\sum_{m=1}^M \lambda_m \cdot h_m(\mathbf{a}, \mathbf{e}, \mathbf{f}, \mathbf{v}))}{\sum_{\mathbf{a}'} \exp(\sum_{m=1}^M \lambda_m \cdot h_m(\mathbf{a}', \mathbf{e}, \mathbf{f}, \mathbf{v}))}$$

$$\hat{\mathbf{a}} = \operatorname{argmax}_{\mathbf{a}} \left(\sum_{m=1}^M \lambda_m \cdot h_m(\mathbf{a}, \mathbf{e}, \mathbf{f}, \mathbf{v}) \right)$$

Any dependency between the aligned words can be represented as a feature function in this framework. Liu et al. (2005) used IBM Model 3 alignment probabilities (in both directions), POS tags transition probabilities, and bilingual dictionary coverage as their feature set. Once the model parameters are learned using GIS (Darroch and Ratcliff, 1972), a greedy search is applied to find the best alignment using a *gain* function of adding a link to the set of existing links. It has been shown that log-linear combination of the mentioned features yield a significant improvement over IBM models. However, most of the improvement comes from using only IBM-3 model probabilities.

Another approach to word alignment is the transformation of the problem into orthogonal non-negative matrix factorization based on a probabilistic model of the alignment data (Goutte et al., 2004). By introducing a set of hidden nodes \mathbf{h} between two sentences \mathbf{e} and \mathbf{f} , the alignment of e and \mathbf{f} is viewed as a product of alignment of \mathbf{e} to \mathbf{h} and alignment of \mathbf{h} to \mathbf{f} .

In order to use matrix factorization, the following constraints are enforced:

1. Coverage: Every word on either side must be aligned to at least one word on the other side, possibly taking NULL words into account.
2. Transitive closure: If f_j is aligned to e_i and e_k , any f_l aligned to e_k must also be aligned to e_i .

Transitive closure is satisfied by connecting each word to a node in the hidden layer. Coverage is guaranteed by ensuring that each word is connected to at least

one node in the hidden layer. For example, general M-N alignments correspond to alignment of multiple English words to one node in the hidden layer, and then alignment of that hidden node to multiple FL words.

An alignment between \mathbf{e} and \mathbf{f} may be represented by a $J \times I$ alignment matrix $A = [a_{ji}]$, such that $a_{ji} > 0$ if f_j is aligned to e_i and $a_{ji} = 0$ otherwise. Similarly, given K hidden nodes, alignment of words to hidden nodes may be represented by a $J \times K$ matrix F and a $I \times K$ matrix E , with positive elements indicating existence of alignment links. An alignment can be represented as $A = F \times E^T$. Assuming the translation matrix is represented by an $J \times I$ matrix $M = [m_{ji}]$, where $m_{ji} \geq 0$ measures the strength of the association between f_j and e_i , finding a suitable alignment matrix A corresponds to finding factors F and E such that:

$$M \approx F \times S \times E^T$$

S corresponds to a diagonal $K \times K$ scaling matrix that gives different weights to hidden nodes. Since F and E contain only positive elements, this is an instance of non-negative matrix factorization.

To guarantee proper alignments, each word is associated to exactly one hidden node, and each hidden node is associated to at least one word on each side. Given this propriety constraint, F and E are orthogonal matrices. Finding the best alignment starting from M therefore reduces the alignment problem to performing a decomposition into orthogonal non-negative factors.

Another commonly-used technique to improve alignments is preprocessing

the training data. The motivation is that an input source text that is closer to the target text in terms of length and word order would be useful for increasing word alignment accuracy. This assumption serves as the basis for approaches that transform texts by grouping some words into some classes, joining/splitting words, and reordering some words (Tillmann et al., 1997). This technique has been shown to produce better alignments than using the original (untransformed) training data.

In another study, the effects of lemmatization¹⁰ and dictionary usage in the training data were investigated for IBM Model 4 (Dejean et al., 2003). Applying the lemmatizer reduces the parameter space for the alignment algorithm by reducing the vocabulary size. Adding bilingual lexicon entries was intended to introduce bias for the alignment model parameters. Interestingly, applying the lemmatizer to all data reduces the performance. When lemmatization is applied only to rare words (with varying thresholds), it performs better. Adding dictionary entries did not improve the alignment quality much although slight improvements were obtained in some sets (Och and Ney (2000b) also reported similar results).

2.3 Phrase Level Alignment

Another rapidly growing research area is segment alignment, i.e., alignment of contiguous textual units more than one word but shorter than a full sentence.

¹⁰Lemmatization is splitting the words into their root forms and suffixes.

Examples of segments are clauses, phrases, syntax tree fragments, and skeleton sentences. An alignment of this type would be very useful for a variety of applications including example-based translation, language teaching, and comparative linguistics. However, the problem is extremely hard to solve due to the difficulty of detecting language-specific clause boundaries, the complexity of partial syntactic analysis, and substantial structural differences across languages, even related ones (Véronis, 2000).

Phrasal (or multi-word) alignment is an important capability because it provides a mechanism for characterizing certain types of differences between languages. For example, in English many technical terms are multi-word compounds, while the corresponding terms in other Germanic languages are often single-word compounds. Also, many common adverbials and prepositions are multi-word units, which may or may not be translated as such (for example, English-Swedish). The usability of bilingual concordances would be greatly improved if phrases could be looked up with the same ease and precision as single words (Macklovitch and Hannan, 1996). However, this is not the case in reality, and identifying phrases and aligning them is crucial for an improved alignment quality.

Alignment of multiple-word expressions or phrases has been investigated by various researchers. Some approaches use preprocessing only on the source side (Melamed, 1997c; Smadja et al., 1996); target correspondences are then estimated during the linking stage. In other approaches, both the source and target texts are pre-processed independently and candidate lists for both source and target multi-

word units are created to be used in the linking process (Ahrenberg et al., 1998; Och and Weber, 1998). Phrase-level correspondences are obtained by coupling phrases of two languages obtained by CKY parsing (Kaji et al., 1992). An iterative method based on the Dice coefficient has been investigated which yields high recall if partially correct translations were included (Kitamura and Matsumoto, 1996). Similarly, a method involving Dice coefficient has been used to align collocations between English and French (Smadja et al., 1996).

One of the best performing phrase alignment systems is a two-level alignment model that systematically considers whole phrases rather than single words as the basis for the alignment models (Och et al., 1999). In the first level, source sentence \mathbf{f} and the target sentence \mathbf{e} are decomposed into a sequence of phrases $\mathbf{v}_\mathbf{f} = 1, \dots, K_f$ and $\mathbf{v}_\mathbf{e} = 1, \dots, K_e$. The translation probability is based on these phrases $\mathbf{v}_\mathbf{e}$ and $\mathbf{v}_\mathbf{f}$ instead of actual words \mathbf{e} and \mathbf{f} . In the second level, the translation of shallow phrases is modeled using alignment templates based on an automatic method for extraction of bilingual classes (Och, 1999). The use of classes instead of actual words has the advantage of a better generalization. This alignment template approach produces better translation results than the single-word based approach.

In a recent study, an HMM-based probabilistic model of word-to-phrase alignment has been proposed (Deng and Byrne, 2005). In this generative model, each target sentence $\mathbf{f} = f_1^J$ is realized as a sequence of K phrases: $\mathbf{f} = v_1^K$. Each phrase in f is generated as a translation of one word in e , which is determined by the alignment sequence $a_1^K : e_{a_k} \rightarrow v_k$. The length of each phrase is specified

by the process ϕ_1^K , which is constrained so that $\sum_{k=1}^K \phi_1^k = J$. To allow phrases to be generated by a NULL word, a binary hallucination sequence h_1^K is defined such that if $h_k = 0$, NULL word generates v_k , otherwise e_{a_k} generates v_k . With all these quantities gathered into an alignment $\mathbf{a} = (\phi_1^K, a_1^K, h_1^K, K)$, the modeling objective is to realize the conditional distribution $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$. With the assumption that $p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = 0$ if $f \neq v_1^K$:

$$\begin{aligned}
p(\mathbf{f}, \mathbf{a}|\mathbf{e}) &= p(v_1^K, \phi_1^K, a_1^K, h_1^K, K) \\
&= \epsilon(J|I) \times && \text{sentence length} \\
&\quad p(K|I, J) \times && \text{phrase count} \\
&\quad p(a_1^K, \phi_1^K, h_1^K|K, I, J) \times && \text{word to phrase alignment} \\
&\quad p(v_1^K|a_1^K, \phi_1^K, h_1^K, K, I, J) && \text{word to phrase translation}
\end{aligned}$$

In other approaches, texts are assumed to have structure at many different levels (character, word or partially ordered bag of lexical units). In one such approach (Ahrenberg et al., 1998), an automatic phrase-extraction program provides bilingual input to a system that makes use of a combination of *K-Vec* (Fung and Church, 1994) and a greedy word-to-word algorithm (Melamed, 1997b) to provide the final alignment. The basic algorithm associated with this approach is enhanced by a number of modules:

1. A morphological module that groups expressions that are identical except a specified set of suffixes,
2. A weight module that adjusts the likelihood of a candidate translation ac-

cording to its position in the sentence, and

3. A phrase module that includes multi-word expressions generated in the pre-processing stage as candidate expressions for alignment.

Multi-word expressions are linked with a relatively high recall, but the precision of these links is not as high as for single words.

Other work on phrase alignment includes structural matching of phrasal texts (Matsumoto et al., 1993), phrasal translation example extraction (Wu, 1995), identification of phrasal sequences (Wang, 1998; Och et al., 1999), and synchronous parsing of two texts for phrase identification and alignment, including collocations and recurrent strings, (Wu, 1997; Alshawhi et al., 2000; Bangalore and Riccardi, 2001).

2.4 Combining Word Alignments

One of the major problems with the IBM models (Brown et al., 1993) and the HMM models (Vogel et al., 1996) is that they are restricted to the alignment of each source-language word to at most one target-language word. For example, while aligning German-English texts, if German is the source language the frequently occurring German word compounds cannot be aligned correctly, as they typically correspond to two or more English words. The standard method to overcome this problem is to use the model in both directions (interchanging the source and target languages) and applying heuristic-based combination techniques to produce

a *refined alignment* (Och and Ney, 2000b; Koehn et al., 2003). The motivation behind the refined alignment method is that the alignment links form clusters for most language pairs, i.e., when there is an alignment link, there is usually another one in its neighborhood.

For combining two sets of alignments A_1 and A_2 , the straightforward solutions are to take the union, i.e., $A = A_1 \cup A_2$, and intersection of the two alignments, i.e., $A = A_1 \cap A_2$. In addition to these two methods, Och and Ney (2000b) introduced a *refined alignment* to combine two sets of alignments, A_1 and A_2 . In a first step, the intersection $A = A_1 \cap A_2$ is determined. Then, A is extended by adding alignment links (i, j) occurring only in A_1 or A_2 if

1. Neither f_j nor e_i has an alignment in A , OR
2. If both of the following conditions hold:
 - The alignment (i, j) has a horizontal neighbor, i.e., $(i - 1, j), (i + 1, j), (i, j - 1), (i, j + 1)$, that is already in A .
 - The set $A \cup (i, j)$ does not contain alignments with both horizontal and vertical neighbors.

In a later study, Koehn et al. (2003) proposed another refined alignment approach called *grow-diag-final*. This particular method extends the intersection of two alignments A_1 and A_2 by adding alignment links (i, j) occurring only in A_1 or A_2 if:

1. Diag-step: Neither f_j nor e_i has an alignment in A , AND the alignment link (i, j) has a neighbor in A . In contrast to Och and Ney’s refined alignment method, diagonal adjacencies are considered part of the neighborhood.
2. Final-step (after no more links can be added in Diag-step): Either f_j or e_i does not have an alignment in A .

Of these 3 methods, the intersection of the two alignments consists of only one-to-one alignment links, and yields a high precision and a low recall alignment. Alternatively, the union of the two alignments yields a high recall and a low precision alignment. The refined alignment method is an attempt to find a balance between the intersection and the union, by starting with more reliable alignment links and then adding less reliable but highly probable alignment links (i.e., links that occur in one of the alignments and have another alignment link in its neighborhood). As stated by Och and Ney (2003), whether a higher precision or a higher recall is preferred depends on the final application for which the word alignment is intended. For example, in statistical machine translation, a higher recall is deemed to be more important. On the other hand, a higher precision would be preferred in lexicography applications. Koehn et al. (2003) claimed that choosing the right heuristic to refine alignments is more important than the choice of model for the initial word alignments. In this thesis, *grow-diag-final* is used as baseline in Chapters 3, 4, 5, and 6. Throughout this thesis, the notation *Aligner(gdf)*, e.g., GIZA++(gdf), represents the alignment generated by the grow-diag-final method

on the outputs of *Aligner* in two different directions (i.e., the first input to grow-diag-final method is the alignment generated by *Aligner* using English as the source language, and the second input is the alignment generated by *Aligner* using FL as the source language).

A recent attempt to symmetrize two alignments obtained by training the same model in two different directions considers the alignment problem as a task of finding the edge cover with minimal costs in a bipartite graph, where the cost of aligning a specific target and source word is computed using the parameters of the input model (Matusov et al., 2004). Once the state occupation probabilities are computed for each input model in both directions, the cost of a link is defined as

$$c_{ij} = \sum_{m=1}^M \lambda_m \cdot h_m$$

where h_m refers to the negated logarithm of state occupation probability for each model and direction. To obtain a more symmetric estimate of the costs, the state occupation probabilities are interpolated loglinearly. For instance, to symmetrize the alignments generated by training a specific model (e.g., IBM Model 4) in two different directions, the following cost function can be used:

$$c_{ij} = \alpha(-\log p_j(i|\mathbf{e}, \mathbf{f})) + (1 - \alpha)(-\log p_i(j|\mathbf{e}, \mathbf{f}))$$

Additional feature functions can be included to compute c_{ij} . Given an $I \times J$ cost matrix C , the best alignment is equivalent to finding a minimum-weight edge cover (or a maximum-weight bipartite matching) in a complete bipartite graph,

where the two node sets of this bipartite graph correspond to the source sentence positions and the target sentence positions, respectively, and the costs of the edges are elements of C . Matusov et al. (2004) reported slightly better results than the heuristic-based combination methods on two different language pairs.

2.5 Applications

Word-level alignment of parallel text corpora is a critical capability for a wide range of NLP applications. In statistical MT, translation models rely directly on word alignments (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2003; Koehn et al., 2003). Extraction (or construction) of bilingual lexicons is a direct application of word alignments, as shown in various studies (Gale and Church, 1991b; Church and Gale, 1991; Fung and Church, 1994; Dagan and Church, 1994; Melamed, 1997c). In addition, word-level alignments are used for:

1. Automatic generation of transfer rules (or mappings) for MT (Menezes and Richardson, 2001; Carbonell et al., 2002);
2. Word-sense disambiguation (Brown et al., 1991a; Gale et al., 1992, 1993; Chang et al., 1996; Diab and Resnik, 2002; Bhattacharya et al., 2004);
3. Projection of resources (such as morphological analyzers, part-of-speech taggers, and parsers) from a resource-rich language into other resource-poor languages (Hwa et al., 2002; Yarowsky et al., 2001); and

4. Cross-language information retrieval (Fluhr, 1995; Oard and Dorr, 1996).

The quality of word-level alignments plays a crucial role in the success of these applications. For example, in statistical MT, it has been shown that improved word alignment directly affects the output quality of statistical MT systems (Och and Ney, 2003; Callison-Burch et al., 2004).¹¹

2.6 Evaluation of Word Alignments

Nearly all word alignment evaluation techniques compare generated word alignments with the manually annotated alignments. However, it is well known in NLP that manually performing a word alignment is a complicated and ambiguous task, even for humans (Melamed, 1998). The major difficulty is that alignments are not always 1-to-1, especially between languages that are structurally different. In fact, it could be argued that, ultimately, text alignment is no easier than the more general problem of natural language understanding (Langlais et al., 1998).

There are various factors that affect word alignment evaluation (Ahrenberg

¹¹It is worth noting that the impact of improved word alignments on MT quality is a subject of debate. Other researchers have proposed techniques that improved word alignments in terms of alignment evaluation metrics but the impact of those improved alignments on MT output have been shown to be relatively smaller (Koehn et al., 2003; Ittycheriah and Roukos, 2005). In Chapter 7 of this thesis, it will be shown that the alignment improvement techniques yield a huge error reduction on word alignments but very little improvement on MT output in terms of the well-known BLEU metric.

et al., 2000; Merkel and Ahrenberg, 1999):

1. The purpose of the alignment system.
2. Unit of comparison. (Is it a word-level comparison or are multi-word units included? Are all words included or are some set of words excluded, such as functional words or frequent words?)
3. Resources used.
4. The use of a gold standard. (Is this prepared before the actual alignment or do the experts evaluate a sample of the output after the alignment? Is a complete alignment of the sample generated or are only a subset of words/phrases aligned? What is the size of the gold standard and what is the distribution of the samples?)
5. Metrics and scoring method. (How are partial alignments measured?)
6. Error analysis. (What is the nature of the mistakes that a particular system makes?)

Once the word alignments are produced manually, it is easier to evaluate the quality of any word alignment with respect to this reference set. This evaluation can be performed automatically and it results in a very precise and reliable evaluation criterion. If the text only consists of single words, the straightforward solution is to use the usual precision and recall metrics in information retrieval, by viewing word alignment as a retrieval problem (i.e., find all correspondences

at the lexical level that exist in a given parallel text or corpus). The precision gives the proportion of segments in the proposed alignment that is considered to be correct. The recall gives the coverage of the alignments in the gold standard by the proposed alignment. However, these metrics may not be suitable when alignments are not one-to-one, for instance when collocations are involved, as well as deletions, insertions, segmentation errors and paraphrases. In a simple approach, the scoring for precision and recall can be adjusted to handle partial alignments by using weighted score (Merkel and Ahrenberg, 1999).

Some approaches to evaluation of word alignment are:

- Measuring results relative to a gold standard, using either a sample of continuous text (Melamed, 1998) or spot checks (Véronis, 1998).
- Evaluating the type of links created by full text alignment system as a bilingual dictionary and measuring recall and precision based on a sample of this dictionary (Ahrenberg et al., 1998).
- Measuring precision only on the highest ranked n candidates suggested by the system (Kitamura and Matsumoto, 1996; Gaussier, 1998).

The state-of-the-art method to evaluate word alignments is measuring precision and recall on the level of alignment links (pairs of word indices) instead of words (Langlais et al., 1998; Simard and Plamondon, 1998; Véronis, 1998). Assuming the alignment $A = \{a_l, a_2, \dots, a_m\}$ and the reference alignment $G = \{g_l, g_2, \dots, g_n\}$, where a_i and g_i is a pair of word indices, precision and recall can

be computed at the level of alignment links. The advantage of this approach is that partial alignments of multi-words are rewarded.

A widely-used strategy is dividing the alignments into two sets: Probable (P) alignments and Sure (S) alignments. The P relation is used especially to align words within idiomatic expressions, free translations, and missing function words. Note that it is assumed that $S \subseteq P$. Using A and G as the alignment and reference alignment, $A = A_P \cup A_S$ and $G = G_P \cup G_S$. Similar measures of recall and precision are defined (Mihalcea and Pedersen, 2003), where T is the alignment type (either P or S):

$$\begin{aligned} Recall_T &= \frac{|A_T \cap G_T|}{|G_T|} \\ Precision_T &= \frac{|A_T \cap G_T|}{|A_T|} \\ Fscore_T &= \frac{2 \times P_T \times R_T}{P_T + R_T} \end{aligned}$$

Under the same settings, alignment error ratio (AER) is defined as follows (Och and Ney, 2000a):

$$AER = 1 - \frac{|A \cap G_S| + |A \cap G_P|}{|A| + |G_S|}$$

When there is only one category of alignments in the gold standard G (i.e., no distinction is made between “sure” and “probable” alignments), the formula can be generalized to the following:

$$\begin{aligned} AER &= 1 - \frac{2 \times |A \cap G|}{|A| + |G|} \\ &= 1 - Fscore \end{aligned}$$

AER is heavily biased toward sure alignments. If an alignment A gets exactly the same set of sure alignments in the gold standard, it will yield an AER of 0, even if it misses a lot of links in the probable alignments. Therefore, if the majority of the links in the gold standard is probable, it might not be a good idea to evaluate word alignment using AER, especially if the recall of the alignments is important for the end-application (Goutte et al., 2004).

One of the important decisions in word alignment is whether to include or exclude links with unaligned words (i.e., links where one word is aligned to NULL word). In one setting, each word is enforced to belong to at least one alignment. If a word does not belong to any alignment, a NULL Probable alignment is assigned by default. This set of evaluations pertains to full coverage word alignments. In another setting, all NULL alignments are removed from both A and G (Mihalcea and Pedersen, 2003).

It has been common practice in the NLP community to evaluate without NULL alignments, and to report precision of probable alignments, recall of sure alignments, and the alignment error rate. This is the strategy adopted for this thesis, as we will see in Chapters 3, 4, 5, and 6.

The evaluation metrics discussed above are useful for studying the accuracy of an alignment. However, it may be more informative to evaluate these in a larger context. While the figures obtained by these metrics are informative, more telling figures can only be obtained by measuring the effect of the alignment system on some specific task (Ahrenberg et al., 1998). For example, word alignments have

been evaluated by computing their effect on bilingual lexicon extraction (Véronis, 1998; Ahrenberg et al., 2000). Another approach is to evaluate statistical alignments by looking at the quality of the corresponding MT output. Word alignments can also be used to project parse trees from one language to another language (Yarowsky et al., 2001). In this case, word alignments can be evaluated based on projected tree comparisons against a hand treebanked corpus (Goodman, 1996; Hwa et al., 2002) or on parser output after training a parser on the projected trees (Carroll et al., 1998). In Chapter 7 of this thesis, word alignments are evaluated within a phrase-based machine translation system to investigate the effects of improved alignments inside another application.

2.7 Discussion

As described in Chapter 1, current statistical word alignment systems are faced with five important challenges:

1. Complexity of the word alignment problem,
2. Lack of enough training data (in the face of limited linguistic resources),
3. Learning statistics correctly,
4. Translation divergences, and
5. Lack of a means for incremental incorporation of linguistic knowledge.

The rest of this section outlines how these five challenges have contributed to a number of shortcomings in existing alignment systems.

The complexity of the word alignment problem forces statistical systems to constrain the search space of hypotheses by employing certain assumptions, or *biases*, about the hypothesis to be learned from the training data. For example, the IBM models, and their variations, (Brown et al., 1993; Och and Ney, 2003) and HMM models (Vogel et al., 1996; Tillmann et al., 1997) restrict source language words to be aligned to at most one target word. Other approaches take this restriction even further by allowing only one-to-one alignments (Cherry and Lin, 2003; Goutte et al., 2004). As a result of their deficiencies in modeling alignments, these systems fail at generating some subset of true word alignments. Alignment combination approaches eliminate some of the problems related to the biases of existing systems (Och and Ney, 2000b; Koehn et al., 2003; Matusov et al., 2004), but these approaches are based on simple heuristics and often provide little improvement over existing systems.

The lack of linguistically annotated data impairs the ability of alignment systems to capture linguistic generalizations. This has led to the tendency for the development of statistical systems that use large parallel texts without any linguistic knowledge about the languages. Several researchers have shown that more training data yields better word alignments (Och and Ney, 2003; Koehn et al., 2003), yet it is unclear how one determines how much data is sufficient for different language pairs. The recent trend in statistical systems is to incorporate

all the data that is available and let the system take care of redundant or noisy data (Och, 2005). Ultimately, with billions and billions of sentences, this is equivalent to memorizing all possible word (or phrase) correspondences so that there is nothing left unseen for any given test set. In practice, this is nearly impossible to achieve, at least for a majority of language pairs. That is, there will always be language pairs where there is only a limited amount of data. Moreover, statistical systems are still susceptible to their biases in modeling alignments; therefore, it is highly unlikely that they will produce 100% correct alignments even with infinite data.

The rareness of some words—coupled with the too-frequent occurrence of other words in the training data—makes it very difficult to choose what statistics to use (Dunning, 1993; Moore, 2004). Moreover, most expressions in the languages are only “semi-frozen” and can still undergo a number of linguistic operations (such as inflection, insertion of adjectives and adverbs, conversion to the passive voice, etc.). For example, one of the major problems with the IBM models (Brown et al., 1993) and HMM models (Vogel et al., 1996) is their tendency to align rare words to several words on the other side in an attempt to reduce the number of unaligned words. The problem with the function words is more dramatic: Statistical systems are often unable to pick up the correct word correspondences for function words because function words occur in every sentence several times and statistical models are based on co-occurrence statistics. Such shortcomings of the models can be addressed by changing the way the alignments are modeled (Moore, 2004; Tillmann et al., 1997; Och and Ney, 2000a). The rareness of words due to morphological

distinctions between languages can be addressed using stemming or data transformation techniques (Tillmann et al., 1997; Dejean et al., 2003). To solve the problem with the alignment of function words, recent studies focus on alignment of phrases rather than words (Marcu and Wong, 2002; Deng and Byrne, 2005). However, the dependence on statistics collected from training data remains to be an issue for all of these approaches.

Translation divergences—structural differences between two languages (Dorr et al., 2002)—have a significant impact on alignment systems. Divergences occur when the underlying meaning of a set of words in one language is distributed over a set of words with different syntactic and lexical semantic properties in the other language, e.g., the English verb *fear* corresponds to *tener miedo de* (*have fear of*) in Spanish. The most common types of alignment errors related to divergences occur when a word in one language is translated into a phrasal structure with the addition or deletion of function words (i.e., conflational, inflational and structural divergences) or into words that have different parts of speech (i.e., categorial divergence). Chapter 3 demonstrates that translationally divergent word pairs are the most significant factor contributing to statistical alignment errors. The current-best approach to handling translation divergences aligns phrases directly rather than aligning words and extracting phrases from word-aligned corpora (Och et al., 1999; Marcu and Wong, 2002; Deng and Byrne, 2005) but these approaches are not very successful at capturing long-distance dependencies.

The lack of a means for incremental incorporation of linguistic knowledge into statistical systems has resulted in a proliferation of ‘build-everything-from-scratch’ approaches. Several researchers have demonstrated that linguistic knowledge such as POS tags, dependency relation, bilingual lexicons, and morphological analyzers improves word alignment (Toutanova et al., 2002; Cherry and Lin, 2003; Ittycheriah and Roukos, 2005). Current approaches to injecting linguistic knowledge into word alignments include changing the model to include dependencies on the linguistic features of the words (Toutanova et al., 2002; Cherry and Lin, 2003) and representing linguistic knowledge as feature functions in a maximum entropy model (García-Varea et al., 2002; Liu et al., 2005; Ittycheriah and Roukos, 2005). There are two problems with this ‘build-everything-from-scratch’ approach: First, the resulting system may lose valuable information that is inherent in the architecture of previous systems, even when the new system produces better alignments. Second, it is quite difficult to assess the usefulness of different pieces of linguistic knowledge.

The remainder of this thesis addresses these challenges, demonstrating that it is possible to improve alignments by constructing techniques for detection and correction of existing word alignments. Chapter 3 introduces a framework called DUSTer that addresses the challenges of translation divergences and lack of training data. Chapter 4 addresses the first four challenges in a system called ALP that identifies and corrects frequent alignment errors automatically and conditions the alignment rules on linguistic knowledge. In Chapters 5 and 6, the shortcomings of

existing systems due to first 4 challenges above are handled by taking advantage of multiple alignments (Multi-Align) and using linguistic knowledge in the form of feature functions during the combination step (NeurAlign).

Chapter 3

DUSTer: Divergence Unraveling for Statistical MT

This chapter describes the first of two rule-based approaches to the alignment problem, a system called DUSTer (*Divergence Unraveling for Statistical Translation*) that uses parallel texts for divergence unraveling in word-level alignment. DUSTer combines linguistic and statistical knowledge to resolve structural differences between languages, i.e., *translation divergences*, during the process of alignment. The goal is to improve word-level alignments produced by existing state-of-the-art statistical systems using linguistic knowledge.

Early alignment algorithms incorporated linguistic cues to address standard alignment issues, e.g., one-to-many/many-to-one mappings, but such systems were often narrow in their coverage. Statistical algorithms (Och and Ney, 2003) are more broadly applicable, requiring only that a large parallel corpus exists for the languages in question, but they are not able to accommodate certain types of MT phenomena. For instance, statistical aligners are incapable of handling com-

plex phrasal constructions because dependencies are not captured between non-consecutive, widely distributed words.

At the heart of the problem is the lack of linguistic knowledge about structural differences between languages, e.g., English verb *fear* vs. Spanish *tener miedo de* (*have fear of*). A statistical word aligner usually aligns *fear* to *miedo*, leaving the main verb *tener* unaligned, because *tener* and *de* are frequent enough in the Spanish corpus to be aligned to a variety of other high-frequency words. Learning statistics from a huge corpus is not sufficient to align words correctly in such cases. In DUSTer, this deficiency is addressed by relating one or more linguistically-motivated categories associated with the (English) input words to those of another language (henceforth, foreign language—FL); the resulting *match sets* are used to infer corrected alignments.

DUSTer employs a set of rules to identify and handle translation divergences. The rules utilize the dependency relationships between words, POS tags of the words, and a set of related semantic-based classes to identify the places where two sentences are divergent. The application of the rules relies on the existence of alignment links between some words in the rule. If a rule is found to be applicable, then DUSTer adds additional alignment links as indicated by the rule. The rules can be tailored to any language easily once the divergences between two languages are identified.¹

¹DUSTer served as an initial study to test the feasibility of alignment corrections for certain divergence types. The goal in this thesis is to move toward automatic construction of these

The rest of this chapter describes different types of translation divergences and relates them to the word alignment errors made by statistical alignment systems. Then an approach to resolving translation divergences is described. The chapter concludes with a presentation of a set of experiments on English-Spanish data.

3.1 Translation Divergences

Divergences between languages occur when the underlying meaning of a set of words in one language is distributed over a set of words with different syntactic and lexical semantic properties in the other language. The translation divergences can be divided into 5 groups (Dorr et al., 2002):

Categorical Divergence: A categorical divergence is the translation of words in one language into words that have different parts of speech as in the translation of an adjective into a noun. In the following examples, the adjectival phrase is translated into a light verb accompanied by a nominal version of the adjective:

to be jealous \Leftrightarrow tener celos (to have jealousy)

to be fully aware \Leftrightarrow tener plena conciencia (have full awareness)

rules. Chapter 4 presents a machine learning approach to demonstrate how similar rules can be generated automatically.

Conflational/Inflational Divergence: A conflation is the translation of two or more words in one language into one word in another language. Inflation is the reverse image of conflation. Common forms of this divergence type are the light-verb construction and manner conflation. The light-verb construction is the translation of a single verb in one language into a combination of a semantically “light” verb and some other meaning unit (maybe a noun or a preposition). for example,

to kick \Leftrightarrow dar una patada (give a kick)

to end \Leftrightarrow poner fin (put end)

Manner conflation is the translation of a single manner verb (*e.g.*, *float*) in one language into a light verb of motion and a manner in the other language. Some examples in English-Spanish are:

to float \Leftrightarrow ir flotando (go (via) floating)

to pass \Leftrightarrow ir pasando (go passing)

Structural Divergence: A structural divergence is the realization of verb arguments in different syntactic configurations in different languages, e.g., the realization of incorporated arguments (such as subject and object) as obliques (i.e., headed by a preposition in a PP). Some English-Spanish examples are:

to enter the house \Leftrightarrow entrar en la casa (enter **in** the house)

ask **for** a referendum \Leftrightarrow pedir un referandum (ask-for a referendum)

Head Swapping Divergence: Head swapping is the inversion of a structural dominance relation between two semantically equivalent words when translating from one language to another. An example is the demotion of the head verb and the promotion of one of its modifiers to the head position. For example, an English motion verb and a preposition are translated as a directed motion verb and a progressive verb in Spanish:

to run in \Leftrightarrow entrar corriendo (enter running)

fly about \Leftrightarrow andar volando (go-about flying)

Thematic Divergence: A thematic divergence occurs when a verb’s arguments are realized in syntactic configurations that reflect different thematic hierarchies (thematic to syntactic mapping orders). For example, the experiencer and theme of a verb can be realized as subject and object in one language and as dative (object of *to*) and subject in another language.

I like grapes \Leftrightarrow Me gustan uvas (to-me please grapes)

I have a headache \Leftrightarrow Me duele la cabeza (to-me hurt the head)

Previous work on divergences (Dorr et al., 2002; Habash and Dorr, 2002) showed that at least 10% of the sentence pairs in Arabic/English and Spanish/English are divergent in some way. In another study of 2K human-confirmed Spanish/English divergence pairs, 98% pairs contained categorial divergences, 83% contained conflational divergences (Light-Verb and Manner confluations), 35% contained structural divergences, 8% contained head swapping divergences, and 6%

contained thematic divergences. These numbers clearly indicate that different divergence types frequently co-occur. For instance, categorial divergence co-occurs with almost all other divergence types. To illustrate the co-occurrences of divergences, consider the following English and Spanish sentences:

Different politicians please Maria.

Maria tiene gustos de políticos diferentes.

There are four co-occurring divergences in this example: Categorical (*please* is a verb in English but *gusto* is a noun in Spanish), conflational (*please* is conflated to *tener gusto*), thematic (*Maria* and *politicians* switch thematic-to-syntactic realization order), and structural (*politicians* is an argument in English but an oblique in Spanish). This indicates that it is important to handle divergences concurrently rather than tackling each divergence one at a time.

Examples of the type given above are what motivates the idea of divergence unraveling for inducing word-alignment improvements. Divergence handling requires careful attention to distinctions in both content and structure. Historically, MT approaches have achieved this by means of transfer rules (Hye Han et al., 2000; Lavoie et al., 2000) or complex interlinguas (Dorr, 1993). However, these techniques rely heavily on resources that are difficult to obtain, e.g., annotated parallel treebanks, parses, or large complex manually checked lexicons. In divergence-unraveling approach in this chapter, resources on the FL side are kept minimal; knowledge-intensive resources are required only in English.

3.2 Relating Alignment Errors to Divergence Types

This section examines the relation between the alignment errors made by a state-of-the-art alignment system (GIZA++ (Och and Ney, 2003)) and the linguistic categories (syntactic and semantic) associated with translation divergences. The results of this analysis validate the use of a set of linguistically-motivated *universal rules* described in Section 3.3.2.

Table 3.1 presents an alignment example (between English and Spanish) taken from a development set of 99 randomly-selected sentence pairs.² Each row shows all alignment links for a given English word. Two different alignments (one by a human and one by GIZA++) are shown in Spanish. Certain additional features, e.g., part-of-speech (POS) labels and semantic word classes (such as *Psych Verbs* (abbreviated as *PsyV*) and Directional Prepositions (abbreviated as *DirP*)) are also shown here.

Analysis of GIZA++ alignments reveals that all alignment errors fall into one of 3 categories:³

1. **Type 1—Missing Links:** For a given English word e_i , GIZA++ omits one

²The development set was selected from a mixture of Bible and UN Corpus sentence pairs. There is no limitation on sentence length: the length of the English sentences is between 7 and 46 words while the length of Spanish sentences is between 6 and 50 words. The average English sentence length is 25 while the average Spanish sentence length is 27. For all 99 sentences, the pairs of English/Spanish word pairs that are misaligned were extracted for the current analysis.

³ (i, j) is used to represent the alignment link between e_i and f_j .

English Word	Human	GIZA
1. Domenech /Noun	[1. el / <i>FuncD</i> 2. dirigente /Noun]	[1. el / <i>FuncD</i> 2. dirigente /Noun]
2. also /Adv	[3. también /Adv]	[3. también /Adv]
3. complained /Verb	[4. se / <i>FuncN</i> 5. quejó / <i>PsyV</i>]	[5. quejó / <i>PsyV</i>]
4. about / <i>Oblique</i>	[6. de / <i>DirP, Oblique</i>]	[6. de / <i>DirP, Oblique</i>]
5. Argentine /Adj	[8. argentinos /Adj]	[8. argentinos /Adj]
6. officials /Noun	[7. funcionarios /Noun]	[7. funcionarios /Noun]

Table 3.1: Human and GIZA++ Alignments Between an English and Spanish Sentence

or more links (i, j) specified in the human alignment. Alternatively, for a given FL word f_k , GIZA++ omits one or more links (i, k) specified in the human alignment.

2. **Type 2—Added Links:** For a given English word e_i , GIZA++ includes one or more links (i, k) not specified in the human alignment.
3. **Type 3—Replaced Links:** For a given English word e_i , GIZA++ substitutes one or more links (i, j) for all links (i, k) specified in the human alignment.⁴

Figure 3.1 illustrates these alignment errors graphically, where dashed lines indicate human alignment links missed by GIZA++, solid lines indicate human alignment links matched by GIZA++, and crossed-out solid lines indicate GIZA++ alignment links not matched by the human alignment.

⁴Although this error could be viewed as a combination of the missed link in Type 1 errors and the added link in Type 3 errors, it is counted as a separate error type for this analysis.

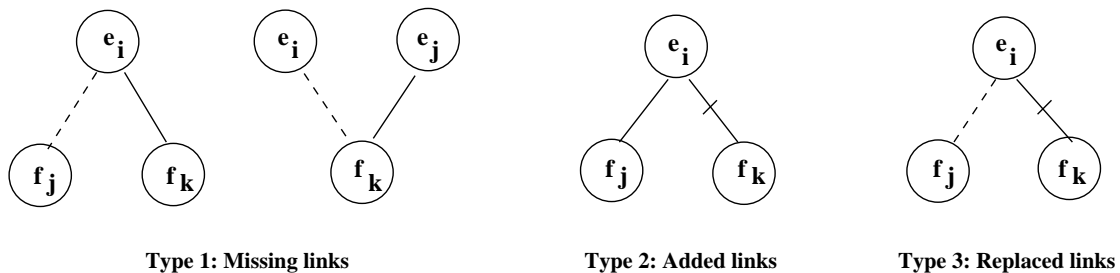


Figure 3.1: Graphical Illustration of Three Types of Alignment Errors

Error Type	Number	Percentage	Example
1. Missing links	833	79.7%	fear → tener miedo
2. Added links	122	11.7%	foreign → relaciones exteriores
3. Replaced links	90	8.6%	stolen → apacientan
Total	1045	100.0%	

Table 3.2: Number and Percentage of Alignment Errors by GIZA++ (on English-Spanish)

Table 3.2 shows the error rate associated with each of these types (for the development set) and their relation to the divergence categories. Missing alignment links are the most frequent alignment error introduced by GIZA++, accounting for 79.7% of the cases. These are precisely the cases that correspond to the divergence classes specified above, most notably conflational, inflational and structural divergences, where a single word in one language corresponds to multiple words in the other language. On the other hand, Added and Replaced alignment links occur in the much rarer cases where the statistical alignment is tripped up, not by issues concerning translation divergences, but by idiosyncratic, domain-specific statistical correspondences from the training corpus.

Further analysis of the errors above allows us to identify the linguistic cat-

Linguistic Category	Missing	Added	Replaced	Example
Adjective	34	14	5	big
Adverb	20	2	3	very
Complement	38	6	4	that
Determiner	11	3	-	the
DirectionP	47	2	4	to
FunctionalDet	99	2	9	a
FunctionalNoun	88	13	4	he
Negation	8	1	-	not
Noun	141	34	14	book
Oblique	134	7	18	on
Pronoun	24	5	5	it
Sem-Class Verbs ⁶	85	17	9	run
Other Verbs	104	15	15	have
TOTAL	833	121	90	

Table 3.3: Semantic and Syntactic Categories of Words from Missing, Added and Replaced Links (on English-Spanish)

egories of words that are most likely to be associated with alignment errors. Table 3.3 provides a count of the semantic classes (Complement, DirectionP, FunctionalDet, FunctionalNoun, Negation, Oblique, Sem-Class Verbs) and POS labels (Adjective, Adverb, Determiner, Noun, Pronoun, Other Verbs) associated with English word(s) involved in missing/added/replaced links.⁵

Table 3.3 suggests the following issues with GIZA++ alignments:

- GIZA++ handles Obliques poorly (most commonly associated with categorical and structural divergences), leaving them unaligned most of the time.

⁵The numbers in Table 3.3 reflect double counting in some cases: If an alignment link that is missed for a word is added to another word, it is counted as one missed and one added link.

⁶Sem-Class Verbs refer to verbs that occur in widely recognized semantically classes, specifically, Aspectual, Change of State, Directional, Light, Locational, Modal, Motion, Psych, and Tense. See Section 3.3.1 for more discussion.

For example, in the translation of *fear* to *tener miedo de*, the Oblique *de* is left unaligned.

- Alignment of verbs (most commonly associated with conflation, head swapping, and thematic divergences) is one of the biggest challenges for GIZA++.

For example, in the translation of *fear* to *tender miedo de*, the Light-Verb *tener* is left unaligned.

- Functional determiners and functional nouns (most commonly associated with categorial and structural divergences) are another source of misalignment that arises quite frequently. For example, in the translation of *complained* to *se quejó*, the Functional-Noun *se* is left unaligned.

The divergence-unraveling approach addresses these issues more specifically.

3.3 DUSter: System Description

This section describes the DUSter system and demonstrates how the system handles divergences effectively in the context of word-level alignment. Previous work demonstrated that mapping words into word classes (Och and Weber, 1998) and grouping/splitting words (Tillmann et al., 1997) are techniques that have proved useful for the tasks of statistical alignment. DUSter takes this one step further by applying a set of general, linguistically-motivated rules to induce alignment improvements.

The key idea behind DUSter is to relate one or more linguistically-motivated

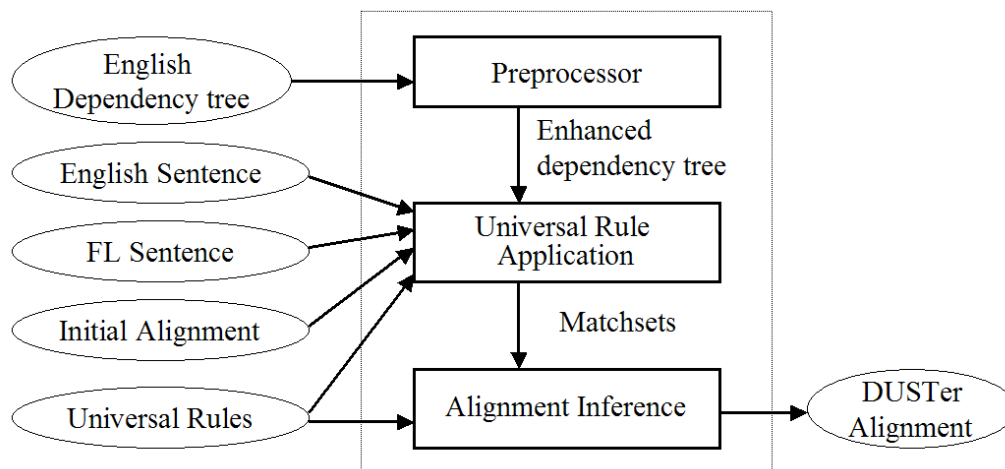


Figure 3.2: Components of DUSTer

categories associated with the English input words to those of the FL; the resulting *match sets* are used to infer corrected alignment links. To achieve this, DUSTer relies heavily on English resources and a set of rules for identifying and resolving common divergence types between languages.

Figure 3.2 shows the major components of DUSTer (inside the dotted rectangle). The input to DUSTer is an English sentence, a foreign language sentence, a word alignment between those two sentences, a dependency tree corresponding to the English sentence, and a set of universal rules for handling translation divergences (which will be described in Section 3.3.2 in detail). The initial alignments may be produced by any existing word alignment system. The dependency tree may be produced by a standard parser.

First, the English dependency tree is preprocessed (adding certain linguistic features of the words) to produce an enhanced dependency tree. After this, the

enhanced tree is passed to a universal rule-application module, along with the original sentences and initial alignment. Application of the rules results in a list of *match sets*, i.e., the indices of words matching both English and the FL components of the universal rules. These match sets are then used to infer a final set of partial alignments. These partial alignments may be combined with alignments produced by other algorithms to induce an overall improved result.

The remainder of this section presents the resources that are necessary for the application of DUSter to a language pair (i.e., the parameters and universal rules) and then illustrates the application of DUSter to alignment correction.

3.3.1 Parameters

DUSter assumes that certain types of words are grouped together into *parameter classes* based on semantic-class knowledge, e.g., classes of verbs including *Aspectual*, *Change of State*, *Directional*, etc. (Levin, 1993). Each parameter is associated with a list of words and a word may be part of more than one parameter. The parameter classes play an important role in identifying and handling translation divergences. The current classification includes 16 classes of parameters. Table 3.4 presents the parameters and some associated English words.

Because the parameters are based on semantic knowledge, the English values can be projected to their corresponding values in a new language. With few exceptions, this can be done simply by translating the words of a parameter class to those of another language. For example, English light verbs (*be*, *do*, *give*, *have*,

Parameter Name	Examples in English
Aspect Verb	<i>begin, complete, end, keep, prevent, quit, repeat</i>
Change of State Verb	<i>age, alter, change, empty, grow, reverse, tighten</i>
Complement	<i>as, because, since, than, that, while, when, yet</i>
Direction Verb	<i>arrive, come, cross, enter, invade, return</i>
Directional Preposition	<i>across, at, down, from, in, to, toward, up</i>
Functional Determiner	<i>a, each, every, his, many, some, that, which</i>
Functional Noun	<i>all, each, every, it, he, him, his, what</i>
Light Verb	<i>be, do, give, have, make, put, take</i>
Location Verb	<i>belong, consist, enclose, insert, put, touch</i>
Modal Verb	<i>can, could, do, have, may, must, should</i>
Motion Verb	<i>bounce, cut, float, jump, kick, move, run, turn</i>
Negation	<i>no, not, none</i>
Oblique	<i>across, by, for, in, of, over, to, under, with</i>
Pleonastic	<i>it, there</i>
Psych Verb	<i>admire, bother, envy, love, mean, respect, try</i>
Tense Verb	<i>shall, will</i>

Table 3.4: DUSTer Parameters in English

make, put, take) are translated to Spanish light verbs (*estar, ser, hacer, dar, tomar, poner, tener*), respectively. Note that it is not necessary to list all morphological variants of the same word in the parameter classes.⁷ For example, in English, the verb *begin* is listed among the Aspectual Verbs, but not all of its variants *begins, began, begun*. Appendix A lists all parameters in English and Spanish.

Different morphological variants are mapped to the root word in a separate module and each variant is treated as a member of the associated parameter class during the execution of the system.

⁷DUSTer considers only inflectional morphology, i.e., derivational morphology is ignored inside DUSTer.

3.3.2 Universal Rule Set

DUSTer makes use of a “universal rule set,” i.e., general rewriting rules that relate one or more linguistically-motivated categories in English—specifically, part-of-speech (POS) labels and semantic word classes—to those of the FL. The term ‘universal’ refers to the structure of the rule and implies that the same rule may be applied to different language pairs that exhibit the same phenomena. For example, the following rules may be used to handle two forms of conflation (Tense-Verb and Light-Verb) between English and 3 other languages:

```
0.AVar.X [English{2 1} Chinese{1} Spanish{1} Hindi{1} ]
```

```
[Verb<1,i> [TenseV<2,Mod,Verb,C:i>]] <-->
```

```
[Verb<1,i>]
```

```
1.B.X [ English{2 1 3} Spanish{2 1 3 4 5} ]
```

```
[PsychV<1,i,CatVar:V_N,Verb> [Noun<2,j,Subj>] [Noun<3,k,Obj>]] <-->
```

```
[LightVB<1,Verb,C:i> [Noun<2,j,Subj>] [Noun<3,i,Obj>]
```

```
[Oblique<4,Pred,Prep,C:i> [Noun<5,k,PObj>]]]
```

These rules correspond to the mappings ‘*will eat*’ \rightarrow ‘*eats*’ and ‘*j fears k*’ \rightarrow ‘*j has fear of k*’, respectively. The first line shows the rule name, the languages to which the rule may be applied, and the relative linear order of the nodes in the surface form for each language involved. The ordering numbers (1, 2, 3, ...) correspond to the node identifiers in the rule specified in the subsequent lines.


```

<rule>          :: <rule_header> <rule_info>

<rule_header>   :: <div_type> [ <language_list> ]
<div_type>      :: type [0-9]+[A-Za-z0-9.]*[A-Za-z][A-Za-z0-9.]*
<language_list> :: <language> { <ordering> }
<ordering>      :: [0-9]+ | [0-9]+ <ordering>

<rule_info>     :: <tree> <--> <tree>
<tree>          :: <node>
<node>          :: [ <node_info> ]
<node_info>     :: <node_type> < <label_list> > | <node_info> <node>
<node_type>     :: <param> | <pos>
<label_list>    :: <node_id> | <node_id>,<labels>
<labels>        :: <label> | <label>,<labels>
<label>         :: <alignment_index> | <pos> | <rel> | <catvar> | Child:~<rel>

```

Figure 3.3: Universal Rules in BNF Format

For example, the second rule specifies the English ordering {2, 1, 3} to indicate that the order of the English surface words is a *Noun* followed by a *Psych Verb* followed by another *Noun*. For the same rule, the Spanish ordering {2, 1, 3, 4, 5} indicates that the order of the Spanish surface words is *Noun*, *Light-Verb*, *Noun*, *Oblique*, and *Noun*. Specifying the order of the words separately from the rule itself allows each rule to be applied to languages that have different word orders. The second and third lines of each rule indicates the English subtree on the left-hand side (LHS) and the foreign language subtree on the right-hand side (RHS), respectively. The BNF specification for a universal rule is presented in Figure 3.3.⁸

Each rule relates the English structure on the LHS of the rule to the FL structure on the RHS of the rule. Square brackets indicate the nesting depth in the overall tree structure, following standard bracketing conventions (Leech et al.,

⁸<language>, <param>, <pos>, <rel>, and <catvar> goes to a terminal symbol that corresponds to a language, parameter, part-of-speech, relation and categorial variation, respectively.

1996). Rule nodes consist of a *node type*—a part-of-speech category (POS) or a parameter type—followed by a list of features. The feature list includes a unique *node identifier* (an integer, e.g., *1*), an *alignment index* that relates potentially aligned LHS and RHS nodes (a letter of the alphabet, e.g., *j*), a POS category (e.g., *Prep*), and a dependency relation with respect to the head node (e.g., *Subj*). For example, the node [*Noun*<*2,j,Subj*>] in rule 1.B.X specifies the type *Noun*, the identifier 2, the alignment index *j*, and the relation *Subject* with respect to the head node (i.e., the dominating *Verb*). Nodes with the same alignment index are viewed as translational equivalents that are aligned in the initial alignments.

Often a word on the LHS (English) corresponds to a categorial variant on the RHS (the FL). In such a case, the feature list includes categorial variation information. For example, the node [*Verb*<*1,i,CatVar:V_N*>] in the first rule specifies that a *V(erb)* (such as *pay*) on the LHS is aligned to a *N(oun)* (such as *payment*) on the RHS. These categorial variations are extracted from a large database called CatVar (Habash and Dorr, 2003). CatVar was developed using a combination of resources and algorithms including the Lexical Conceptual Structure (LCS) Verb and Preposition Databases (Dorr, 2001), the Brown Corpus section of the Penn Treebank (Marcus et al., 1993), an English morphological analysis lexicon developed for PC-Kimmo (Englex) (Antworth, 1990), NOMLEX (Macleod et al., 1998), Longman Dictionary of Contemporary English (LDOCE) (Procter, 1983), WordNet 1.6 (Fellbaum, 1998), and the Porter stemmer (Porter, 1980).

Note that, in addition to the simple indices, i , j , etc., DUSTer uses *conflated indices* ($C:i$, $C:j$, etc.) to refer to semantically light words that co-occur with high-content words but are generally unaligned in the initial alignments. Nodes marked $C:i$ are taken to be related structurally to a (single) high-content node marked i . The rationale for distinguishing between these two types of indices is that the semantically-light words (corresponding to nodes marked $C:i$) are generally unaligned in the initial word-alignment process, whereas the co-occurring high-content word (corresponding to the node marked i) usually has an initial alignment link. Nodes marked $C:i$ inherit their alignment links from the node marked i during alignment correction. For example, the conflated index $C:i$ on the RHS of rule 1.B.X associates two semantically light nodes—node 1 (LightVB) and node 4 (Oblique)—with node 3 (Obj), which has the same alignment index (i) as node 1 in LHS. In this case, the initial alignment associated with these two nodes (node 3 in RHS and node 1 in LHS) is copied to the two $C:i$ nodes in RHS. That is, the conflated indices provide the appropriate mechanism to copy an alignment link from a single (high-content) word pair (i.e., *fear, miedo(fear)*) to one or more light-content words in the FL (i.e., *fear, tener(have)* and *fear, de(of)*).

Figure 3.4 illustrates the relation between the LHS and the RHS of the rule 1.B.X. Solid lines between the two sides indicate the alignment links that are assumed to be available in the initial alignments (i.e., the nodes marked with simple indices i , j , etc.). The dashed lines indicate the alignment corrections that

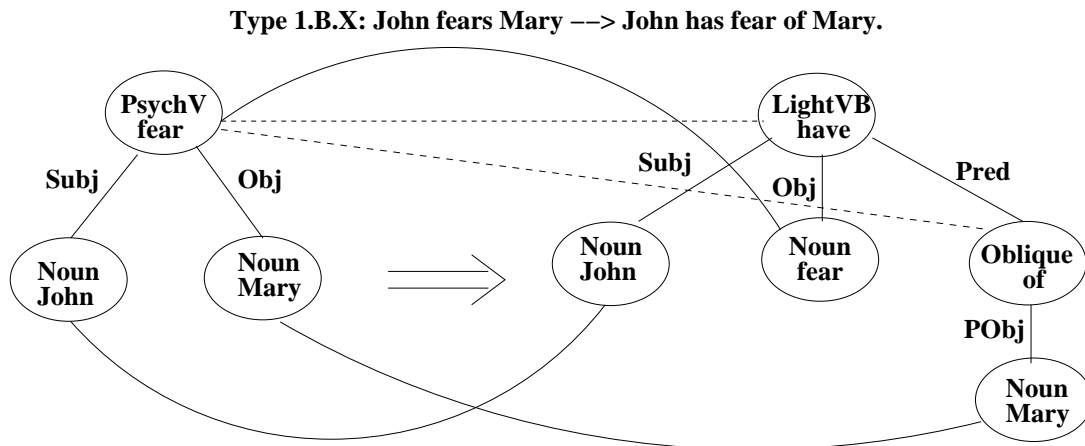


Figure 3.4: Universal Rule Application Example

must be added to produce the final alignment (i.e., the nodes marked with $C:i$).⁹ To test the applicability of a specific rule, the English dependency tree and surface relative order are matched against the LHS of the rule and the FL surface string is matched against the RHS side of the rule. If both conditions are satisfied, then the rule is applicable.

The current set of universal rules supports 4 foreign languages: Spanish, Hindi, Chinese, and Arabic. There are 21 rules for Hindi, 28 rules for Spanish, 44 rules for Arabic and 65 rules for Chinese.¹⁰ It is worth noting that Chapter 4 will demonstrate that it is possible to design a machine-learning framework where rules of this type are automatically induced.

⁹It is important to note that DUSTer does not require a FL dependency tree as input. The only inputs to the rules are the English dependency tree, the FL surface string, and the initial automatic alignments.

¹⁰Note that most of the rules are applicable to multiple languages, therefore the total number of the rules is not the sum of the numbers given above.

Rule Name	Rule Description	English	Spanish
0.A.X	Modal-Verb Deletion	may go	irá
0.A.X	Tense-Verb Deletion	will go	irá
0.C.X	to-Infinitive Deletion	to go	ir
1.A.X	Light-Verb Conflation	try	poner a prueba
1.B.X		desire	tener interés
1.BVar.X		know	darse cuenta
1.C.X		leap	dar un salto
1.D.X	Light-Verb	it is called the name	se llama
1.HVar1A.X	Contraction	be located in X	situar
1.HVar2.X		there are	hay
2.A.X	Manner	teaches	anda enseñando
2.A.XX	Conflation	regard	quedar considerando
2.B.X		is accomplished	se fue realizado
2.B.XX		is spent	se va gastando
3.A.X	Path	walk out	salió caminando
3.A.XX	Conflation	go over	ir atravesando
4.A.X	Head Swapping	usually go	suele ir
5.A.X	Thematic	I am pained	me duelen
5.A.XX	Divergence	I am pained	me duelen
5.B.X		I love it	me gusta/encanta
5.B.XX		he wants	le falta
5.BVar.X		he loves it	le gusta/encanta
5.BVar.XX		he wants	le falta
5.BVar.XXX		he loves it	le gusta/encanta
5.BVar.XXXX		he wants	le falta
6.A.X	Categorial	went on journey	fue viajando
6.A.XX	Divergence	mean to weep	querer llorando
6.B.X		I am jealous	tengo celos

Table 3.5: Set of Universal Rules in English-Spanish

This chapter focuses on application of DUSTER to English-Spanish data. Table 3.5 lists some universal rules in English-Spanish, along with some examples. Appendix B includes the entire list of the rules that are applicable to English and Spanish.

3.3.3 Setting Up DUSTer for a New Language

One of the most important design decisions for DUSTer is *ease of retargetability to a new language*. The goal is for DUSTer to be applicable to a new language with minimal effort and in a reasonably short amount of time (under one week). All that is needed is a native informant who has some linguistic background in the new language. The setup requirements for a new language are listed below, in two categories: (1) language-specific settings for DUSTer and (2) parser-specific settings for the input dependency tree. Each of these is addressed, in turn, below.

Language-Specific DUSTer Settings: There are three steps required for specifying language-specific settings:

1. **Preparing parameter files:** DUSTer requires that the settings for parameter classes be provided by a human in advance. As discussed in Section 3.3.1, the parameter classes contain certain words that are useful for identifying and handling divergences. In its current state, the words in each parameter class are explicitly provided by a human. Therefore, for each new language, these parameter classes are associated with the corresponding words in that language. As discussed in Section 3.3.1, this can be achieved by translating the words in English to the corresponding words in the new language for each parameter class. Some parameter classes in the new language may include words that are not translations of the words in English parameter classes; these must be added by the native informant to the list for the new language.

2. **Preparing a morphology file:** To be able to treat every morphological variant of a specific word in a sentence in the same way, some level of morphological analysis is required on for both languages. For languages that are rich in resources, such as English, the sentences can be analyzed morphologically using morphological analyzers and the results can be used instead of the original sentences. However, to keep the number of resources on the FL side as minimum as possible, DUSTER does not require a morphological analyzer on the FL side. Instead, it is sufficient for the native-informant to enumerate only the morphological variants of those words that are members of the parameter classes. For example, the Spanish verb *comenzar* (to begin), a member of the AspectV parameter class, is associated with its morphological variants in the morphology file (*comenzo*, *comenzas*, *comenzó*, *comencé*, etc.). This step is important because the morphological variants of the parameters that are missed in the morphology file will not be treated as members of the relevant parameter class(es).
3. **Updating Universal rules:** DUSTER requires that the set of universal rules be updated for the new language. Two important steps in this process are: 1) adding the word order to the applicable rules for the new language, and 2) adding new rules. For the first step, a native informant examines each rule, decides if the rule is applicable or not, and adds the correct word if the rule is found to be applicable. If there are other linguistic phenomena that are

not covered by the initial set of rules, the informant should add a new rule to handle these specific phenomena. Adding a rule is straightforward: the native informant specifies the order of words involved in the rule and then creates two subtrees for the LHS and RHS of the rule.

Parser-Specific DUSTer Settings: DUSTer is designed to be compatible with any dependency parser, provided the requirements below are met:

1. **Part-of-speech (POS) mapping files:** The POS tags vary from parser to parser. To make DUSTer broadly applicable to different tagging conventions, a general set of 10 different POS tags is used: *Adjective*, *Adverb*, *Complementizer*, *Conjunction*, *Determiner*, *Noun*, *Particle*, *Preposition*, *Punctuation*, and *Verb*. The POS labels used in the input dependency tree are mapped into these 10 general tags by means of a POS mapping file that must be specified by the DUSTer user, a task that requires under an hour of human inspection—even with an elaborate tag set, e.g., those of the Penn Treebank Project (Marcus et al., 1993).
2. **Relation mapping Files:** This is similar to the POS mapping above. The goal is to handle different dependency relations inside DUSTer. To make DUSTer broadly applicable to different relational conventions, a general set of 6 dependency relations is used: *Modifier (Mod)*, *Direct Object (Obj)*, *objects with prepositional phrases (PObj)*, *Predicative (Pred)*, *Sentence (S)*, and *Subject (Subj)*. The relational labels used in the input dependency tree are

mapped into these 6 general tags by means of a relational mapping file that must be specified by the DUSTer user.

The degree to which DUSTer is portable to a new language (Hindi) was tested during the Darpa TIDES-2003 Surprise Language Exercise (Dorr et al., 2003). The entire process took only a few days, demonstrating that the approach is promising for the rapid, large-scale acquisition of parallel data for previously unhandled languages.

Table 3.6 shows a breakdown of times for different tasks involved in the porting process—not only for Hindi, but also for two other languages, Arabic and Chinese. The process took less than 3 person-days total for each of these languages. In the case of Hindi, the required time was even less: only 1.5 days.

Task	Hindi	Arabic	Chinese
Parameter Setting	0.5 days	0.4 days	2.1 days
Morph Specification	1 day	2 days	0 days
Total Time	1.5 days	2.4 days	2.1 days

Table 3.6: Times Needed To Prepare DUSTer for 3 Languages

Although the parameter-setting task was *shorter* for Hindi and Arabic than for Chinese, the task of producing morphological variants for each word in the parameter classes took *longer* for these two languages because of their morphologic richness. Thus, there is a time tradeoff that balances out all three languages in the end: the overall time for incorporating a new language into DUSTer (i.e., parameter setting plus adding morphological variants) comes out to be about the

English Sentence: <i>She will fear her enemies .</i>					
Spanish Sentence: <i>Ella tendrá miedo de sus enemigos .</i>					
Enhanced English Dependency Tree					
Node	Word	POS	Parent	Rel	Features
1	She	Noun	3	Subj	[FunctionalN]
2	will	Verb	3	Mod	[TenseV CatVar:V_N, V_AJ]
3	fear	Verb	*root*	*	[PsychV CatVar:V_N, V_AJ, V_AV]
4	her	Noun	5	Mod	[FunctionalN]
5	enemies	Noun	3	Obj	[CatVar:N_AJ]
Initial Alignment: (She, Ella), (fear, miedo), (her,sus), (enemies, enemigos)					

Figure 3.5: Example Sentences, Dependency Tree and Initial Alignment

same for all three languages: 1.5–2.5 days.

3.3.4 Improving Alignments Using DUSTER

This section describes how DUSTER uses the universal rules and parameters above to infer corrected alignment links from a sentence pair, an English dependency tree, and an initial alignment, as in the example of Figure 3.5.

The dependency tree, produced initially by the Collins parser (Collins, 1997), is augmented during the preprocessing step (the first module in Figure 3.2) with semantic parameters and CatVar information. For example, the English word *that* belongs to 3 parameter classes: *Complement*, *Functional Determiner*, and *Functional Noun*. Thus, the tree node corresponding to *that* is augmented to include these parameter labels.¹¹ The CatVar feature is specified on the LHS (English)

¹¹Certain parameters may be filtered out if the associated POS is incompatible. For example, if the word *that* is tagged as a *Determiner*, then the parameters *Complement* and *Functional Noun* need not be added as possible parameter labels.

only, due to emphasis on English-heavy resources. This feature designates a potential POS for the corresponding (aligned) FL word. For example, the word *historic* (*Adjective*) may be categorically related to *history* (*Noun*) or *historically* (*Adverb*). These variants correspond to the following CatVar specifications: *CatVar_AJ_N* and *CatVar_AJ_AV*.

The enhanced dependency is next passed to the Universal-Rule Application component. The initial alignment is used as a strict filter on the application of the universal rules. Specifically, all LHS/RHS nodes related by a simple index (i.e., i, j) must have a corresponding alignment link in the initial alignments. Rules violating this requirement are strictly ruled out. The remaining (potentially-applicable) rules are checked for a match against the POS tags and parameter classes associated with the words in the two input sentences.

In addition to checking for a match, the rule-matching process identifies specific sentence positions of matching tokens. Given a modified (English/FL) sentence pair and a specific rule, the rule-application module returns the *match sets* corresponding to positions of words that match the RHS and LHS nodes of the rule. For example, the match set $(([2,3],[2]))$ for a particular rule indicates a match of the LHS against words 2 and 3 in English and a match of the RHS against word 2 in the FL.¹² In the example given in Figure 3.5 the match sets resulting from the application of rules 0.AVar.X and 1.B.X are $(([2,3],[2]))$ and $(([1,3,5],[1,2,3,4,6]))$,

¹²Because the rule may match the input in more than one place, the match sets are stored as lists of lists.

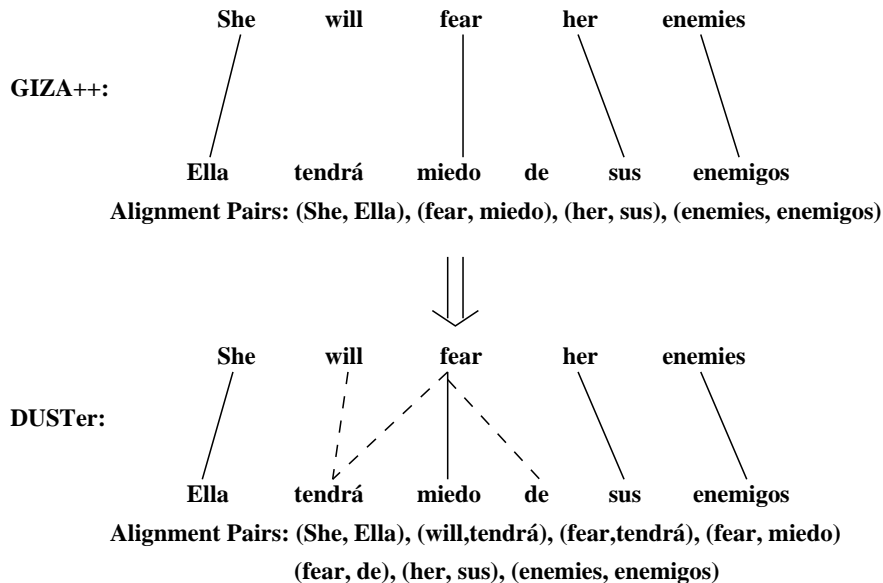


Figure 3.6: DUSTer’s Inferred Alignments from Initial GIZA++ Output

respectively.

The final step of DUSTer is alignment inference, a straightforward extraction of corrected alignment links using the match sets and the initial alignment. Formally, for each match-set element $([\dots, i, \dots] [\dots, j, \dots])$, where i and j carry the same coindex k (or the same conflated coindex), the link (i, j) is added to the partial alignments. As a final step, the partial alignment provided by DUSTer is filled out with the initial alignment links (for links left unmodified by DUSTer) to produce a complete alignment. Using GIZA++ (Och, 2000) as the initial aligner, Figure 3.6 shows the inferred alignment for the sentence pair in the current example.

3.4 Evaluation

This section presents the current status of DUSter and a comparison of DUSter output to that of a state-of-the-art alignment system—GIZA++ (Och, 2000).

3.4.1 Settings

The evaluation of DUSter involves a set of 199 English-Spanish sentence pairs (nearly 5K words on each side) from a mixed corpus (UN + Bible + FBIS). Dependency trees are generated using Collins (Collins, 1997).

As input alignments, two sets of alignments were generated by GIZA++ in two directions interchanging the source and target language, i.e., $\text{GIZA++}(e \rightarrow s)$ and $\text{GIZA++}(s \rightarrow e)$. GIZA++ was trained (prior to this experiment) on 48K sentence pairs from a mixed corpus (UN + Bible + FBIS), with nearly 1.2M of words on each side. This pre-experiment GIZA++ training involved 10 iterations of Model 1, 5 iterations of HMM and 5 iterations of Model 4. The average sentence length for the GIZA++ training data was 24 words for English and 26 words for Spanish.

Three different combined alignments were generated by the following combination techniques as possible initial aligners:

1. Intersection of both directions (which will be represented as $\text{GIZA++}(\text{int})$),
2. The union of both directions (which will be represented as $\text{GIZA++}(\text{union})$),

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow s$)	87.0	67.0	24.3
DUSTer[GIZA++($e \rightarrow s$)]	84.6	70.6	23.0
GIZA++($s \rightarrow e$)	88.0	67.5	23.6
DUSTer[GIZA++($s \rightarrow e$)]	84.2	72.1	22.3
GIZA++(int)	98.2	59.6	25.9
DUSTer[GIZA++(int)]	92.2	64.0	24.4
GIZA++(union)	80.6	74.9	22.3
DUSTer[GIZA++(union)]	79.5	77.8	21.4
GIZA++(gdf)	83.8	74.4	21.2
DUSTer[GIZA++(gdf)]	82.5	77.4	20.1

Table 3.7: GIZA++ and DUSTer Results (on English-Spanish)

3. A heuristic combination technique called *diag-final* (Koehn et al., 2003) (which will be represented as GIZA++(gdf)).

For evaluating word alignments, the precision, recall and error rate were computed on the entire set of sentence pairs (see Section 2.6 for a description of the evaluation metrics). A manually aligned corpus was used as the gold standard. The manual annotation was done by a bilingual English-Spanish speaker. Every link in the gold standard is considered a sure alignment link (i.e., $P = S$) during evaluation.

3.4.2 Results

Table 3.7 summarizes the evaluation results for five different initial alignments. The differences between the alignment error rates for DUSTer and GIZA++ are statistically significant at a 95% confidence level using a two-tailed t-test.

For all 5 initial alignments, DUSTer yields a relative reduction of 4-6% in

alignment error rate. In the best case, where GIZA++(gdf) is used as the initial alignment, DUSter yields an alignment error rate of 20.1%—a relative reduction of 5.2% over GIZA++(gdf). In each experiment, the precision goes down while the recall is improved. The main reason for this behavior is that DUSter adds alignment links based on neighboring words. Another important observation is that the improvement by DUSter depends on the behavior of the initial aligners. For instance, when the initial alignment has a high precision but low recall, as in the case of intersection of alignments, DUSter yields a higher improvement on recall and AER because there are more links that can be added to the initial alignment. On the other hand, when the initial alignment has low precision but high recall, the improvement by DUSter is relatively lower.

Whereas this result appears to be a modest improvement, the strictest possible application of the DUSter rules has been used to obtain these results. In particular, all LHS/RHS nodes carrying a simple index **must** have a corresponding alignment link in the initial alignments—and all unindexed nodes must **not** have a corresponding alignment link in the initial alignment. Relaxation of this constraint might increase the improvement by DUSter.

3.5 Summary

This chapter introduced a novel approach—DUSter—to improve word alignments by proper handling of translation divergences. DUSter identifies the places where

two sentences are divergent using a set of manually crafted rules, and proposes new alignment links based on already existing links. The evaluation of this approach on English-Spanish data has demonstrated alignment improvements of up to 6% in terms of alignment error rate.

In DUSter, the alignment errors stemming from the initial alignment biases and the lack of training data can be corrected by identifying those errors and creating rules that correct them. As discussed in Section 3.2, function words are part of translation divergences most of the time. As a result, DUSter improves the alignment of function words by employing a set of rules to properly handle translation divergences. Finally, DUSter allows easier incorporation of linguistic knowledge into word alignment since the rules can be constructed to include any linguistic resources. As a result, existing systems do not need to be modified to take advantage of additional linguistic resources.

DUSter’s success depends on the coverage and accuracy of the set of universal rules. For handling all translation divergences, it is necessary for a bilingual speaker to identify all possible types of divergences for a given language pair. Since the rules are conditioned on the existence and absence of certain alignment links, DUSter also depends on the behavior of the initial alignment; thus, the rules need to be tailored according to a given alignment. If one identifies all the places where an initial aligner makes errors and create rules handling those cases, DUSter will yield better word alignments. This is the motivation for the transition to the alignment improvement approach in Chapter 4, where such rules are automatically induced.

Chapter 4

Alignment Link Projection (ALP)

Word alignment is an exponential problem and all word alignment algorithms, just like all other machine learning techniques, make a set of assumptions (or *biases*) to reduce the hypothesis space. Because of these biases, learning algorithms tend to make similar errors throughout the entire data. As discussed in Chapter 3, finding common errors made by a specific algorithm and correcting them improves the overall result.

The crucial question is how to extract the patterns of errors made by a particular word alignment system. As discussed in Chapter 3, providing a set of manually constructed rules to catch those errors yields promising results but the improvement over the baseline is relatively low. The coverage of the rules described in the previous chapter was not adequate to capture common errors made by the initial alignment system for the following reasons:

1. The rules are too general in that they are not tailored to accommodate the specific characteristics of individual alignment systems; thus, common errors

made by the initial alignment system are not necessarily addressed by the rules.

2. The rules are too specific in that they are designed to handle a small set of divergence examples that are not necessarily represented by the cases that arise in most data sets.

These two deficiencies can be addressed by using automatically induced rules instead of manually constructed rules. An error-driven learning approach allows users to identify the frequently occurring errors that are made by a particular alignment system; thus, it is more comprehensive. Moreover, an automated rule generation technique eliminates the need for a bilingual speaker to identify divergence types for each language pair; thus, it is less labor-intensive.

This chapter presents a new approach, *Alignment Link Projection (ALP)*, that post-processes a given alignment using linguistically-oriented rules, learns common alignment errors made by the system, and attempts to correct them. The idea is similar to that of DUSTER, where manually-crafted rules are used to correct alignment links related to language divergences. This approach differs, however, in that the rules are extracted automatically—not manually—by examining an initial alignment and categorizing the errors according to features of the words. As in DUSTER, ALP assumes the initial alignment system adequately captures certain kinds of word correspondences but fails to handle others. ALP starts with an initial alignment and then fills out (i.e., *projects*) new word-level alignment relations (i.e.,

links) from existing alignment relations. ALP then deletes certain alignment links associated with common errors, thus improving precision and recall.

ALP adapts transformation-based error-driven learning (TBL) (Brill, 1993) to the problem of word alignment. Following the TBL formalism, ALP attempts to find an ordered list of transformation rules (within a pre-specified search space) to improve a baseline annotation. The transformation rules decompose the search space into a set of consecutive words (windows) within which alignment links are added to, or deleted from, the initial alignment. This window-based approach exploits the clustering tendency of alignment links, i.e., when there is a link between two words, there is frequently another link in close proximity.

TBL is an appropriate choice for this problem for the following reasons:

1. It can be optimized directly with respect to an evaluation metric.
2. It learns rules that improve the initial prediction iteratively, so that it is capable of correcting previous errors in subsequent iterations.
3. It provides a readable description (or classification) of errors made by the initial system, thereby enabling alignment refinements.

The rest of this chapter presents a brief overview of TBL and then describes the adaptation of TBL to the word alignment problem.

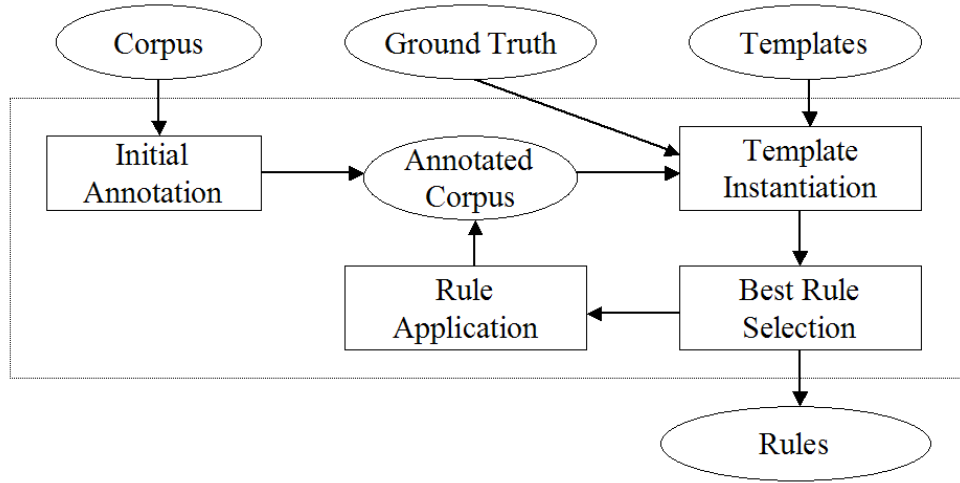


Figure 4.1: TBL Architecture

4.1 Transformation-based Error-driven Learning

As shown in Figure 4.1, the input to TBL is an unannotated parallel corpus, a ground truth, and a set of rule templates. In the first step of TBL, an unannotated corpus is passed to an initial annotator, resulting in an annotated corpus. This annotator may be quite simple (e.g., random annotation) or more sophisticated. On each iteration, the output of the previous iteration is compared against the ground truth, and an ordered list of transformation rules is learned that make the previous annotated data better resemble the ground truth.

A set of *rule templates* determines the space of allowable transformation rules. A rule template has two components: a triggering environment (condition of the rule) and a rewrite rule (action taken). On each iteration, these templates are instantiated with features of the template constituents when the condition of the rule is satisfied. For example, in the context of part-of-speech tagging, a template

might look like:

Condition : The preceding word is a <POS_tag>

Rewrite rule : Change the tag from <POS_tag_1> to <POS_tag_2>

A possible instantiation of this template is as follows:

Condition : The preceding word is a Determiner

Rewrite rule : Change the tag from Modal-Verb to Noun

This particular instantiation enables the correction of the POS tag for the word *can* in the following sentence:

Incorrect: The/*determiner* can/*modal* rusted/*verb* ./*punctuation*

Correct : The/*determiner* can/*noun* rusted/*verb* ./*punctuation*

This process eventually identifies all possible instantiated forms of the templates. Among all these possible rules, the transformation whose application results in the best score—according to some objective function—is identified. This transformation is added to the ordered list of transformation rules. At the end of each iteration, the selected transformation rule is applied to the annotated corpus, which is then passed as training data to the next iteration. This entire learning process is repeated on the transformed corpus: instantiating rule templates, choosing the best rule, and applying it to the current state. The learning stops when there is no transformation that improves the current state of the data or a pre-specified threshold is reached.

When presented with new data, the transformation rules are applied in the order that they were added to the list of transformations. The output of the system is the annotated data after all transformations are applied to the initial annotation.

To apply TBL to a new problem, one should specify:

1. An initial state annotator,
2. A set of templates,
3. How to instantiate the templates in each iteration, and
4. How to choose the best transformation rule.

TBL approach is very similar to the idea behind decision trees (Quinlan, 1986). Decision tree learning, like TBL, is a supervised learning method that outputs a set of questions that can be asked about an entity to determine its proper classification. Decision trees are built by finding the question which partitions the search space in the best way according to a given measure, such as maximum entropy, splits the training data according to that question, and then recursively reapplies this procedure on each resulting subset.

One reason for choosing TBL over decision tree learning is that it has been proven that any decision tree can be converted into a transformation list (an ordered list of transformation rules), i.e., Decision Trees \subseteq Transformation Lists, and there exist transformation lists for which no equivalent decision trees exist, i.e., Decision Trees \neq Transformation Lists (Brill, 1995). Moreover, decision trees

are subject to the data sparsity problem as the depth of the decision tree increases, whereas TBL can utilize the whole training data in each iteration. This is particularly important in situations where annotated training data is very limited, as in the case of word alignments.

TBL has been applied to various NLP tasks, e.g., part of speech tagging (Brill, 1995), prepositional phrase attachment disambiguation (Brill and Resnik, 1994), parsing (Brill, 1996), dialogue act tagging (Samuel et al., 1998), and phrase chunking (Florian et al., 2000). It has been shown to be quite effective in many applications. For example, TBL has been shown to achieve more than 97% accuracy in part-of-speech tagging.

4.2 Notation

The following is the definitions and notation that are used for providing a formal presentation of the ALP system:

- $\mathbf{e} = e_1, \dots, e_i, \dots, e_I$ is a sentence in language L_1 and $\mathbf{f} = f_1, \dots, f_j, \dots, f_J$ is a sentence in language L_2 .
- An *alignment link* (i, j) corresponds to a translational equivalence between e_i and f_j .
- A *neighborhood* of an alignment link (i, j) —denoted by $N(i, j)$ —consists of 8 possible alignment links in a 3×3 window with (i, j) in the center of the window. Each element of $N(i, j)$ is called a *neighboring link* of (i, j) .

1. Annotate the training data using an initial aligner
2. For each rule template:
 - 2.1. Instantiate the template with all possible values for the constituents
3. For each rule instantiation:
 - 3.1. Apply the rule to a copy of the the training set
 - 3.2. Score the result using the objective function
4. Select the rule with the best score, and
5. If the score for the best rule is higher than the previous score
 - 5.1. Update the training data using this rule
 - 5.2. If the number of rule applications is below a threshold, go to step 2

Figure 4.2: Pseudo-code for Alignment Link Projection

- $neighbor_exists_A(i, j)$ denotes whether there is an alignment link in the neighborhood of the link (i, j) in a given alignment A . Formally,

$$neighbor_exists_A(i, j) = \begin{cases} true & \text{if } \exists x, x \in N(i, j) \text{ \& } x \in A \\ false & \text{otherwise} \end{cases}$$

- $nullE_A(i)$ is *true* if and only if e_i is not aligned to any word in \mathbf{f} in a given alignment A . Similarly, $nullF_A(j)$ is *true* if and only if f_j is not aligned to any word in \mathbf{e} in a given alignment A . Formally,

$$nullE_A(i) = \begin{cases} false & \text{if } \exists j, (i, j) \in A \\ true & \text{otherwise} \end{cases}$$

$$nullF_A(j) = \begin{cases} false & \text{if } \exists i, (i, j) \in A \\ true & \text{otherwise} \end{cases}$$

4.3 Description and Parameters of ALP

ALP is a TBL implementation that projects alignment links from an initial input alignment. Figure 4.2 presents the algorithm for alignment link projection in the TBL framework.

ALP starts with a parallel corpus (which might be enriched with linguistic features such as POS tags), a set of rule templates, and a ground-truth alignment for a given corpus. The first step (Step 1 in Figure 4.2) is word-level alignment of the corpus using an initial annotator, which is usually an existing word alignment system, to obtain a word-aligned corpus. Next, for each rule template provided to the system, ALP goes over the annotated corpus and finds the words that satisfy the condition of the template (Step 2 in Figure 4.2). If the template is applicable to the words in question, a new rule is generated by instantiating the constituents of the template with features of the words, and by recording whether the action taken by the rule results in a correct or incorrect alignment link.

After all the templates are instantiated with all possible values, ALP determines which rule results in the best score by applying the rule to a copy of the current state of the annotated corpus, and evaluating against a ground truth (Steps 3–4 in Figure 4.2). In the next step, ALP checks whether the application of the best rule results in a higher score than the previous alignment. If this is the case, ALP updates the current state of the annotated corpus by applying the rule, which results in a different alignment from the previous iteration (Step 5 in Figure 4.2).

For computational reasons, a maximum number of rule applications is allowed. If that threshold is reached, ALP stops and outputs the final alignment. Otherwise, ALP returns to instantiation of the templates (i.e., to Step 2 in Figure 4.2) using the corpus with the updated alignment. ALP stops when the best rule found in the current iteration yields a lower score than the previous alignment, or when the maximum number of rule applications is satisfied.

Given a parallel corpus and a ground-truth word-alignment, ALP needs four parameters to be specified in order to learn transformation rules for alignment improvement:

1. Initial alignment,
2. Set of templates,
3. Template instantiation method, and
4. Best rule selection method.

The rest of this section describes several variations of ALP by setting these four parameters in different ways.

4.3.1 Initial Alignment

Any existing word-alignment system may be used as the initial annotation step of the TBL algorithm. For the experiments presented in this chapter, the initial aligner is GIZA++ (Och, 2000), which is a state-of-the-art word alignment system.

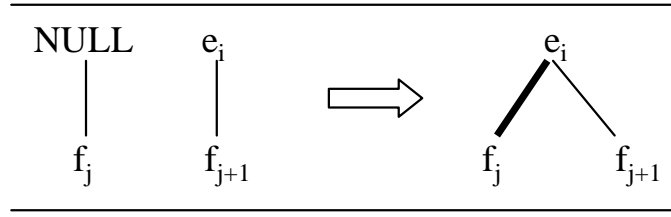


Figure 4.3: Graphical Representation of a Template

Various initial alignments are generated using GIZA++ in different directions, and by combining those uni-directional alignments using simple heuristics, including: (1) intersection, (2) union and (3) the *grow-diag-final* (gdf) method in (Koehn et al., 2003).¹ Note that these initial aligners are treated as black boxes in the ALP framework.

4.3.2 TBL Templates

Alignment improvement templates consider consecutive words (of size 1, 2 or 3) in both languages. The condition portion of a TBL rule template tests for the existence of an alignment link between two words. The action portion involves the addition or deletion of an alignment link. For example, the rule template,

Condition: $(NULL, j), (i, j + 1)$

Rewrite rule: add (i, j)

which is illustrated in Figure 4.3, is applicable only when a word (e_i) in one language is aligned to the second word (f_{j+1}) of a phrase (f_j, f_{j+1}) in the other language, and the first word of the phrase (f_j) is unaligned in the initial alignment.

¹The gdf method is the best known alignment combination technique.

The action taken by this rule template is to add a link between e_i and f_j .² For the sake of complexity reduction, if the action taken by the rule involves the addition of a link, it is implicitly assumed that the condition of the rule requires the absence of that link. Similarly, if the action taken by the rule involves the deletion of a link, it is implicitly assumed that the condition of the rule requires the existence of that link.

ALP employs three different sets of templates to project new alignment links or delete existing links in a given alignment:

1. Expansion of the initial alignment according to another alignment
2. Deletion of spurious alignment links
3. Correction of multi-word (one-to-many or many-to-one) correspondences

Each of these is described below.

4.3.2.1 Expansion Templates

Expansion templates are used to extend an initial alignment given another alignment as the validation set. This approach is similar to grow-diag-final method in that it adds links based on knowledge about neighboring links, but it differs in that it *also* uses features of the words themselves to decide which neighboring links to add.

²A thick line indicates an added link.

Condition	Action
$(i, j) \in A, (i - 1, j - 1) \in V$	add $(i - 1, j - 1)$
$(i, j) \in A, (i - 1, j) \in V$	add $(i - 1, j)$
$(i, j) \in A, (i - 1, j + 1) \in V$	add $(i - 1, j + 1)$
$(i, j) \in A, (i, j - 1) \in V$	add $(i, j - 1)$
$(i, j) \in A, (i, j + 1) \in V$	add $(i, j + 1)$
$(i, j) \in A, (i + 1, j - 1) \in V$	add $(i + 1, j - 1)$
$(i, j) \in A, (i + 1, j) \in V$	add $(i + 1, j)$
$(i, j) \in A, (i + 1, j + 1) \in V$	add $(i + 1, j + 1)$
$(i - 1, j - 1) \in A, (i + 1, j + 1) \in A, (i, j) \in V$	add (i, j)
$(i + 1, j - 1) \in A, (i - 1, j + 1) \in A, (i, j) \in V$	add (i, j)

Table 4.1: Templates for Expanding the Alignment A According to a Validation Alignment V

The ALP expansion templates are presented in Table 4.1. The first 8 templates add a new link to the initial alignment A if there is a neighboring link in the validation alignment V . The final two templates enforce the presence of at least two neighboring links in the validation set V before adding a new link.

4.3.2.2 Deletion Templates

Existing alignment algorithms (e.g., GIZA++) are biased toward aligning some words, especially infrequent ones, in one language to many words in the other language in order to minimize the number of unaligned words, even if many incorrect alignment links are induced.³ Deletion templates are useful for eliminating the resulting spurious links.

³This is a well-known characteristic of statistical alignment systems—motivated by the need to ensure a target-word translation e_i for each source word f_j while modeling $p(\mathbf{f}|\mathbf{e})$ —for downstream MT (Brown et al., 1993; Och and Ney, 2003; Moore, 2004).

Condition	Action
$(i, j) \in A, (i, k) \in A,$ $neighbor_exists_A(i, j),$ $not(neighbor_exists_A(i, k))$	del (i, k)
$(i, j) \in A, (k, j) \in A,$ $neighbor_exists_A(i, j),$ $not(neighbor_exists_A(k, j))$	del (k, j)

Table 4.2: Templates for Deleting Spurious Links in a Given Alignment A

The basic idea is to remove alignment links that do not have a neighboring link if the word in question has already been aligned to another word. Table 4.2 lists two simple templates to clean up spurious links. The first template deletes spurious links for a particular word e_i in \mathbf{e} and the second template deletes spurious links for a particular word f_j in \mathbf{f} . The motivation is that if a word is involved in multiple alignment links, the links that do not have any other link in its neighborhood can be safely deleted. The templates in Table 4.2 are graphically illustrated in Figure 4.4. The first template deletes the spurious alignment link between e_2 and f_6 since e_2 is aligned to two FL words and there is no link in the neighborhood of the link $(2, 6)$. Similarly, the second template deletes the alignment link between e_6 and f_3 because f_3 is aligned to two English words and there is no link in the neighborhood of the link $(6, 3)$.

The deletion templates are particularly useful when they are applied before the application of templates that add alignment links. This is because the templates that add links rely heavily on the correctness of existing links: if the initial alignment has many incorrect alignment links, the templates that add links will generate other incorrect links based on the initial incorrect links. The deletion

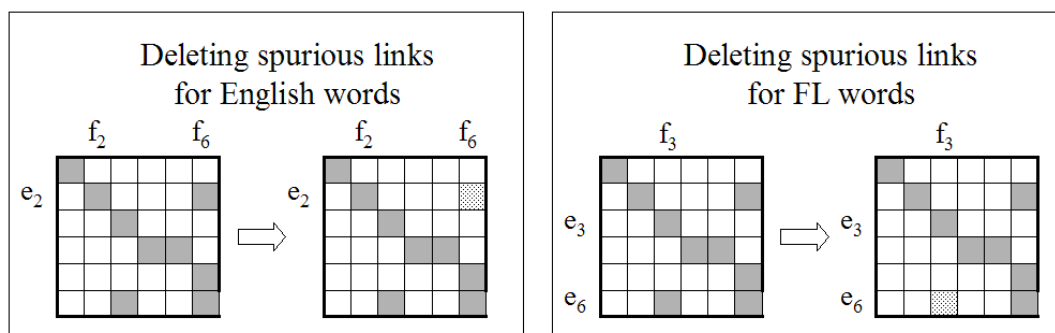


Figure 4.4: Illustration of Deletion Templates

templates are designed to eliminate incorrect links at the outset, so that the link addition step does not proliferate the number of incorrect links.

4.3.2.3 Multi-Word Correction Templates

Current alignment algorithms produce one-to-one word correspondences quite successfully. However, accurate alignment of phrasal constructions (many-to-many correspondences) is still problematic. On the one hand, the ability to provide *fully* correct phrasal alignments is impaired by the occurrence of high-frequency function words and/or words that are not exact translations of the words in the other language. On the other hand, most alignment systems are capable of providing *partially* correct phrasal alignments that may be exploited for downstream alignment enrichment.

To establish the types of multi-word templates used for alignment correction in ALP, a preliminary study was conducted using 40 manually-aligned English-Spanish sentences from a mixed corpus (UN + Bible + FBIS) as a gold standard.

# Gold-Standard Correspondence	English Words	Spanish Words
	GIZA++/ Gold-Standard	GIZA++/ Gold-Standard
1	746 / 818	840 / 1133
2	102 / 123	57 / 77
3	42 / 49	9 / 16
> 3	14 / 19	8 / 14
> 1	158 / 191	74 / 107

Table 4.3: Number of Words with at Least One Correct Alignment Link (on English-Spanish)

This study revealed that, out of the 191 English words that were aligned to more than one Spanish word in the gold standard, 158 were correctly aligned by an existing alignment system (GIZA++) to at least one of the words in the Spanish phrasal construction. Similarly, out of the 107 Spanish words that were aligned to more than one English word in the gold standard, 74 were correctly aligned by GIZA++ to at least one of the words in the English phrasal construction. Table 4.3 presents the results of this study using GIZA++ with English as the source language.⁴

The ALP templates for handling multi-word correspondences are grounded in the outcome of this finding. That is, the templates are based on the (frequently correct) assumption that at least one alignment link in a many-to-many correspondence is correctly identified in the initial alignment. Table 4.4 lists the templates for correcting alignment links in multi-word correspondences. The first five templates handle $(e_i \rightarrow f_j f_{j+1})$ correspondences, the next five handle $(e_i e_{i+1} \rightarrow f_j)$

⁴The same analysis was done for the other direction and similar results were obtained.

Condition	Action
$nullF_A(j), (i, j+1) \in A$	add (i, j)
$nullF_A(j+1), (i, j) \in A$	add $(i, j+1)$
$(i, j) \in A, (i, j+1) \in A$	del (i, j)
$(i, j) \in A, (i, j+1) \in A$	del $(i, j+1)$
$nullF_A(j), nullF_A(j+1)$	add (i, j) , add $(i, j+1)$
$nullE_A(i), (i+1, j) \in A$	add (i, j)
$nullE_A(i+1), (i, j) \in A$	add $(i+1, j)$
$(i, j) \in A, (i+1, j) \in A$	del (i, j)
$(i, j) \in A, (i+1, j) \in A$	del $(i+1, j)$
$nullE_A(i), nullE_A(i+1)$	add (i, j) , add $(i+1, j)$
$(i+1, j+1) \in A, nullE_A(i), nullF_A(j)$	add (i, j)
$(i, j) \in A, nullE_A(i+1), nullF_A(j+1)$	add $(i+1, j+1)$
$(i, j) \in A, (i+1, j) \in A, (i+1, j+1) \in A$	add $(i, j+1)$
$(i, j) \in A, (i, j+1) \in A, (i+1, j+1) \in A$	add $(i+1, j)$
$nullE_A(i), (i-1, j) \in A, (i+1, j) \in A$	add (i, j)
$nullF_A(j), (i, j-1) \in A, (i, j+1) \in A$	add (i, j)

Table 4.4: Templates for Handling Multi-Word Correspondences in a Given Alignment A

correspondences, the next four handle $(e_i e_{i+1} \rightarrow f_j f_{j+1})$ correspondences, and the final two handle $(e_{i-1} e_i e_{i+1} \rightarrow f_j)$ and $(e_i \rightarrow f_{j-1} f_j f_{j+1})$ correspondences.

Two examples of multi-word correction templates are presented in Figure 4.5, where the first template corresponds to 6th template in Table 4.4 and the second corresponds to 14th template in Table 4.4. The figure also provides English-Spanish words that satisfy the conditions of the templates. The first template adds the missing alignment link between *I* and *quiero*, and the second template adds the missing alignment link between *water* and *el*.

The alignment rules given above may introduce errors that require additional cleanup. Thus, two simple templates are introduced (shown in Table 4.5) to accommodate the deletion or addition of links between a single pair of words. These

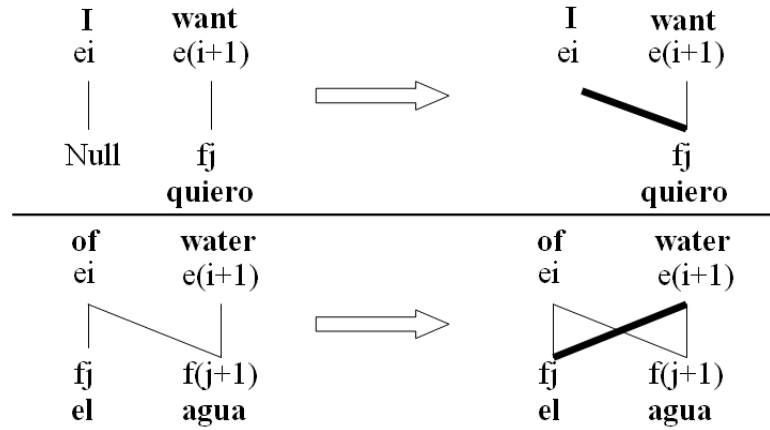


Figure 4.5: Graphical Illustration of Two Multi-word Correction Templates

Condition	Action
$(i, j) \in A$	del (i, j)
$nullE_A(i), nullF_A(j)$	add (i, j)

Table 4.5: Templates for Correcting One-to-One Correspondences in a Given Alignment A

templates are also useful to catch systematic errors made between one English and one foreign language word. For instance, if it is found that a particular word alignment system aligns an English determiner and a Spanish verb incorrectly most of the time, the first template in Table 4.5 will catch those errors and correct them.

4.3.3 Instantiation of Templates

ALP starts with a set of templates and an initial alignment and attempts to instantiate the templates during the learning process. The templates can be instantiated using two methods: Simple (a word is instantiated with a specific feature) or Generalized (a word is instantiated using a special keyword **anything**).

ALP requires only a small amount of manually aligned data for this process—

a major strength of the system. However, if the templates were instantiated with the actual words of the manual alignment, the frequency counts (from such a small data set) would not be high enough to derive reasonable generalizations. Thus, ALP adds new links based on linguistic features of words, rather than the actual words. Using these features is what sets ALP apart from systems like the grow-diag-final approach. Specifically, three features are used to instantiate the templates:

1. **POS tags on both sides:** ALP uses POS tags assigned by the MXPOST tagger (Ratnaparkhi, 1996) for English and Chinese, and Connexor for Spanish. POS tags have been shown to be effective in improving different alignment models (Toutanova et al., 2002; Liu et al., 2005).
2. **Dependency relations:** ALP utilizes dependencies for a better generalization, if a dependency parser is available in either language. The experiments reported here use only an English dependency parser (a version of the Collins parser (Collins, 1997) that has been adapted for building dependencies). No such resource is used for the other language. Dependency relations have been shown to be useful in improving alignment systems (Cherry and Lin, 2003).
3. **A set of word classes:** ALP uses 16 different word classes, 9 of which are different semantic verb classes while the other 7 are function words, prepositions, and complementizers. These are based on the parameter classes of (Dorr et al., 2002), which were described in detail in Section 3.3.1. These pa-

parameter classes have been shown to be useful in improving alignment systems (Dorr et al., 2002; Ayan et al., 2004).

To address the issue of data sparseness, the set of original POS tags and dependency relations were converted to a more generalized set of 10 POS tags and 6 relations, as described in Section 3.3.3. The final set of POS tags consists of *Adjective*, *Adverb*, *Complementizer*, *Conjunction*, *Determiner*, *Noun*, *Particle*, *Preposition*, *Punctuation* and *Verb*. The final set of relations consists of *Modifier (Mod)*, *Direct Object (Obj)*, *objects with prepositional phrases (PObj)*, *Predicative (Pred)*, *Sentence (S)* and *Subject (Subj)*.

If both POS tags and dependency relations are available, they can be used together to instantiate the templates. That is, a word can be instantiated in a TBL template with: (1) a POS tag (e.g., Noun, Adjective); (2) a relation (e.g., Subject, Direct Object); (3) a parameter class (e.g., Change of State Verb); or (4) different subsets of (1)–(3). In addition, a more generalized form of instantiation is provided, where words in the templates may match the keyword **anything**.

Possible instantiations of the templates in Figure 4.5 on English-Spanish data are as follows:

1. $e_i = \textit{FunctionalNoun}$, $e_{i+1} = \textit{Verb}$, $f_j = \textit{Verb}$.
2. $e_i = \textit{Predicate}$, $e_{i+1} = \textit{Noun}$, $f_j = \textit{FunctionalDet}$, $f_{j+1} = \textit{Noun}$.

4.3.4 Best Rule Selection

The rules are scored using two different metrics: (1) accuracy of the rule unto itself; and (2) the overall impact of the application of the rule on the entire data. Both metrics may be used for selecting the best rule after generating all possible instantiations of templates. More specifically:

1. **Rule Accuracy:** The goal is to minimize the errors introduced by the application of a transformation rule. This is similar to the best rule metric that has been proposed in the original TBL algorithm. The accuracy of a rule r is computed as $good(r) - 2 \times bad(r)$, where $good(r)$ is the number of alignment links that are corrected by the rule, and $bad(r)$ is the number of incorrect alignment links that are generated by the rule.
2. **Overall impact on the training data:** The accuracy mechanism (above) is useful for biasing the system toward higher precision. However, if the overall system is evaluated using a metric other than precision (e.g., recall), the accuracy mechanism may not guarantee that the best rule is chosen at each step. Thus, the best rule may be chosen according to the evaluation metric to be used for the overall system.

4.4 Evaluation Data and Settings

ALP was evaluated using 5-fold cross validation on two different data sets:

1. A set of 199 English-Spanish sentence pairs (nearly 5K words on each side)

from a mixed corpus (UN + Bible + FBIS).

2. A set of 491 English-Chinese sentence pairs (nearly 13K words on each side) from 2002 NIST MT evaluation test set.

The details about the English-Spanish data set can be found in Section 3.4.1. For English-Chinese, the 2002 NIST MT evaluation test set was used, and each sentence pair was aligned by two native Chinese speakers who are fluent in English. Each alignment link appearing in both annotations was considered a *sure link*, and links appearing in only one set were judged as *probable*. The annotators were not aware of the specifics of the alignment-correction approach.

In experiments with ALP, the initial alignments were generated by GIZA++ (Och, 2000). The details of how English-Spanish alignments were generated can be found in Section 3.4.1. For the English-Chinese experiments, an additional 107K sentence pairs from the FBIS corpus (nearly 4.1M English and 3.3M Chinese words) were used to train GIZA++ using 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4. The average sentence length for the training data was 38 words for English and 30 words for Chinese.

As in the evaluation of DUSter (see Section 3.4), 5 variations of GIZA++ alignments were used as the initial alignments: Uni-directional GIZA++ alignments ($\text{GIZA++}(e \rightarrow f)$ and $\text{GIZA++}(f \rightarrow e)$), where e corresponds to English and f corresponds to the foreign language (c for Chinese and s for Spanish), and three combined alignments ($\text{GIZA++}(\text{int})$, $\text{GIZA++}(\text{union})$ and $\text{GIZA++}(\text{gdf})$).

Alignments	English-Spanish			English-Chinese		
	Pr	Rc	AER	Pr	Rc	AER
GIZA++($e \rightarrow f$)	87.0	67.0	24.3	70.4	68.3	30.7
GIZA++($f \rightarrow e$)	88.0	67.5	23.6	66.0	69.8	32.2
GIZA++(int)	98.2	59.6	25.9	94.8	53.6	31.2
GIZA++(union)	80.6	74.9	22.3	58.3	84.5	31.6
GIZA++(gdf)	83.8	74.4	21.2	61.9	82.6	29.7

Table 4.6: GIZA++ Results (on English-Spanish and English-Chinese)

Table 4.6 summarizes the precision, recall and alignment error rate values (in percentages) for five possible initial alignments on the whole set of 199 sentence pairs in English-Spanish, and on the whole set of 491 sentence pairs in English-Chinese. The best results for each metric are highlighted in boldface. Using intersection of both directions gives the best precision. The best recall, on the other hand, is obtained by the union of both directions. The lowest alignment error rate is obtained by the heuristic method grow-diag-final.

For k -fold cross validation experiments, the pairs of sentences were divided randomly into 5 groups. For each fold, 4 groups were taken as the ground truth for learning transformation rules and the other group was used as a gold standard for evaluating word alignments. This process was repeated 5 times so that each sentence pair was tested exactly once. Then the precision, recall and error rate were computed on the entire set of sentence pairs for each data set (see Section 2.6 for a description of the evaluation metrics).⁵

⁵The number of alignment links varies over each fold. Therefore, evaluation is performed on all data at once instead of evaluating on each fold and then averaging. This is similar to micro averaging vs. macro averaging in information retrieval literature (Yang, 1999). The motivation

4.5 Experimental Results

This section describes the evaluation of ALP variants using different combinations of settings of the four parameters described above. The two language pairs in this experiment were English-Spanish and English-Chinese.

ALP was compared to each of the five alignments above using different settings of 4 parameters: $ALP[IA, T, I, BRS]$, where IA is the initial alignment, T is the set of templates, I is the instantiation method, and BRS is the metric for the best rule selection at each iteration. In the rest of this section, the following abbreviations are used:

1. **Initial Aligners:** Five different initial alignments were used, as described in Section 4.4:

- $GIZA++(e \rightarrow f)$,
- $GIZA++(f \rightarrow e)$,
- $GIZA++(int)$,
- $GIZA++(union)$, and
- $GIZA++(gdf)$.

behind the choice of evaluating on all data at once is that word alignments are always evaluated at the document level rather than at the sentence level within word-alignment community. As a sanity check, the alignments are also evaluated by computing scores on each fold and then averaging the results. Both approaches yield nearly the same scores. Therefore, throughout this thesis, only the results that are obtained by evaluating on all data at once are reported.

2. **Set of Templates:** Three different templates were used:

- T_E is the set of expansion templates from Table 4.1,
- T_D is the set of deletion templates from Table 4.2, and
- T_{MW} is the set of multi-word templates from Table 4.4 (supplemented with templates from Table 4.5).

3. **Instantiation of Templates:** Two different instantiations were used:

- *sim*: Simple instantiation, where the words are instantiated using a specific POS tag, relation, parameter class or combination of those, and
- *gen*: Generalized instantiation, where the words can be instantiated using either simple instantiation or with a special keyword **anything**.

4. **Best Rule Selection:** Two different metrics were used:

- *acc*: The accuracy of the rule, and
- *aer*: The alignment error rate on the entire training data after applying the rule (Only sure alignment links are used as the ground truth to learn rules inside ALP. Therefore, *aer* here refers to the AER of sure alignment links).

We performed statistical significance tests on all alignment error rates reported below using two-tailed paired t-tests. Unless otherwise indicated, the differences between ALP and initial alignments were found to be statistically signif-

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow s$)	87.0	67.0	24.3
ALP[GIZA++($e \rightarrow s$), (T_D, T_{MW}), <i>gen, aer</i>]	85.6	76.4	19.3
GIZA++($s \rightarrow e$)	88.0	67.5	23.6
ALP[GIZA++($s \rightarrow e$), (T_D, T_{MW}), <i>gen, aer</i>]	87.1	76.7	18.4
GIZA++(int)	98.2	59.6	25.9
ALP[GIZA++(int), (T_D, T_{MW}), <i>gen, aer</i>]	88.8	72.3	20.3
GIZA++(union)	80.6	74.9	22.3
ALP[GIZA++(union), (T_D, T_{MW}), <i>gen, aer</i>]	86.3	79.2	17.4
GIZA++(gdf)	83.8	74.4	21.2
ALP[GIZA++(gdf), (T_D, T_{MW}), <i>gen, aer</i>]	86.9	80.5	16.4

Table 4.7: ALP Results Using Different Initial Alignments (on English-Spanish)

icant within the 95% confidence interval. Moreover, the differences among ALP variations themselves were statistically significant within 95% confidence interval.

4.5.1 Results for Different Initial Alignments

In the first set of experiments, ALP was applied to different initial alignments, using deletion and multi-word templates, generalized instantiation, and AER for the best rule selection.

Table 4.7 presents the precision, recall and AER results for 5 different initial alignments on English-Spanish data. With initial uni-directional alignments, ALP achieves a slightly lower precision but a significantly higher recall than those of the initial alignments. The relative reduction in alignment error rate is 20.6% for one direction (19.3% vs. 24.3%), and 22.0% for the other direction (18.4% vs. 23.6%). When ALP with uni-directional GIZA++ alignments was compared to grow-diag-final, ALP still brings about significant reductions in AER: 9.0% relative

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow c$)	70.4	68.3	30.7
ALP[GIZA++($e \rightarrow c$), (T_D, T_{MW}), <i>gen, aer</i>]	79.1	68.1	26.6
GIZA++($c \rightarrow e$)	66.0	69.8	32.2
ALP[GIZA++($c \rightarrow e$), (T_D, T_{MW}), <i>gen, aer</i>]	83.3	66.0	26.2
GIZA++(int)	94.8	53.6	31.2
ALP[GIZA++(int), (T_D, T_{MW}), <i>gen, aer</i>]	91.7	56.8	29.5
GIZA++(union)	58.3	84.5	31.6
ALP[GIZA++(union), (T_D, T_{MW}), <i>gen, aer</i>]	82.5	70.0	24.1
GIZA++(gdf)	61.9	82.6	29.7
ALP[GIZA++(gdf), (T_D, T_{MW}), <i>gen, aer</i>]	82.1	72.7	22.8

Table 4.8: ALP Results Using Different Initial Alignments (on English-Chinese)

reduction in one direction (19.3% vs. 21.2%), and 13.2% relative reduction in the other direction (18.4% vs. 21.2%).

With the intersection of the uni-directional alignments as the initial alignment, ALP results in a relative reduction of 9.6% in precision (88.8% vs. 98.2%), but the recall increases from 59.6% to 72.3%—a relative increase of 21.3%. Overall the alignment error rate is significantly lower than the initial alignment—a 21.6% relative reduction.

The union and grow-diag-final heuristic results in alignments with a higher precision and a higher recall than the previous three initial alignments. When ALP is run with these initial alignments, it increases recall to 79.2% for GIZA++(union), and to 80.5% for GIZA++(gdf) while increasing the precision to 86.3-86.9%. The AER is reduced to 17.4% with GIZA++(union) as initial alignment, and to 16.4% with GIZA++(gdf) as the initial alignment—a relative reduction of nearly 22% for both cases.

Similarly, Table 4.8 presents the precision, recall and AER results for 5 different initial alignments on English-Chinese data. As on English-Spanish data, ALP yields significantly lower error rates with respect to the initial alignments on English-Chinese data.

With initial uni-directional alignments, ALP achieves a lower recall but a significantly higher precision than those of the initial alignments. The relative reduction in alignment error rate is 13.4% for one direction (26.6% vs 30.7%), and 18.6% for the other direction (26.2% vs. 32.2%). When ALP with uni-directional GIZA++ alignments was compared to grow-diag-final, ALP still brings about significant reductions in AER: 10.4% relative reduction in one direction (26.6% vs. 29.7%), and 11.8% relative reduction in the other direction (26.2% vs. 29.7%).

With the intersection of the uni-directional alignments as the initial alignment, ALP results in a relative reduction of 3.6% in precision (91.7% vs. 94.8%), but the recall increases from 53.6% to 56.8%—a relative increase of 6%. Overall the alignment error rate is reduced from 31.2% to 29.5%—a significant error reduction of 5.4%.

The union and grow-diag-final heuristic result in alignments with an amazingly low precision and a very high recall when compared to the previous 3 initial alignments. When ALP is run with these initial alignments, it increases precision significantly—from 58.3% to 82.5% for ALP with GIZA++(union) as the initial alignment, and from 61.9% to 82.1% for ALP with GIZA++(gdf) as the initial alignment. On the other hand, the recall is significantly reduced: For ALP with

GIZA++(union) as the initial alignment, the recall goes from 84.5% to 70%, and for ALP with GIZA++(gdf) as the initial alignment, the recall significantly drops from 82.6% to 72.7%. Overall the AER is reduced to 24.1% with GIZA++(union) as initial alignment, and to 22.8% with GIZA++(gdf) as the initial alignment—a relative reduction of nearly 23% for both cases.

Another interesting observation is that GIZA++ behaves very differently for two languages, Spanish and Chinese. The recall values for two languages are very close to each other, but the precision of the alignments for English-Chinese are quite low with respect to that of English-Spanish. The intersection of the uni-directional alignments on English-Chinese data results in a lower precision, and lower recall than the intersection of the uni-directional alignments on English-Spanish. However, the most significant difference between the two languages occurs for the union and grow-diag-final method. The precision of the alignments on English-Chinese is too low while the recall is 8-10 points higher than the recall of alignments on English-Spanish. When ALP is applied to the English-Spanish data, the recall always goes up regardless of the initial alignments. This suggests that ALP reduces AER on English-Spanish data by adding missing links most of the time. On the other hand, the precision always goes up, and recall always goes down significantly (except for GIZA++(int)) on the English-Chinese data. This suggests that ALP reduces AER on English-Chinese data by mostly deleting links. The most likely cause for this distinction is that the initial alignments are low in precision, but high in recall.

English-Spanish

Alignments	Pr	Rc	AER
GIZA++(int)	98.2	59.6	25.9
ALP[GIZA++(int), T_E , gen , aer]	90.9	69.9	21.0
ALP[GIZA++(int), (T_D , T_{MW}), gen , aer]	88.8	72.3	20.3
GIZA++(gdf)	83.8	74.4	21.2

English-Chinese

Alignments	Pr	Rc	AER
GIZA++(int)	94.8	53.6	31.2
ALP[GIZA++(int), T_E , gen , aer]	88.5	63.9	25.5
ALP[GIZA++(int), (T_D , T_{MW}), gen , aer]	91.7	56.8	29.5
GIZA++(gdf)	61.9	82.6	29.7

Table 4.9: ALP Results Using GIZA++(int) as Initial Alignment (on English-Spanish and English-Chinese)

4.5.2 Results for Different Sets of Templates

ALP was run using the intersection of GIZA++($e \rightarrow s$) and GIZA++($s \rightarrow e$) alignments as the initial alignment in two different ways: (1) With T_E using the union of the unidirectional GIZA++ alignments as the validation set, and (2) with T_D and T_{MW} applied one after another. Table 4.9 presents the precision, recall and AER results on English-Spanish and English-Chinese data. GIZA++(gdf) is included in both tables for comparison purposes.

For English-Spanish data, using the expansion templates (T_E) against a validation set produced results comparable to the grow-diag-final method. For English-Chinese data, using the expansion templates produced a significantly better AER than that of GIZA++(gdf). The precision goes down by 6.6% but the recall goes up by 19.2%. As a result, the the relative reduction in AER is 18.3% with respect to the initial alignment GIZA++(int). When compared to GIZA++(gdf),

the relative reduction in AER is 14.1%. The major difference between the grow-diag-final heuristic and ALP is that ALP results in a much higher precision but also in a lower recall because ALP is more selective in creating a new link during the expansion stage. This difference is due to the additional constraints provided by word features. Grow-diag-final, on the other hand, leaves out only a small (but mostly incorrect) portion of the union alignment, which results in higher recall and lower precision scores.

The version of ALP that applies deletion (T_D) and multi-word (T_{MW}) templates sequentially achieves lower recall but higher precision than grow-diag-final on English-Spanish data set although the 3.3% relative reduction in AER was not statistically significant. The results for the English-Chinese data set is quite different. For this data set, using expansion templates yields significantly better results than using deletion (T_D) and multi-word (T_{MW}) templates sequentially. Moreover, using deletion and multi-word templates do not provide any improvement over GIZA++(gdf). This result also supports the claim that the improvement on the Chinese data set, when the deletion and multi-word templates are used, comes mainly from deleting lots of spurious links. Since the intersection alignment contains very few spurious links, these templates do not provide much help. On the other hand, when the expansion templates were used on the intersection alignment, which has a very high precision, ALP improves AER by adding links based on those initial reliable links.

English-Spanish

Alignments	Pr	Rc	AER
GIZA++(gdf)	83.8	74.4	21.2
ALP[GIZA++(gdf), (T_D, T_{MW}), <i>sim, aer</i>]	87.9	79.0	16.8
ALP[GIZA++(gdf), (T_D, T_{MW}), <i>gen, aer</i>]	86.9	80.5	16.4

English-Chinese

Alignments	Pr	Rc	AER
GIZA++(gdf)	61.9	82.6	29.7
ALP[GIZA++(gdf), (T_D, T_{MW}), <i>sim, aer</i>]	79.5	75.0	22.7
ALP[GIZA++(gdf), (T_D, T_{MW}), <i>gen, aer</i>]	82.1	72.7	22.8

Table 4.10: ALP Results Using Different Template Instantiations (on English-Spanish and English-Chinese)

4.5.3 Using Different Types of Instantiation

Table 4.10 presents the precision, recall, and AER scores for two different methods for instantiating the templates on two different data sets: English-Spanish and English-Chinese. For these experiments, GIZA++(gdf) was used as the initial alignment, deletion and multi-word templates were used as the set of templates, and alignment error rate was used for best rule selection.

On the English-Spanish data, using generalized instantiation instead of simple instantiating results in a slightly lower AER (16.4% vs. 16.8%). On the English-Chinese data, using simple instantiation results in a slightly higher AER (22.7% vs 22.8%). However, on both datasets, the difference between the alignment error rates is not significant when the method for instantiating the rules is changed.

English-Spanish

Alignments	Pr	Rc	AER
GIZA++(gdf)	83.8	74.4	21.2
ALP[GIZA++(gdf),(T_D, T_{MW}), <i>sim</i> , <i>acc</i>]	87.8	77.7	17.6
ALP[GIZA++(gdf),(T_D, T_{MW}), <i>sim</i> , <i>aer</i>]	87.9	79.0	16.8
ALP[GIZA++(gdf),(T_D, T_{MW}), <i>gen</i> , <i>acc</i>]	88.5	77.8	17.2
ALP[GIZA++(gdf),(T_D, T_{MW}), <i>gen</i> , <i>aer</i>]	86.9	80.5	16.4

English-Chinese

Alignments	Pr	Rc	AER
GIZA++(gdf)	61.9	82.6	29.7
ALP[GIZA++(gdf),(T_D, T_{MW}), <i>sim</i> , <i>acc</i>]	79.6	75.1	22.7
ALP[GIZA++(gdf),(T_D, T_{MW}), <i>sim</i> , <i>aer</i>]	79.5	75.0	22.7
ALP[GIZA++(gdf),(T_D, T_{MW}), <i>gen</i> , <i>acc</i>]	81.8	73.0	22.8
ALP[GIZA++(gdf),(T_D, T_{MW}), <i>gen</i> , <i>aer</i>]	82.1	72.7	22.8

Table 4.11: ALP Results Using Different Rule Selection Methods (on English-Spanish and English-Chinese)

4.5.4 Using Different Methods for Best Rule Selection

Table 4.11 presents the precision, recall, and AER scores using two different methods of best-rule selection in each iteration. The table presents results for two data sets and also for two different template instantiation methods.

On the English-Spanish data set, using *aer* instead of *acc* as the best rule selection metric gives a reduction of 0.8 points with either template instantiation method. On the English-Chinese data set, using accuracy or alignment error rate does not make any difference in terms of precision, recall, and AER, regardless of what method is used for instantiating the templates. On both data sets, the differences between the alignment error rates are not statistically significant, which indicates that the metric used for choosing the best rule in each iteration does not affect the overall results.

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow c$)	72.2	70.5	28.6
ALP[GIZA++($e \rightarrow c$), (T_D, T_{MW}), <i>gen, aer</i>]	80.2	70.2	25.0
GIZA++($c \rightarrow e$)	67.5	71.6	30.6
ALP[GIZA++($c \rightarrow e$), (T_D, T_{MW}), <i>gen, aer</i>]	84.3	67.3	24.9
GIZA++(int)	93.0	55.9	29.9
ALP[GIZA++(int), (T_D, T_{MW}), <i>gen, aer</i>]	90.5	59.4	28.0
GIZA++(union)	60.6	86.3	29.5
ALP[GIZA++(union), (T_D, T_{MW}), <i>gen, aer</i>]	83.2	72.0	22.6
GIZA++(gdf)	64.0	84.4	27.7
ALP[GIZA++(gdf), (T_D, T_{MW}), <i>gen, aer</i>]	82.8	74.0	21.7

Table 4.12: ALP Results Using Different Initial Alignments When GIZA++ is Trained on More Data (on English-Chinese)

4.5.5 Effects of Training Size for Input Aligners

This section discusses whether there is room for improvement when GIZA++ is trained using more training data. For this experiment, a training set of 241K sentence pairs from the FBIS corpus (nearly 9.2M English and 7.3M Chinese words) was used for training GIZA++. As in the initial experiments, training involved 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4. The average sentence length for this training data is 38 words for English and 30 words for Chinese.

Table 4.12 summarizes the precision, recall and AER scores for each of the 5 initial alignments that are used in earlier experiments. For each of the 5 initial alignments, all scores except the precision value for GIZA++(int) went up nearly 2% (absolute increase). As before, the best precision (93.0%) is obtained by GIZA++(int), the best recall (86.3%) is obtained by GIZA++(union), and the

lowest AER (27.7%) is obtained by GIZA++(gdf).

Table 4.12 also presents the results when ALP is applied to each of these 5 initial alignments. In the best case, when GIZA++(gdf) was used as the initial alignment, the relative increase in precision is 29.4% (82.8% vs. 64.0%). As in earlier experiments, the recall value decreases by 12.3% relatively (74.0% vs. 84.4%). Overall ALP yields an AER of 21.7%—a relative reduction of 21.6% against GIZA++(gdf). The relative improvement is very close to the one ALP produces when a smaller training data was used to obtain the initial alignments (21.6% vs 23.2%). For other initial alignments, similar relative improvements are achieved, which indicates that ALP is still effective, even with better initial alignments.

4.5.6 Analysis of ALP Rules

This section investigates the differences between DUSTER rules and the rules generated by DUSTER in order to understand why ALP significantly performs better than DUSTER. The behavior of ALP is examined according to language pairs.

The major differences between DUSTER rules and the rules generated by ALP can be summarized as follows:

1. DUSTER rules are verb-oriented, i.e., only the translation divergences related to verbs are handled. In contrast, ALP also discovers rules that correct alignment links related to nouns, adjectives, etc. As a result, ALP handles

a larger portion of the corpus when compared to DUSTer.

2. DUSTer rules are more complex than the rules discovered by ALP. Most of the time, DUSTer needs to match 3 to 5 nodes in the subtrees against initial alignment in order to apply a rule. As a result, most of those DUSTer rules could not be applied to the corpus that was used in the experiments.⁶ ALP, on the other hand, attempts to correct frequent alignment errors by examining only a small window of 3 words, with fewer restrictions on the initial alignment. This leads to discovery of simpler rules than DUSTer and results in coverage of a bigger subset of the corpus when compared to DUSTer.
3. ALP not only covers translation divergences (the only purpose of DUSTer), but also generates transformation rules associated with common alignment errors, which may have no relation to translation divergences. A simple example is the incorrect alignment of punctuations and verbs.

Appendix C and D list the first 20 transformation rules discovered by ALP on English-Spanish and English-Chinese data. A comparison of DUSTer rules and ALP rules on English-Spanish data show that:

⁶An analysis of DUSTer rule application step demonstrated that only a few DUSTer rules were actually applied to English-Spanish data. The rest of the rules could not be applied because either the alignment links that are imposed by the rule could not be found in the initial alignment, or the rule covers a linguistic phenomena that do not exist in the given corpus.

1. The first 3 DUSter rules in Appendix B are combined into one rule by ALP (rule 7 in Appendix C).
2. The longer DUSter rules are not captured by ALP.
3. ALP captures several rules that are related to nouns that are not handled by DUSter.

It is worth noting that the rules that are generated by ALP are constrained by the space defined by the transformation templates. For efficiency, this thesis constrains the templates so that at most 3 consecutive words are examined on each side. DUSter, however, can handle words with longer dependencies and with more words. Theoretically, the template set that is used by ALP can be extended to deal with longer dependencies and more words, but at a higher computational cost.

The set of rules generated by ALP is strongly influenced by the behavior of the initial aligner. ALP relies on the fact that the initial aligner is good at capturing certain types of links and the errors usually follow a pattern. The performance of existing alignment systems (e.g., GIZA++) is better for languages that are close to each other, e.g., English and Spanish, and worse for structurally different languages, e.g., English-Chinese. As a result, ALP is more successful at generating multi-word correction rules for English-Spanish than English-Chinese because those templates rely on phrases that are aligned partially correctly. When the precision of the initial aligner is low (e.g., English-Chinese), ALP cannot suc-

cessfully generate multi-word correction rules. A closer look at the rules that are generated by ALP on English-Chinese data (Appendix D) supports this claim, where all the rules are simple deletion rules.

ALP yields similar relative improvements on both data sets, but the source of improvement is different: On English-Spanish data, ALP improves alignments mostly by adding new alignment links. On English-Chinese data, most of the improvement comes from deletion of several incorrect links that are generated by the initial aligner. This suggests that improvement of the performance of ALP requires the identification of the sources of errors in the initial alignment so that the templates are correctly chosen. For this purpose, it might be useful to do an error analysis on the initial alignment based on linguistics properties of the words (as in Section 3.2), and adjust the set of templates accordingly.

4.6 Summary

This chapter presented a new approach to classify the errors made by existing alignment systems, and to improve alignments by correcting those common errors. An adaptation of transformation-based error-driven learning to the problem of word alignment has been presented.

This approach has been evaluated on two different data sets and the results indicate that alignment link projection yielded significant improvements over the initial alignments. The best results are obtained by using deletion and multi-

word correction templates, generalized instantiation of templates, and alignment error rate for choosing the best rule in each iteration. With these settings, ALP yields an AER of 16.4% on English-Spanish data when GIZA++(gdf) is used as initial alignment—a relative error reduction of 22%. Similarly, ALP achieves an AER of 22.8% on English-Chinese data when GIZA++(gdf) is used as the initial alignment—a relative error reduction of 23%. This chapter demonstrated that ALP brings similar relative improvements—even when better initial alignments are used—due to the use of more data in training the initial aligner.

The effectiveness of ALP depends on the behavior of the initial aligner and the set of templates that are used to identify common errors. For obtaining the best possible performance, it might be useful to analyze the types of errors the initial aligner makes, and adjust the set of templates accordingly.

ALP solves the problems related to biases of existing alignment systems and lack of enough training data by automatically learning the alignment errors and correcting them. Moreover, if the initial alignment systems make frequent errors on function words and words related to translation divergences, these errors can be identified and corrected automatically using manually aligned data for training. Finally, it is easy to inject linguistic knowledge into the alignment process without modifications to the initial alignment systems. This is accomplished through rule learning based on linguistic properties of the words.

Chapter 5

Multi-Align: A Framework for Combining Multiple Alignments

The previous chapter described how an error-driven learning approach can be used to improve one word-alignment system. Another way to improve a given word alignment is to validate it against another one and to identify the places where they agree and where they do not. An easy solution is one that uses the intersection or union of two alignments to obtain an alignment with either high precision or high recall. A more complicated solution is one where the outputs of different systems are combined using an additional model.

In this chapter, a new framework based on the concept of classifier ensembles, *Multi-Align*, is proposed for incremental testing of different alignment algorithms and their combinations. In the Multi-Align framework, each alignment system is treated as a classifier, and then the alignment combination problem is transformed into a classifier ensemble problem.

A classifier's ability to meaningfully respond to novel patterns, or generalize,

is perhaps its most important property; however, the generalization is not unique, and different classifiers provide different generalizations (Tumer and Ghosh, 1996a). Selecting the *best* classifier is not necessarily the ideal choice, because other *less successful* classifiers might provide valuable information that is not utilized when a particular hypothesis is selected.

“It is now recognized that the key to pattern recognition problems does not lie wholly in learning machines, statistical approaches, spatial filtering,..., or in any other particular solution which has been vigorously advocated by one or another group during the last one and a half decades as the solution to the pattern recognition problem. ... No single model exists for all pattern recognition problems and no single technique is applicable to all problems. ... Rather what we have is a bag of tools and a bag of problems (Kanal, 1974).”

“To solve really hard problems, it is time to stop arguing over which type of pattern classification technique is best Instead we should work at a higher level of organization and discover how to build managerial systems to exploit the different virtues and evade the different limitations of each of these ways of comparing things (Minsky, 1991).”

These observations lead to the concept of combining different classifiers, i.e., *ensemble* of classifiers. In the ensemble framework, the final decision of classification is made by using outputs of several classifiers, in an attempt to reduce biases

of individual learning algorithms.

Ensembles of classifiers have been widely used in many machine learning applications (e.g., pattern recognition). However, they haven't been used much by researchers in machine translation and word alignment. Although ensembles have been shown to perform better than individual systems, the trend within MT community is to build everything from scratch instead of combining different systems and taking advantage of the merits of different systems.

This thesis introduces a new framework that is based on classifier ensembles to combine different word alignments. The motivation behind this framework is that ensembles of word alignments systems will lead to elimination of most of the errors produced by the individual systems, as it has been shown in many machine learning applications. As a result, a combination approach will yield alignments that are more accurate than those produced by existing systems. Moreover, this approach eliminates the need for rebuilding existing alignment systems when incorporating new information such as linguistic knowledge.

This chapter also introduces the concept of search space decomposition, which has been previously applied to other problems (Tumer and Ghosh, 1996a), into word alignment problem. All existing alignment systems attempt to build one alignment model, without distinguishing alignment links. The alignment combination framework in this thesis divides alignment links into different partitions (i.e., link classes) based on linguistic features of words, and builds a different alignment model for each partition. Later in this chapter, it will be shown that such a decom-

position of the search space allows the system to generalize differently for different partitions and to obtain a better alignment.

The remainder of this chapter presents a thorough background on classifier ensembles, followed by a description of a new framework for combining multiple alignments for an improved word alignment. Finally, an implementation of Multi-Align is presented in which perceptrons are used for combining multiple alignments.

5.1 Classifier Ensembles

Formally, the classification problem is defined as follows (Dietterich, 1997): In supervised learning, a learning program is given training examples of the form $\{(x_1, y_1), \dots, (x_m, y_m)\}$ for some unknown function $y = f(x)$. The x_i values are typically vectors of the form $(x_{i1}, x_{i2}, \dots, x_{in})$ whose components are discrete or real-valued. The components x_{ij} are called the *features* of x_i . The y values are typically drawn from a discrete set of classes $\{1, \dots, K\}$ in the case of *classification*. Given a set S of training examples, the learning algorithm outputs a *classifier* h — a hypothesis about the true function f — from the hypothesis space H . An *ensemble* of classifiers is a set of classifiers, $\{h_1, \dots, h_L\}$, whose individual outputs are combined in some way to classify new examples. The resulting classifier is a function of the individual hypotheses, i.e., $h_e = g(h_1, \dots, h_L)$.

Two obvious measures comparing the error of the ensemble (ϵ_e) to the error of an individual classifier (ϵ_i) are: (1) Error difference ($\epsilon_i - \epsilon_e$) and (2) Error ratio ($\epsilon_r =$

ϵ_e/ϵ_i) (Ali and Pazzani, 1996). Error ratio is a more reasonable approximation to the *usefulness* of the ensemble, when the error rates of the individual classifiers are low.

5.1.1 Why Ensembles Work

Dietterich (2000) provides an informal reasoning, from statistical, computational and representational viewpoints, of why ensembles of classifiers produce better results than individual classifiers.

1. **Statistical Viewpoint:** A learning algorithm can be viewed as searching a space H of hypotheses to identify the *best* hypothesis in the space. Training data may not provide sufficient information for choosing a single best classifier from hypothesis space H because there can be many hypotheses in H that give the same classification accuracy on the training data. An ensemble of classifiers can *average* the votes of individual hypotheses and reduce the risk of choosing the wrong one.
2. **Computational Viewpoint:** Many learning algorithms work by performing some form of local search that may get stuck in local optima (e.g., neural networks). Even if there is sufficient data, it may still be very difficult to reach the best hypothesis. For instance, it has been proven that optimal training of both neural networks and decision trees is NP-hard. Constructing an ensemble allows one to start local searches from different points. Therefore, an

ensemble may provide a better approximation to the true unknown function than that of any of the individual classifiers.

3. **Representational Viewpoint:** The hypotheses space H may not contain the true function f . Instead, it may include several equally good approximations to f . By forming weighted sums of hypotheses drawn from H , it may be possible to expand the space of representable functions.

These three fundamental issues are the the most important ways in which existing learning algorithms fail. Hence, ensemble methods *have the promise of* reducing (and perhaps even eliminating) these three key shortcomings of standard learning algorithms.

5.1.2 When Ensembles Work

The main goal of constructing an ensemble of classifiers is to reduce the classification error of the individual classifiers. The Bayes error¹ provides the lowest achievable error rate for a given pattern classification problem (Tumer and Ghosh, 1996a). Combining different classifiers is an attempt to reduce the error to the level of Bayes error.

The expected error of an algorithm can be divided into two components:

¹The classifier that assigns a vector x to the class with the highest posterior is called the Bayes classifier, and the error associated with this classifier is called the Bayes error (Tumer and Ghosh, 1996b).

(1) Bias, which is the consistent error that the algorithm makes over many different runs (i.e., measuring how close the average classifier produced by the learning algorithm will be to the target function f); and (2) Variance, which is error that fluctuates from run to run (i.e., measuring how much each of the learning algorithm's guesses will vary with respect to each other) (Krogh and Vedelsby, 1995; Bay, 1998).² Formally, bias is the error of the *ideal voted hypothesis*, which is the result one would obtain from combining an infinite number of classifiers, each trained on an independent set of examples. Variance is the difference between the expected error rate and the ideal voted hypothesis error rate (Kong and Dietterich, 1995). Combining is primarily a way of reducing model variance, though in certain situations it also reduces bias (Ghosh, 2002).

If the average error rate for an example is less than 50% and the component classifiers in the ensemble are independent in the production of their errors, the expected error for that example can be reduced to zero as the number of classifiers combined goes to infinity (Hansen and Salamon, 1990). In other words, for an ensemble of classifiers to be more accurate than its components, the following conditions should be satisfied:

1. The classifiers should be *accurate*, i.e., the error rate is lower than random guessing,

²Note that the Bayes error rate is included in the bias term in this framework. A different decomposition of error (Opitz and Maclin, 1999) represents Bayes error rate as a third component of the error.

2. The classifiers should be *diverse*, i.e., they make different errors on specific instances.

The first condition corresponds to a good generalization of individual classifiers while the latter ensures that, when they do make errors on new data, these errors are not shared with any other models, i.e., the individual classifiers generalize differently (Schapire, 1990; Rogova, 1994; Dietterich, 1997). Satisfying the first condition is relatively easy: There have been several classification algorithms that have been proven to be successful in different problems and domains. Hence, the critical issue when constructing an ensemble is choosing classifiers whose errors are at least somewhat uncorrelated.

Error correlation, ϕ_e , measures the proportion of the test examples on which members of an ensemble make the same kinds of classification errors (Ali and Pazzani, 1996). Two models are said to make a correlated error when they both classify an example of class i as belonging to class j , where $j \neq i$. Formally, let ϕ_{ij} denote the correlation between the i -th and j -th models. Let $h_i(x) = y$ denote the event that model i has classified example x to class y . Let $f(x)$ denote the true class of x . Then,

$$\phi_{ij} = p(h_i(x) = h_j(x), h_i(x) \neq f(x))$$

Let $e = \{h_1, \dots, h_L\}$ be the ensemble of L models. Then, ϕ_e , the degree to which the errors in e are correlated, has the following definition:

$$\phi_e = \frac{1}{L(L-1)} \sum_{i=1}^L \sum_{j \neq i}^L \phi_{ij}$$

When the fraction of correlated errors, ϕ_e , is small, multiple models have a substantial impact on reducing the error.

Note that this definition of error correlation is not *normalized*: its maximum value is the lower of the two classification errors. An alternative definition of error correlation attempts to normalize this value by defining ϕ_e as the conditional probability that both classifiers make the same error, given that one of them makes an error (Gama, 1999):³

$$\phi_{ij} = p(h_i(x) = h_j(x) | h_i(x) \neq f(x) \vee h_j(x) \neq f(x))$$

.

Another attempt to quantify the diversity of the classifiers is the definition of complementary rate of classifiers (Brill and Wu, 1998). The complementary rate of classifiers A and B is defined as:

$$Comp(A, B) = (1 - \frac{|E_A \cap E_B|}{|E_A|}) * 100$$

where E_A is the set of instances that A made an error and E_B is the set of instances that B made an error. $Comp(A, B)$ measures the percentage of time when tagger A is wrong that tagger B is correct (note that $Comp(A, B) \neq Comp(B, A)$), and therefore providing an upper bound on combination accuracy.⁴

³Note that the first definition of ϕ_{ij} is not symmetric but the second is symmetric.

⁴Correlation error attempts to quantify the degree to which two classifiers make correlated errors while complementary rate attempts to quantify the degree to which a classifier make errors that are uncorrelated with any other classifier.

If the complementary rate of classifiers A and B is low, i.e., if all the classifiers made the same errors, or if the errors that lower-accuracy classifiers made were merely a superset of higher accuracy classifier errors, then the ensemble method would not provide much improvement. It has been shown empirically that the complementarity between classifiers decreases while the size of the training data increases (Banko and Brill, 2001), due in part to the fact that a larger training corpus will reduce the data set variance and any bias arising from this.

5.1.3 Creating Ensemble Members

The choice of members for a classifier ensemble is an area of extensive research.

Generally speaking, ensemble members can be created in two different ways:

1. **Different Generalizations of the Same Algorithm:** The best known and widely explored technique is (adaptive) re-sampling. A set of models for an ensemble is commonly created by using some form of sampling, such that each model in the ensemble is trained on a different sub-sample of the training data. Some techniques for resampling (or subsampling) are bagging (Breiman, 1996), wagging (or weighted bagging) (Webb, 2000), boosting (Schapire, 1990; Freund and Schapire, 1997), k-fold partitioning (or cross-validated committees) (Parmanto et al., 1996; Tumer and Ghosh, 1996a). Resampling techniques might be effective in producing models that generalize differently but they will not necessarily reduce the test set error. If the

training data for each classifier is not sufficient to generalize effectively, the error rate of individual classifiers increases, which leads to increased error rates for the ensemble (Frayman et al., 2002). Some other methods that are used to generate different versions of the same algorithm are injecting randomness to the algorithm (as in neural networks) and manipulation of the feature space.

The approach used in this thesis is similar to decomposition of complex problems into smaller problems (i.e., local experts) (Tumer and Ghosh, 1996a). This involves spatial partitioning of training instances using the proximity of patterns in the input space such that the complexity of each classifier's task is reduced (weighted expert combining). For local experts, multi-layer perceptrons or RBF networks have been used (Tumer and Ghosh, 1996a). Mixtures-of-experts (Jacobs et al., 1991) and hierarchical mixtures-of-experts (Jordan and Jacobs, 1994) also employ this type of problem decomposition.

2. **Different Classifiers Based on Different Models:** Different learning algorithms can be used to train different models on the same training data. The basic motivation is that the use of different learning methods is more likely to result in different patterns of generalization (Frayman et al., 2002). Choosing algorithms that are complementary to each other is critical for success of the ensembles. Hence, when classifiers from different learning algorithms are combined, they should be checked (e.g., by cross validation) for

accuracy and diversity, and some form of weighted combination should be used (Dietterich, 1997). For instance, Frayman et al. (2002) used a linear method (linear regression), an efficient nonlinear method (multi-layer perceptrons), a logistic regression (MLP with a single hidden node) and a clustering algorithm (k - nearest neighbor) as members of the ensemble in their study.

5.1.4 Combining Ensemble Members

The methods for combining ensemble members can be divided into two groups:

1. **Cooperative Combination:** It is assumed that all of the ensemble members will make some contribution to the ensemble decision, even though this contribution may be weighted in some way. The most popular techniques are uniform and weighted voting, where majority wins all. Voting can be effective in reducing both the bias of a particular training corpus and the bias of a specific learner. When a training corpus is very small, there is much more room for these biases to surface and therefore for voting to be effective. With large training data, little is gained by voting; indeed voting might hurt accuracy (Banko and Brill, 2001). In weighted voting, each classifier h_l generates a class probability estimate rather than a simple classification, and the final classifier output for a specific instance x is the class with the highest probability, i.e., $\operatorname{argmax}_{c \in C} \sum_{1 \leq l \leq L} p(c|x, h_l)$. Bernardo and Smith (1994) motivate this approach which states that to maximize predictive accuracy,

instead of making classifications based on a single learned model, one should ideally use all hypotheses (models) in the hypothesis space.

The approach adopted in this thesis is based on *stacked generalization* (or *stacking*) (Wolpert, 1992; Ting, 1996). An additional model is used to learn how to combine the models with weights that vary over the feature space. Stacking algorithms first generate outputs from individual (level-0) classifiers, and then use these outputs to learn a level-1 classifier. To generate a training set for a level-1 classifier, a leave-one-out or cross-validation procedure is applied. First, each of the level-0 classifiers are trained on almost the entire training set, leaving one example for testing. Next, each classifier is applied to the test instance. Finally, these outputs are used as the feature set for learning the level-1 classifier. The level-1 classifier might use the original input features as well as the outputs of level-0 classifiers. Note that uniform voting and weighted voting are simple instantiations of stacked generalization.

Stacked generalization is intended to combine different algorithms, rather than different instantiations of the same algorithm on resampled data; however, it has been shown that stacking with model trees performs better than other combining methods regardless of the choice of the individual base classifiers (Dzeroski and Zenko, 2002).

2. Competitive Combination: It is assumed that for each input, only the

most appropriate ensemble member will be selected based on either the inputs or outputs of the ensemble members. Two common techniques for competitive combination are gating and rule-based switching. In gating, under the divide and conquer approach employed by mixtures-of-experts (Jacobs et al., 1991) and hierarchical mixtures-of-experts (Jordan and Jacobs, 1994), a more complex problem is decomposed into a set of simpler problems. The data is partitioned into regions and a different classifier is learned for each region. A gating model is used to output a set of scalar coefficients that weights the contributions of the various inputs.

In rule-based switching, the models may be switched on the basis of the input or the output of one of the models (Ting, 1996). It has been shown that better results are obtained through the use of a more explicit rule-based switching between models (Frayman et al., 2002).

5.1.5 Ensembles in NLP

Some of the applications in NLP that employ ensemble methods are as follows:

1. POS tagging (Brill and Wu, 1998; van Halteren et al., 1998; Marquez et al., 1998; Abney et al., 1999; Marquez et al., 1999)
2. PP attachment (Abney et al., 1999)
3. Base noun phrase identification and chunking (Tjong Kim Sang et al., 2000)

4. Word sense disambiguation (Escudero et al., 2000; Pedersen, 2000; Florian and Yarowsky, 2002)
5. Statistical parsing (Henderson and Brill, 2000)
6. Automatic thesaurus extraction (Curran, 2002)
7. Named entity recognition (Collins and Singer, 1999; Carreras et al., 2002; Wu et al., 2002)

All of these methods employ either bagging or boosting techniques to generate different generalizations, and then combine them via voting methods. The work reported in this thesis is the first application of classifier ensembles to the word alignment problem. In contrast to other NLP applications, an additional model for combining classifiers is learned, following the idea behind stacked generalization.

5.2 Multi-Align

Multi-Align is a general alignment framework where the outputs of different aligners are combined to obtain an improvement over the performance of any single aligner. As described in Section 5.1, classifier ensembles have proven successful in many machine learning applications. Multi-Align is an attempt to take advantage of classifier ensembles in the context of word alignments.

In the Multi-Align framework, one can combine any number of word alignments systems, regardless of the assumptions the individual aligners make or the

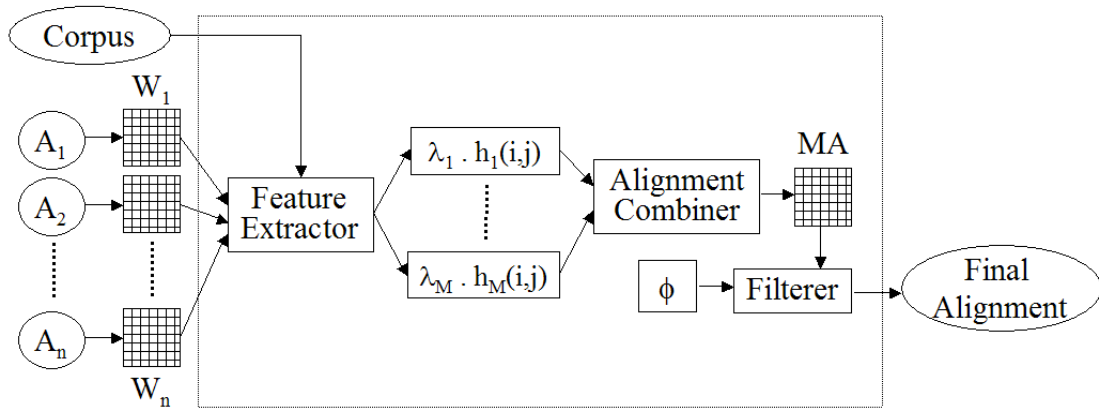


Figure 5.1: Multi-Align Framework for Combining Multiple Alignments

resources they employ. One of the motivations behind Multi-Align is the need for incorporation of linguistic knowledge into alignment modeling. As described in Section 2.2, different pieces of linguistic knowledge, such as POS tags and dependency trees, have been shown to improve word alignments. The Multi-Align framework provides a mechanism for combining linguistically-informed alignment approaches with statistical aligners without the need for complex modifications to existing systems. Moreover, the framework allows exploiting the best properties of each individual aligner.

Figure 1.3 illustrates the Multi-Align design. In this framework, first, n different word-alignments systems, A_1, \dots, A_n , generate word alignments between a given English sentence and a foreign language (FL) sentence. Then a *Feature Extractor* takes the output of these alignments systems and the parallel corpus (which might be enriched with linguistic features) and extracts a set of feature functions based on linguistic properties of the words and the input alignments. Each feature

function h_m is associated with a model parameter λ_m . Next, an *Alignment Combiner* uses this information to generate a single word-alignment matrix based on the extracted feature functions and the model parameters associated with them. The contribution of each feature function to a particular alignment link is proportional to the model parameter associated with that feature. In a final step, a *Filterer* filters the alignment links according to their confidence in the final alignment matrix. The decision to include a particular alignment link in the final alignment is based on a confidence threshold ϕ .

The basic data structure in Multi-Align is the *word alignment matrix*, W , as used in current statistical-alignment approaches (Och and Weber, 1998). W is a $I \times J$ matrix, where I is the number of words in the English sentence \mathbf{e} , J is the number of words in the foreign language sentence \mathbf{f} .⁵ Let e_i be the word in \mathbf{e} in position i and f_j be the word in \mathbf{f} in position j . Each entry of the matrix, W_{ij} , corresponds to the alignment link between e_i and f_j . The value of W_{ij} is the alignment probability for the alignment link between e_i and f_j .

Formally, assuming that n different aligners are used to generate the word alignments between two sentences, let A_k be the k^{th} alignment system and W_{A_k} be the word alignment matrix generated by this system. Each entry of W_{A_k} , say $W_{A_k}(i, j)$, is given a probability that the corresponding words e_i and f_j are aligned together.

⁵It is implicitly assumed that if an English word is not aligned to a FL word, then it is aligned to NULL word, and vice versa.

Multi-Align employs M feature functions $h_m(i, j)$, where $m \in \{1, \dots, M\}$, i corresponds to the index of an English word, and j corresponds to the index of the FL word. The feature function $h_m(i, j)$ can be linguistic properties of the words e_i or f_j , such as POS tags of the words. Moreover, a feature function $h_m(i, j)$ might be extracted from the alignment matrices generated by the individual alignment systems (e.g., the total number of existing links around the given link (i, j)).

In a simpler model, each feature function h_m is associated with a model parameter λ_m . The contribution of each feature to a particular alignment link is proportional to a model parameter associated with that feature, i.e., $\lambda_m \cdot h_m(i, j)$. Assuming that the set of feature functions $\{h_1(i, j), \dots, h_M(i, j)\}$ is represented by $h_1^M(i, j)$, and the set of model parameters $\{\lambda_1, \dots, \lambda_M\}$ is represented by λ_1^M , the final value of the alignment link between e_i and f_j is a function g of λ_1^M and $h_1^M(i, j)$:

$$MA_{ij} = g(\lambda_1^M, h_1^M(i, j))$$

For a given confidence threshold ϕ , e_i and f_j are aligned together if $MA_{ij} \geq \phi$.

In preliminary studies using Multi-Align, individual alignment systems were observed to handle some types of alignments better than others. For example, an individual aligner may handle the links between words with the same POS tags better than links between words with different POS tags. Thus, this thesis adopts a more generalized model where alignment links are categorized into different groups, or *link classes*, according to the POS tags of the words involved; different model parameters and confidence thresholds are then learned for link class. The moti-

vation behind the link classes is similar to that of the local experts described in Section 5.1.4. The more complex problem of word alignment is divided into smaller problems based on link classes. Learning different model parameters conditioned on link classes for each feature function is shown to induce better word alignments than using only one model parameter for each function.

This generalized model assumes that the links are divided into link classes and the definitions of the model parameters and confidence threshold are modified as follows: Let $lc(i, j)$ be the link class associated with the link (i, j) , and let $C = \{C_1, \dots, C_N\}$ be the set of different link classes. Each feature function h_m is associated with N model parameters, i.e., $\lambda_{m,C_1}, \dots, \lambda_{m,C_N}$. The contribution of each feature to a particular alignment link is proportional to a model parameter associated with that feature based on the link class, i.e., $\lambda_{m,lc(i,j)} \cdot h_m(i, j)$. Assuming that the set of feature functions $\{h_1(i, j), \dots, h_M(i, j)\}$ is represented by $h_1^M(i, j)$, and the set of model parameters $\{\lambda_{1,lc(i,j)}, \dots, \lambda_{M,lc(i,j)}\}$ is represented by $\lambda_1^M(i, j)$, the final value of the alignment link between e_i and f_j is a function g of $\lambda_1^M(i, j)$ and $h_1^M(i, j)$:

$$MA_{ij} = g(\lambda_1^M(i, j), h_1^M(i, j))$$

The definition of the confidence threshold is also modified so that there exists a different threshold for each link class rather than only one confidence threshold for all links. Assuming that the confidence threshold for a link class C_l is represented by ϕ_{C_l} , e_i and f_j are aligned if $MA_{ij} \geq \phi_{lc(i,j)}$.

Parameters of Multi-Align can be set manually or learned via machine learn-

ing algorithms. To illustrate the generality of the framework, this thesis shows how these parameters are set to obtain 3 different combined alignments of 2 systems, A_1 and A_2 : intersection, union, and a refined alignment method called *grow-diag-final* (Koehn et al., 2003). For the sake of complexity reduction, it is assumed that:

$$W_{A_k}(i, j) = \begin{cases} 1 & \text{if } (i, j) \in A_k \\ 0 & \text{otherwise} \end{cases}$$

The intersection of A_1 and A_2 can be computed using only one link class and 2 feature functions corresponding to the alignment matrices generated by A_1 and A_2 . There is only 1 model parameter for each function and each of these is set to 1. Using a weighted sum of features as a function g , it is sufficient to set the confidence threshold to 2 to obtain $A_1 \cap A_2$.⁶ Formally,

$$lc(i, j) = \mathbf{c}$$

$$h_1(i, j) = W_{A_1}(i, j)$$

$$h_2(i, j) = W_{A_2}(i, j)$$

$$\lambda_{1,\mathbf{c}} = \lambda_{2,\mathbf{c}} = 1$$

$$g(\lambda_1^2(i, j), h_1^2(i, j)) = \lambda_{1,\mathbf{c}} \cdot h_1(i, j) + \lambda_{2,\mathbf{c}} \cdot h_2(i, j) = W_{A_1}(i, j) + W_{A_2}(i, j)$$

$$\phi_{\mathbf{c}} = 2$$

$$A_1 \cap A_2 = \text{multi_align}(\{A_1, A_2\}, lc, \lambda_1^2, h_1^2, g, \phi_1^1)$$

The union of A_1 and A_2 can be computed analogously. The only difference

⁶The notation ϕ_1^N denotes the set of confidence thresholds $\{\phi_{C_1}, \dots, \phi_{C_N}\}$.

```

 $lc(i, j) = \mathbf{c}$ 
 $h_1(i, j) = W_{A_1}(i, j) \quad \lambda_{1,\mathbf{c}} = 1$ 
 $h_2(i, j) = W_{A_2}(i, j) \quad \lambda_{2,\mathbf{c}} = 1$ 
 $h_3(i, j) = nullE_{A_3}(i) \quad \lambda_{3,\mathbf{c}} = 1$ 
 $h_4(i, j) = nullF_{A_3}(j) \quad \lambda_{4,\mathbf{c}} = 1$ 
 $h_5(i, j) = neighbor\_exists_{A_3}(i, j) \quad \lambda_{5,\mathbf{c}} = 1$ 
 $h_6(i, j) = W_{A_3}(i, j) \quad \lambda_{6,\mathbf{c}} = 3$ 
 $g(\lambda_1^6(i, j), h_1^6(i, j)) = \sum_{m=1}^{m=6} \lambda_{m,\mathbf{c}} \cdot h_m(i, j)$ 
 $\phi_{\mathbf{c}} = 4$ 

grow_diag( $\mathbf{A}_1, \mathbf{A}_2$ )
 $A_{GD} = A_1 \cap A_2$ 
repeat until no more points can be added to  $A_{GD}$ 
     $A_{GD} = multi\_align(\{A_1, A_2, A_{GD}\}, lc, \lambda_1^6, h_1^6, g, \phi_1^1)$ 

 $lc(i, j) = \mathbf{c}$ 
 $h_1(i, j) = nullE_{A_1}(i) \quad \lambda_{1,\mathbf{c}} = 1$ 
 $h_2(i, j) = nullF_{A_1}(j) \quad \lambda_{2,\mathbf{c}} = 1$ 
 $h_3(i, j) = W_{A_1}(i, j) \quad \lambda_{3,\mathbf{c}} = 3$ 
 $h_4(i, j) = W_{A_2}(i, j) \quad \lambda_{4,\mathbf{c}} = 2$ 
 $g(\lambda_1^4(i, j), h_1^4(i, j)) = \sum_{m=1}^{m=4} \lambda_{m,\mathbf{c}} \cdot h_m(i, j)$ 
 $\phi_{\mathbf{c}} = 3$ 

final( $\mathbf{A}_{init}, \mathbf{A}_{expand}$ )
 $A_{GDF} = multi\_align(\{A_{init}, A_{expand}\}, lc, \lambda_1^4, h_1^4, g, \phi_1^1)$ 

```

Figure 5.2: Representation of Grow-diag-final Method in Multi-Align Framework

is the setting for the confidence threshold: To obtain $A_1 \cup A_2$, it is sufficient to set $\phi_{\mathbf{c}}$ to 1 instead of 2, keeping the other parameters the same as before.

Figure 5.2 presents one of many possible ways for instantiating *grow-diag-final* (Koehn et al., 2003) in the Multi-Align framework. Koehn’s approach includes two parts: *grow-diag* and *final*. The original algorithm works iteratively, where new links are added based on the results of previous iteration. For each iteration, assuming that only one link class is used for the alignment links, 6 features, and 6

model parameters associated with these features are used to represent *grow-diag*. The first two features control whether the added link is in the union of the initial alignments. The next 3 features control whether the words in the new link (i, j) has been aligned or not, and whether there is a neighboring link to (i, j) . The final feature is used to keep the links that already exist in the starting alignment. Using a weighted sum of the features, setting the confidence threshold to 4 is sufficient for obtaining the same alignment links in Koehn’s *grow-diag*. This confidence threshold allows us to keep the links in the intersection alignment and the starting alignment at subsequent iterations. Moreover, new links are added if and only if the words involved have been unaligned, and there is at least one neighboring link. Note that the original algorithm depends on the order of the links that are visited. To obtain exactly the same alignments, multi-align function should be implemented the same way.

The representation of the *final* function is similar. The model includes 4 features, where the first two control whether one of the words is unaligned. The third feature enables the retention of links in the initial alignment, and the fourth feature guarantees the addition of links that are only in the given alignment. Given the model parameters in Figure 5.2, using a weighted sum of the features and setting the confidence threshold to 3 is sufficient to obtain the alignment links in the original algorithm. Using the *grow-diag* and *final* functions in Figure 5.2, *grow-diag-final* can be computed on two input alignments A_1 and A_2 as follows:

grow_diag_final(A_1, A_2)

$A_{GD} = grow_diag(A_1, A_2)$

$A_{GDF} = final(A_{GD}, A_1)$

$A_{GDF} = final(A_{GDF}, A_2)$

Multi-Align has three advantages with respect to MT systems: ease of adaptability, robustness, and user control.

1. **Ease of Adaptability:** Multi-Align eliminates the need for complex modifications of pre-existing systems to incorporate new linguistic resources. A variety of different statistical and symbolic word alignment systems may be used together, such as statistical alignments (Och, 2000), bilingual dictionaries acquired automatically using lexical correspondences (Ker and Chang, 1997; Melamed, 2000), lists of closed-class words and cognates, tree-based alignment systems (Cherry and Lin, 2003), and linguistically-motivated alignments (Dorr et al., 2002).
2. **Robustness:** Individual alignment systems have inherent deficiencies that result in partially incorrect alignments. Multi-Align relies on the strengths of certain systems to compensate for the weaknesses of other systems.
3. **User Control:** The effect of different linguistic information is difficult to observe and control when linguistic knowledge is injected into statistical systems. Multi-Align avoids this problem by helping users to understand which

linguistic resources are useful for word alignment. Additionally, the contribution of each aligner may be weighted according to its impact on the target application. For example, if the end application is bilingual-lexicon construction, the alignment of function words may not be as important as the alignment of high-content words. Therefore, links involving functional words can be assigned lower confidence values than the other alignment links. On the other hand, if the end application is statistical machine translation, all alignment links are of the same importance, and a complete set of alignment links is preferable.

5.3 Preliminary Study: Alignment Combination Using Weighted Summation

Multi-Align makes use of five important parameters for determining whether two words e_i and f_j should be aligned:

1. $lc(i, j)$: How to choose the features that divide the alignment links into different groups
2. $h_1^M(i, j)$: How to choose the features used for combining alignments
3. $\lambda_1^M(i, j)$: How to set the parameters associated with the features for each link class
4. g : How to combine the model parameters associated with the features

5. $\phi_{lc(i,j)}$: How to set the confidence threshold for each link class

This section presents a version of Multi-Align where features include linguistic properties of the words involved and alignment-based features. Later a supervised learning method will be presented that estimates the model parameters associated with the features and the confidence threshold.

5.3.1 Set of Features

Two types of features to divide the links to different groups and to combine different alignments are used:

1. Linguistic features
2. Alignment-based features

For a particular alignment link (i, j) , linguistic features include POS tags of both words (e_i and f_j) and a dependency relation for one of the words (e_i). POS tags are generated using the MXPOST tagger (Ratnaparkhi, 1996) for English and Chinese, and Connexor for Spanish. Dependency relations are produced using a version of the Collins parser (Collins, 1997) that has been adapted for building dependencies. The details of how POS tags and dependency relations are generated for the data sets used in the experiments were described in Section 4.3.3.

Alignment-based features are extracted from the outputs of individual alignment systems. For each alignment A_k , the following are some of the alignment features that can be used to describe an instance (i, j) :

1. Whether (i, j) is an element of A_k or not
2. Fertility of (i.e., number of words in \mathbf{f} that are aligned to) e_i in A_k
3. Fertility of (i.e., number of words in \mathbf{e} that are aligned to) f_j in A_k
4. For each neighbor $(x, y) \in N(i, j)$, whether $(x, y) \in A_k$ or not

When working with a limited amount of manually annotated data, using too many features leads to data sparseness, which usually results in poor generalization (overfitting). Therefore, it might be better to use variants, or combinations, of these features to reduce feature space. For instance, to avoid data sparseness, it might be better to use only one feature to represent the number of existing alignment links in the neighborhood rather than to use one feature for each neighbor.

5.3.2 Perceptron Learning

A perceptron is a *program* that learn concepts from a given set of training inputs, i.e., it learns whether the presented input is **true** or **false**. Since it was introduced in 1958 by Frank Rosenblatt, perceptrons have become one of the most commonly used learning techniques due to their simplicity and ability to generalize easily from a small amount of training data. They have been shown useful in many areas, especially for simple problems in pattern classification.

The input to the perceptron network is a vector \mathbf{p} with R features, i.e., the input layer consists of R input units. The network consists of a single output layer of S perceptron neurons, which is connected to R input units through a set of

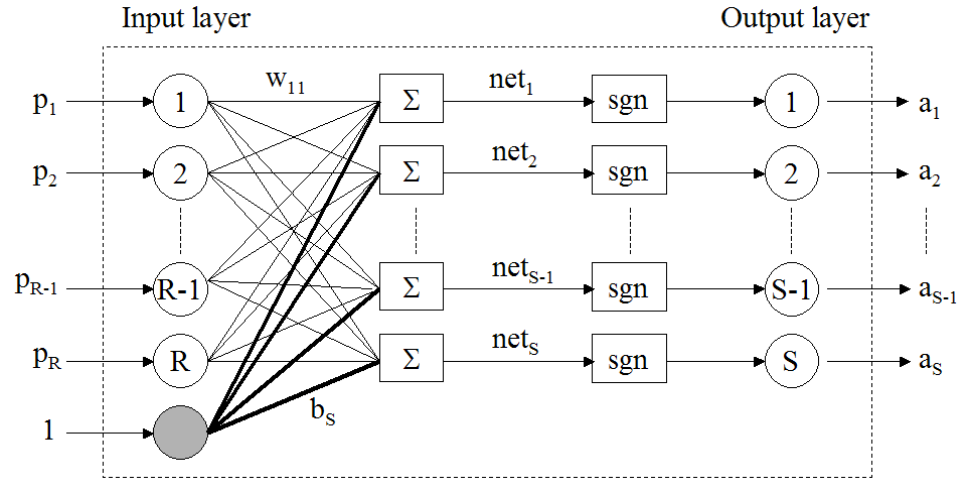


Figure 5.3: A Perceptron Network with R Input Units and S Output Units

weights. As illustrated in Figure 5.3, each input unit j is connected to each output unit i through a weight w_{ij} . The value of each output unit is computed by passing the weighted sum of the input units (Σ in Figure 5.3), or the *net input*, through a hard limit transfer function (sgn in Figure 5.3). As a result, each output unit has a value of either 0 or 1. Geometrically, this is equivalent to separating the plane of the input vectors into two parts by using a line (or a plane) going through the origin of the coordinate system. To eliminate the need for the separating line (or plane) to go through the origin, perceptrons employ an additional input unit, which is called the *bias unit* and has always a value of 1. The weights associated with this unit, b_1, \dots, b_s , are the biases of the network (the thick lines in Figure 5.3). Assuming the input vector is represented by \mathbf{p} and the weights between the output unit i and all input units are represented by $\mathbf{w}_i = \{w_{i1}, \dots, w_{iR}\}$, the net input to the output neuron i is equivalent to $\mathbf{w}_i \cdot \mathbf{p} + b_i$. The output of the network is:

$$a_i = \begin{cases} 1 & \text{if } \mathbf{w}_i \cdot \mathbf{p} + b_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

The basic idea behind perceptron learning is to adjust the weights on the connections between the input and output layers, and learn the correct output based on the provided training data. Adjustment of weights is achieved by feeding the network with a set of training vectors, and comparing the results against a ground truth. If the network output is different from the truth, the weights are adjusted accordingly to produce outputs closer to the truth. The weights are updated by adding the difference between the truth and the network output to the old weight. This training procedure, called *the perceptron learning rule*, is repeated until the network can classify all training instances correctly or it can no longer improve the performance. Formally, the perceptron learning rule is as follows:

$$w_{ij} = w_{ij} + (t_i - a_i) \cdot p_j$$

$$b_i = b_i + (t_i - a_i)$$

where t_i is the correct output and a_i is the current network output.

Perceptron learning starts with a random initial guess of the weights and the bias. Then, training vectors are presented to the network one by one. For each input vector, the weights and the bias are adjusted accordingly using the perceptron learning rule above. An entire pass through all of the input training vectors is called an epoch. If an epoch produced no errors, the perceptron learning

stops successfully. Otherwise, another pass on the training vectors is performed until the network produces no errors. After learning the weights and the bias successfully, the perceptron can respond to unseen data with a 0 or 1 value, i.e., generalize from the input vectors in the training data.

5.3.3 Using Perceptrons for Alignment Combination

In this section, the combination function is assumed to be a weighted sum of the features and the model parameters associated with them. Formally, for each alignment link (i, j) :

$$g(\lambda_1^M(i, j), h_1^M(i, j)) = \sum_{m=1}^{m=M} \lambda_{m,lc(i,j)} \cdot h_m(i, j)$$

An alignment link is included in the final matrix if $g(\lambda_1^M(i, j), h_1^M(i, j)) \geq \phi_{lc(i,j)}$.

For a given set of features, the crucial step is to compute the parameters associated with the features, and the confidence threshold. To learn these parameters, this chapter introduces an application of single-layered perceptrons to alignment combination problem.⁷

To compute the parameters of Multi-Align, a different perceptron is learned for each link class. Each perceptron includes M input units, and 1 output unit. During perceptron learning for the link class c , each feature h_m corresponds to an input unit, and the model parameter λ_m corresponds to the weight between the

⁷Later, in Chapter 6, another learning method based on multi-layer perceptrons will be presented.

input unit p_m and the output unit. The bias b of the perceptron corresponds to $-\phi_c$. The net input to the output neuron is equal to $g(\lambda_1^M(i, j), h_1^M(i, j)) - \phi_c$ for a given alignment link (i, j) . If the net input for a given input vector is greater than 0, the perceptron outputs a 1, i.e., the alignment link (i, j) is included in the final alignment matrix.

Perceptron learning requires a ground truth, i.e., the true alignment T , to learn the weights between the input layer and the output layer. A common practice is to use data that is manually annotated by bilingual speakers as the true alignment. If an alignment link (i, j) is an element of a manual alignment T , then the correct output for that instance is 1, and 0 otherwise.

5.4 Evaluation Data and Settings

The combination technique described above was evaluated using 5-fold cross validation on an English-Chinese data set, which consists of 491 English-Chinese sentence pairs (nearly 13K words on each side) from the 2002 NIST MT evaluation test set. The details of the data set and k -fold validation were given in Section 4.4.

Evaluation of the alignments is based on precision, recall and alignment error rate on the entire set of sentence pairs for each data set as in Section 4.4. (See Section 2.6 for a description of the evaluation metrics.)

The input alignments were the same five initial alignments that were used in Section 4.4.

The following features were used to obtain link classes and to combine alignments:

1. $posE_i$: POS tag for e_i .
2. $posF_j$: POS tag for f_j .
3. $relE_i$: Dependency relation for e_i .
4. $align_{ij}$: Whether the link (i, j) is aligned or not (one for each input alignment).
5. $fertE_i$: Fertility of e_i (one for each input alignment).
6. $fertF_j$: Fertility of f_j (one for each input alignment).
7. $NC(i, j)$: Total number of existing links in $N(i, j)$ (one for each input alignment).

Link classes were obtained using linguistic features (1–3 above). As input to the combination module, alignment-based features (4–7 above) were used.

Perceptrons are sensitive to the initial weights. To reduce the effect of initialization, 11 runs of learning were performed for each training set. The final output for each training is obtained by a majority voting over 11 runs.

5.5 Experimental Results

This section presents various experiments to test the effects of feature selection for dividing links into classes, the effects of different feature functions used in combination, and the behavior of the Multi-Align approach when the initial alignments are obtained using more data.

5.5.1 Effects of Feature Selection for Link Classes

The linear combination technique above was evaluated using different features to obtain link classes on English-Chinese data using 2 input alignments. Table 5.1 summarizes the precision, recall, and alignment error rate for different features for creating different link classes. For this experiment, the only feature functions that are used in combination step are the outputs of input alignment systems. For ease of comparison, the precision, recall and AER values for five GIZA++ alignments are also included in Table 5.1.

Using only one POS tag or only the dependency relation for e_i yields alignments with an alignment error rate of 28.5–31.0%. Using only the POS tag for e_i gives better alignments than GIZA++(int), GIZA++(union), and GIZA++(gdf) but the difference is not big. For the other two features that are used to obtain link classes, the alignments are worse than GIZA++(gdf), but the difference is small.

Using POS tags on both sides gives significantly better results than the other 3 combination methods. Using $posE_i$ and $posF_j$, an alignment error rate of 25.0%

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow c$)	70.4	68.3	30.7
GIZA++($c \rightarrow e$)	66.0	69.8	32.2
GIZA++(int)	94.8	53.6	31.2
GIZA++(union)	58.3	84.5	31.6
GIZA++(gdf)	61.9	82.6	29.7

Multi-Align($\{\text{GIZA++}(e \rightarrow c), \text{GIZA++}(c \rightarrow e)\}$)				
Features for Link Classes	Features for Combination	Pr	Rc	AER
None	$align_{ij}$	61.7	78.2	31.4
$posE_i$	$align_{ij}$	67.5	76.5	28.5
$posF_j$	$align_{ij}$	62.7	80.2	30.0
$relE_i$	$align_{ij}$	62.3	78.2	31.0
$posE_i, posF_j$	$align_{ij}$	72.0	78.6	25.0
$posE_i, relE_i$	$align_{ij}$	71.7	73.2	27.6
$posE_i, relE_i, posF_j$	$align_{ij}$	73.1	75.1	26.0

Table 5.1: Multi-Align: Effects of Feature Selection for Link Classes (on English-Chinese)

is achieved—a significant relative reduction of 18.2% in AER over GIZA++(gdf).

Alignments were also evaluated where all 3 features or only $posE_i$ and $relE_i$ were used to obtain link classes but both methods yield alignments worse than using only $posE_i$ and $posF_j$.

As noted above, only the outputs of input alignment systems were used as feature functions during the combination. This is equivalent to choosing the output of one aligner in cases where the outputs of input aligners differ since the result is correct most of the time when two aligners agree on their decision on a particular link. Dividing links into classes allows for different generalizations to be made for each link class (based on POS or dependency relations).

5.5.2 Effects of Different Feature Functions

The linear combination technique was evaluated on English-Chinese data using different feature functions for combining 2 input alignments. Table 5.2 summarizes the precision, recall, and alignment error rate for different feature functions. For this experiment, based on the results obtained in the previous section, the links were divided into different classes using POS tags for both languages.

Features for Link Classes	Features for Combination	Pr	Rc	AER
$posE_i, posF_j$	$align_{ij}$	72.0	78.6	25.0
$posE_i, posF_j$	$align_{ij}, NC(i, j)$	76.2	68.8	27.6
$posE_i, posF_j$	$align_{ij}, fertE_i, fertF_j$	81.4	68.2	25.6
$posE_i, posF_j$	$align_{ij}, fertE_i, fertF_j, NC(i, j)$	80.3	69.4	25.3

Table 5.2: Multi-Align: Effects of Different Feature Functions (on English-Chinese)

Surprisingly, using additional feature functions hurts the performance when compared against the use of only alignment system outputs as feature functions. Using the number of existing links in the neighborhood hurts the performance significantly, while using fertilities and neighbor counts together gives slightly higher alignment error rate. The best AER—25.0%—is obtained when $posE_i$ and $posF_j$ are used to obtain link classes, and the alignment-system outputs as the feature functions.

On the other hand, using additional feature functions improves precision significantly at the expense of reduced recall. For instance, when $align_{ij}$, $fertE_i$, $fertF_j$, and $NC(i, j)$ are used as feature functions, a relative improvement in precision of 10.3% is obtained over using only $align_{ij}$ but at the expense of a much

lower recall (69.4% vs. 78.6%). These results indicate that the user should choose the feature functions carefully based on the application where the alignments will be utilized.

5.5.3 Effects of Training Size for Input Aligners

This section investigates whether there is room for improvement when GIZA++ is trained on more training data. For this experiment, a training set of 241K sentence pairs from FBIS corpus (nearly 9.2M English and 7.3M Chinese words) was used for training GIZA++. As in earlier GIZA++ experiments, GIZA++ involved 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4. The average sentence length for this training data is 38 words for English and 30 words for Chinese.

Table 5.3 summarizes the precision, recall and AER scores for each of the 5 initial alignments used in the earlier experiments. For each of the 5 initial alignments, all scores except the precision for GIZA++(int) went up nearly 2% (absolute increase). As before, the best precision (93.0%) is obtained by GIZA++(int), the best recall (86.3%) is obtained by GIZA++(union), and the lowest AER (27.7%) is obtained by GIZA++(gdf).

When $posE_i$ and $posF_j$ were used to obtain link classes and $align_{ij}$ was used for the feature functions, a 24.0% AER was obtained—a significant relative reduction of 13.4% over GIZA++(gdf).

The best case, which involved the use of $posE_i$ and $posF_j$ to obtain link

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow c$)	72.2	70.5	28.6
GIZA++($c \rightarrow e$)	67.5	71.6	30.6
GIZA++(int)	93.0	55.9	29.9
GIZA++(union)	60.6	86.3	29.5
GIZA++(gdf)	64.0	84.4	27.7

Multi-Align($\{\text{GIZA++}(e \rightarrow c), \text{GIZA++}(c \rightarrow e)\}$)				
Features for Link Classes	Features for Combination	Pr	Rc	AER
$posE_i, posF_j$	$align_{ij}$	75.6	76.5	24.0
$posE_i, posF_j$	$align_{ij}, NC(i, j)$	77.3	71.6	25.6
$posE_i, posF_j$	$align_{ij}, fertE_i, fertF_j$	84.4	71.0	22.7
$posE_i, posF_j$	$align_{ij}, fertE_i, fertF_j, NC(i, j)$	77.3	72.6	25.0

Table 5.3: Multi-Align: Effects of Using More Training Data for Initial Alignments (on English-Chinese)

classes and $align_{ij}$, $fertE_i$, and $fertF_j$ as feature functions, resulted in 84.4% precision, 71.0% recall and 22.7% AER. When compared to GIZA++(gdf), this is equivalent to a relative increase of 31.9% in precision, a relative reduction of 15.9% in recall, and a relative reduction of 18.0% in AER.

Regardless of the features used for link class selection and the combination step, the linear combination method yields similar relative improvements when the initial alignments are obtained using more training data.

5.6 Summary

Classifier ensembles have been shown to produce better results than the members of the ensembles in many applications. They also have been used in other NLP applications but not in word alignment. This chapter presented a new framework,

Multi-Align, to combine outputs of multiple alignment systems for improving word alignment based on the concept of classifier ensembles. In the proposed framework, each word alignment system is treated as a black box, where their outputs, along with some form of linguistic knowledge about the corpus, are fed into the Multi-Align framework. These inputs are transformed into a set of feature functions, each of which is associated with a model parameter.

Multi-Align yields better alignments than the input alignment systems by choosing the right feature functions and optimizing the model parameters. A preliminary study showed that learning model parameters via perceptrons and using a weighted summation for combining feature functions produced significantly better alignments when compared to 3 well-known combination algorithms.

Multi-Align is a general framework in that it can be tuned easily according to the applications where the alignments will be utilized. Users are able to tune the parameters accordingly to control the effects of input alignment systems or additional resources.

Chapter 6

NeurAlign

This chapter presents a new instantiation of the Multi-Align framework called *NeurAlign* that combines input alignment systems using multi-layer perceptrons rather than single-layer perceptrons. As in Multi-Align, NeurAlign treats individual alignment systems as black boxes and merges their outputs in the same classifier-ensemble spirit that was described in Chapter 5, i.e., individual alignments are transformed into a set of feature functions and an additional model is learned to assign weights to the feature functions.

Although the previous chapter demonstrated the usefulness of single-layer perceptrons for learning how to assign weights to different feature functions, there are several limitations of single-layer perceptrons:

1. Perceptrons can solve only linearly separable problems. If the set of input vectors can be separated into their correct categories by a straight line (or plane in case of multiple output units), then the input vectors are said to be linearly separable and perceptrons are guaranteed to solve the problem

in finite time. Otherwise, perceptrons cannot learn the correct classification regardless of the number of iterations.¹

2. An outlier input vector, i.e., an input vector much larger or much smaller than other vectors, slows the convergence process.

It has been shown that multi-layer perceptrons are capable of handling these two limitations successfully (Rosenblatt, 1958). Neural nets with two or more layers and non-linear activation functions are capable of learning any function of the feature space with arbitrarily small error. Neural nets have been shown to be effective particularly with:

1. High-dimensional input vectors,
2. Relatively sparse data, and
3. Noisy data with high within-class variability.

The alignment combination problem carries all these characteristics. Thus, neural networks are a good choice to learn an additional model to combine alignments.²

¹Boolean exclusive-or problem is the most famous linearly non-separable problem which perceptrons are unable to solve.

²Neural nets is a suitable choice for learning an additional model to combine alignments. However, it is worth noting that there are other equally good machine learning techniques, such as support-vector machines, that can be used to instantiate Multi-Align. The goal in this chapter is to demonstrate the effectiveness of Multi-Align framework using one of many possible choices.

This chapter introduces a new technique, *NeurAlign*, that learns how to combine individual word alignment systems using multi-layer perceptrons. Using the Multi-Align framework, the alignment combination problem is transformed into a classifier ensemble problem. Employing different feature functions and learning model parameters using neural networks, *NeurAlign* yields significant improvements over the input alignments and current-best alignment combination techniques.

The rest of this chapter gives some background on neural networks and describes how *NeurAlign* fits into the Multi-Align framework. In addition, the methodology for combining multiple alignments using neural networks is described. Finally, various experiments are presented that demonstrate the effectiveness of *NeurAlign* on different language pairs (with or without any resources on the FL side) using different input alignments and varying sizes of manually-annotated data.

6.1 Neural Networks

A multi-layer perceptron (MLP) is a feed-forward neural network that consists of several units (neurons) that are connected to each other by weighted links. As illustrated in Figure 6.1, an MLP consists of one input layer, one or more hidden layers, and one output layer. The external input is presented to the input layer and propagated forward through the hidden layers. Ultimately, an output vector

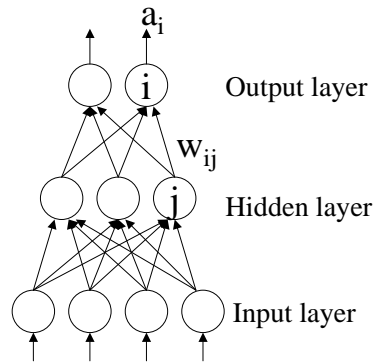


Figure 6.1: Multilayer Perceptron Overview

is created in the output layer. Each unit i in the network computes its output with respect to its net input $net_i = \sum_j w_{ij}a_j$, where j represents all units in the previous layer that are connected to the unit i . The output of unit i is computed by passing the net input through a non-linear activation function f , i.e. $a_i = f(net_i)$.

What makes neural nets more powerful than plain perceptrons is the non-linearity (i.e, the capability to represent nonlinear functions) introduced into the network by the activation functions. Without nonlinearity, hidden units would not make nets more powerful than just plain perceptrons (which do not have any hidden units, just input and output units). The reason is that a composition of linear functions is again a linear function. It is the nonlinearity (i.e, the capability to represent nonlinear functions) that makes multilayer networks so powerful. Almost any nonlinear function may be used, although for backpropagation learning it must be differentiable and it helps if the function is bounded. The most commonly used non-linear activation functions are the log sigmoid function $f(x) = \frac{1}{1+e^{-x}}$, which ‘squeezes’ its input to the range of $[0,1]$, or hyperbolic tangent sigmoid function

$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$, which ‘squeezes’ its input into the range of $[-1,1]$. The latter has been shown to be more suitable for binary classification problems.

The learning algorithm is based on a gradient descent in error space, where the error is defined as $E = \sum_p E_p$, for an input pattern p :

$$E_p = \frac{1}{2} \sum_i (t_i - a_i)^2$$

where t_i is the true output and a_i is the output of neural nets for the output unit i . The weights of the units are changed according to the gradient of the error:

$$\Delta w = -\eta \nabla E$$

where η is a constant scaling factor (i.e., learning rate). The backpropagation algorithm successively computes ∇E by propagating the error from the output layer toward the input layer. The basic idea is to compute the partial derivatives $\frac{\partial E}{\partial w_{ij}}$ by repeatedly applying the chain rule. The individual weight change components, i.e., the weight change for the connection from unit j to unit i , is defined as:

$$\Delta w_{ij} = -\eta \nabla_{ij} E = -\eta \frac{\partial E}{\partial w_{ij}}$$

The error gradient can be divided into three components:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_i} \frac{\partial a_i}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}}$$

After a series of computing partial derivatives, the weight update function can be written as follows:

$$\Delta w_{ij} = \eta \delta_i a_j$$

where

$$\delta_i = \begin{cases} (t_i - a_i)f'_i(net_i) & \text{if } i \text{ is an output unit} \\ f'_i(net_i) \sum_k \delta_k w_{ik} & \text{otherwise} \end{cases}$$

The critical question is the computation of weights associated with the links connecting the neurons. The typical technique for such a computation is the back-propagation algorithm, which uses the principle of gradient descent (Rumelhart et al., 1986). The weights can be updated either after examining each pattern and computing the gradient (online learning) or only once after examining all patterns (batch learning). Two major problems with the gradient descent method are: (1) it is very difficult to choose the learning step appropriately; (2) updating the weights can be costly, depending on the choice of the non-linear activation function f . Moreover, the change in weight is dependent on the size of the partial derivative $\frac{\partial E}{\partial w_{ij}}$, which might be an important issue when sigmoid activation functions are utilized. There have been various algorithms for handling these shortcomings of the gradient descent method.³

The approach used in this thesis uses the resilient backpropagation (RPROP) algorithm (Riedmiller and Braun, 1993), which is based on the gradient descent method, but converges faster and generalizes better. The motivation behind the resilient backpropagation (RPROP) training algorithm is that it eliminates the harmful effects of the magnitudes of the partial derivatives. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the

³See (Riedmiller, 1994) for a review of these methods.

derivative has no effect on the weight update. The size of the weight change is determined by a separate update value. The update value for each weight and bias is increased by a factor δ_{inc} whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations. The update value is decreased by a factor δ_{dec} whenever the derivative with respect that weight changes sign from the previous iteration. If the derivative is zero, then the update value remains the same. Whenever the weights are oscillating the weight change will be reduced. If the weight continues to change in the same direction for several iterations, then the magnitude of the weight change will be increased. A complete description of the RPROP algorithm is given in (Riedmiller and Braun, 1993).

6.2 NeurAlign in Multi-Align Framework

Multi-Align uses five parameters for determining whether two words e_i and f_j should be aligned:

1. $lc(i, j)$: Function for dividing links into classes,
2. $h_1^M(i, j)$: Set of feature functions,
3. $\lambda_1^M(i, j)$: Parameters associated with the feature functions,
4. g : Combination function, and
5. $\phi_{lc(i,j)}$: Confidence threshold for each link class.

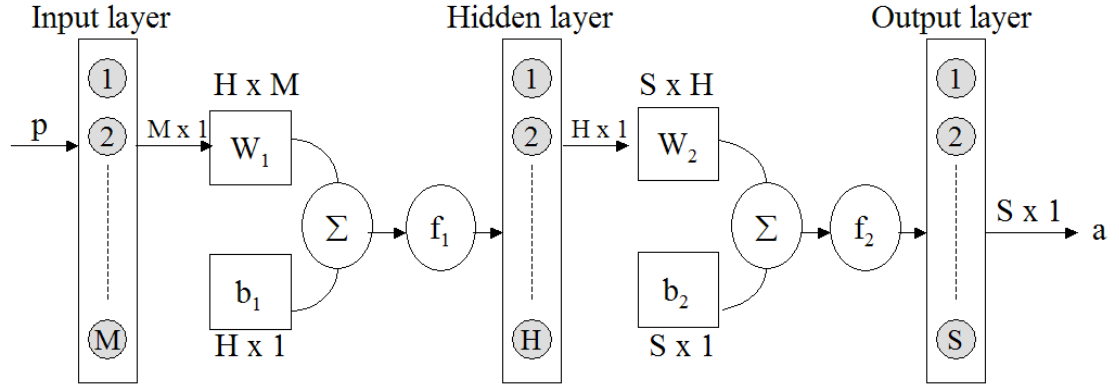


Figure 6.2: NeurAlign in Multi-Align Framework

The set of feature functions will be described in detail in Section 6.3. For now, it is assumed that each alignment link is represented by M feature functions. NeurAlign uses a two-layered perceptron to combine alignments, as depicted in Figure 6.2. Assuming that the set feature functions for a particular alignment link is represented by p , which is a $M \times 1$ matrix, the output of the neural network in Figure 6.2 is equivalent to:

$$a = f_2(w_2 \times f_1(w_1 \times p + b_1) + b_2)$$

The size of the matrix w_1 is $H \times M$ and the size of the matrix w_2 is $S \times H$, where H is the number of hidden nodes in the hidden layer, and S is the number of output units. The matrices b_1 and b_2 correspond to the matrices for the bias values. f_1 and f_2 correspond to the non-linear transfer functions. In the experiments described below, the hyperbolic tangent sigmoid function $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ is used for f_1 and f_2 .

For the alignment combination problem, the output layer consists of only one node indicating whether an alignment link exists or not between two words. Given this, the combination function g in the Multi-Align framework is equivalent to output of the neural network in Figure 6.2, i.e., $g(p) = a$. When there exists only one hidden node in the hidden layer, i.e. $H = 1$, w_1 corresponds to the vector of model parameters λ_1^M associated with M feature functions, i.e., $w_{11} = \lambda_1, w_{12} = \lambda_2$, and so on. If $H > 0$, each column of the matrix w_1 corresponds to a particular model parameter λ_m , i.e., each feature function is represented by H model parameters. An alignment link is included in the final matrix if $g(p) > 0$, i.e., the confidence threshold is set to 0 for each link class ($\phi_{lc(i,j)} = 0$).

NeurAlign has two variations: NeurAlign₁, which uses all training data at once, and NeurAlign₂, which divides links into classes and learn different model parameters for each link class. In the first case, function $lc(i, j)$ is constant. For NeurAlign₂, function $lc(i, j)$ is conditioned on the set of features that are used to divide the links into classes. In experiments in Section 6.4, POS tags associated with the words will be used to divide links into classes.

6.3 NeurAlign Approach

Let A_k be an alignment between sentences \mathbf{e} and \mathbf{f} , where each element $a \in A_k$ is an alignment link (i, j) . Let $\mathcal{A} = \{A_1, \dots, A_l\}$ be a set of alignments between \mathbf{e}

and \mathbf{f} . The true alignment is T , where each $a \in T$ is of the form (i, j) .⁴

The goal is to combine the information in A_1, \dots, A_l such that the resulting alignment is closer to T . A straightforward solution is to take the intersection or union of the individual alignments, or to perform a majority voting for each possible alignment link (i, j) . Here, an additional model is used to learn how to combine outputs of A_1, \dots, A_l .

The task of combining word alignments is decomposed into two steps: (1) Extracting features; and (2) Learning a classifier from the transformed data. Each of these two steps is described below in turn.

6.3.1 Extracting Features

Given sentences \mathbf{e} and \mathbf{f} , a (potential) alignment instance (i, j) is created for all possible word combinations. A crucial component of building a classifier is the selection of features to represent the data. The simplest approach is to treat each alignment-system output as a separate feature upon which to build a classifier. However, when only a few alignment systems are combined, this feature space is not sufficient to distinguish between instances. A common classification strategy is to supply the input data to the set of features as well.

Two types of features are used to describe each instance (i, j) in the alignment combination: (1) linguistic features and (2) alignment features. Linguistic features include POS tags of both words (e_i and f_j) and a dependency relation for one of

⁴See Section 4.2 for the notation for alignment links and related predicates.

the words (e_i).

Alignment features consist of features that are extracted from the outputs of individual alignment systems. For each alignment $A_k \in \mathcal{A}$, the following are some of the alignment features that can be used to describe an instance (i, j) :

1. Whether (i, j) is an element of A_k or not
2. Translation probability $p(f_j|e_i)$ computed over A_k ⁵
3. Fertility of (i.e., number of words in \mathbf{f} that are aligned to) e_i in A_k
4. Fertility of (i.e., number of words in \mathbf{e} that are aligned to) f_j in A_k
5. For each neighbor $(x, y) \in N(i, j)$, whether $(x, y) \in A_k$ or not (8 features in total)
6. For each neighbor $(x, y) \in N(i, j)$, translation probability $p(f_y|e_x)$ computed over A_k (8 features in total)

It is also possible to use variants (or combinations) of these features to reduce the feature space.

Figure 6.3 shows an example of how the outputs of 2 alignment systems, A_1 and A_2 , are transformed for an alignment link (i, j) into data with some of the features above. The numbers -1 and 1 are used to represent the absence and

⁵The translation probabilities can be borrowed from the existing systems, if available. Otherwise, they can be generated from the outputs of individual alignment systems using likelihood estimates.

		f_{j-1}	f_j	f_{j+1}	Features for the alignment link (i, j)	
A_1	e_{i-1}				pos(e_i) , pos(f_j)	Noun, Prep
	e_i	X	X		rel(e_i)	Modifier
	e_{i+1}			X	outputs of aligners	1 (for A_1), -1 (for A_2)
A_2		f_{j-1}	f_j	f_{j+1}	neighbors (A_1)	-1, -1, -1, 1 , -1, -1, -1, 1
	e_{i-1}	X			neighbors (A_2)	1 , -1, -1, -1, 1 , -1, -1, 1
	e_i			X	neighbors ($A_1 \cup A_2$)	1 , -1, -1, 1 , 1 , -1, -1, 1
	e_{i+1}			X	total neighbors	2 (for A_1), 3 (for A_2)
					fertility(e_i)	2 (for A_1), 1 (for A_2)
					fertility(f_j)	1 (for A_1), 0 (for A_2)

Figure 6.3: An Example of Transforming Alignments into Classification Data

existence of a link, respectively. The neighboring links are presented in row-by-row order.

For each sentence pair $\mathbf{e} = e_1, \dots, e_I$ (of length I) and $\mathbf{f} = f_1, \dots, f_J$ (of length J) in the classification data, a set of $I \times J$ instances is generated, one for each alignment system A_i

Supervised learning requires the correct output, i.e., the true alignment T . If an alignment link (i, j) is an element of T , the correct output is set to 1, and to -1 otherwise.

6.3.2 Learning A Classifier

Once the alignments are transformed into a set of instances with several features, the remaining task is to learn a classifier from these data. In the case of word alignment combination, there are important issues to consider for choosing an

GIZA++		Manual		
$e \rightarrow s$	$s \rightarrow e$	-1	1	Total
-1	-1	150,797	1,546	152,343
-1	1	497	491	988
1	-1	545	456	1,001
1	1	69	3,672	3,741
Total		151,908	6,165	158,073

Table 6.1: Distribution of Instances According to GIZA++ Outputs (on English-Spanish)

appropriate classifier. First, there is a very limited amount of manually annotated data. This may give rise to poor generalizations because it is very likely that new tests will include lots of cases that are not observed in the training data.

Second, the distribution of the data according to the classes is skewed. A preliminary analysis on 199 English-Spanish sentences from a mixed corpus (UN + Bible + FBIS) was conducted to compare the outputs of two alignment systems (GIZA++ alignments in either direction (Och and Ney, 2000b)) to a set of manually annotated data. As shown in Table 6.1, only 6K (4%) of nearly 158K instances are assigned to a class of 1. Moreover, only 60% of those 6K instances that are assigned to class 1 are also assigned to class 1 by the individual alignment systems.

Finally, given the distribution of the data, it is difficult to find the right features to distinguish between instances. Thus, it is prudent to use as many features as possible and let the learning algorithm filter out the redundant features.

The next two sections describe how neural nets are used at different levels to build a good classifier.

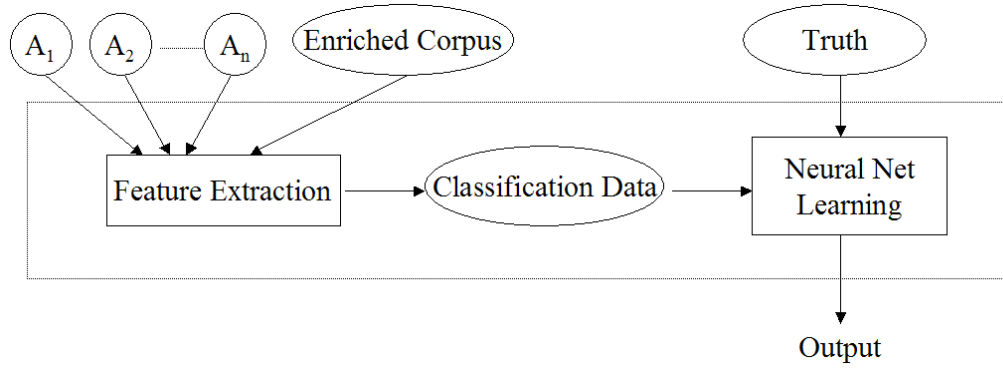


Figure 6.4: NeurAlign₁—Alignment Combination Using All Data At Once

6.3.2.1 NeurAlign₁: Learning All At Once

Figure 6.4 illustrates how alignments are combined using all the training data at the same time (NeurAlign₁). First, the outputs of individual alignments systems and the original corpus (enriched with additional linguistic features) are passed to the feature extraction module. This module transforms the alignment problem into a classification problem by generating a training instance for every pair of words between the sentences in the original corpus. Each instance is represented by a set of features (described in Section 6.3.1). The new training data is passed to a neural net learner, which outputs whether an alignment link exists for each training instance.

6.3.2.2 NeurAlign₂: Multiple Neural Networks

The use of multiple neural networks (NeurAlign₂) enables the decomposition of a complex problem into smaller problems. *Local experts* are learned for each smaller

		SPANISH						
		Adj	Adv	Comp	Det	Noun	Prep	Verb
E	Adj	18	-	-	82	40	96	66
N	Adv	-	8	-	-	50	67	75
G	Comp	-	-	12	-	46	37	96
L	Det	-	-	-	10	60	100	-
I	Noun	42	77	100	94	23	98	84
S	Prep	-	-	-	93	70	22	100
H	Verb	42	-	-	100	66	78	43

Table 6.2: Error Rates According to POS Tags for GIZA++($e \rightarrow s$) (on English-Spanish)

problem and these are then merged. Following Tumer and Ghosh (Tumer and Ghosh, 1996a), *NeurAlign₂* is designed to partition training instances spatially using proximity of patterns in the input space to reduce the complexity of the tasks assigned to individual classifiers.

A preliminary analysis was conducted for the purpose of determining how to divide training data into subsets for neural-network training. 100 English-Spanish sentence pairs were randomly selected from a mixed corpus (UN + Bible + FBIS) to observe the distribution of errors according to POS tags in both languages. The cases where the individual alignment and the manual annotation were different—a total of 3,348 instances—were examined. Of these, 1,320 were misclassified by GIZA++ ($e \rightarrow s$).⁶ A standard measure of error, i.e., the percentage of misclassified instances out of the total number of instances, is used for this analysis.

Table 6.2 shows error rates (by percentage) for GIZA++($e \rightarrow s$) according

⁶This analysis ignored the cases where both systems produced an output of -1 (i.e., the words are not aligned).

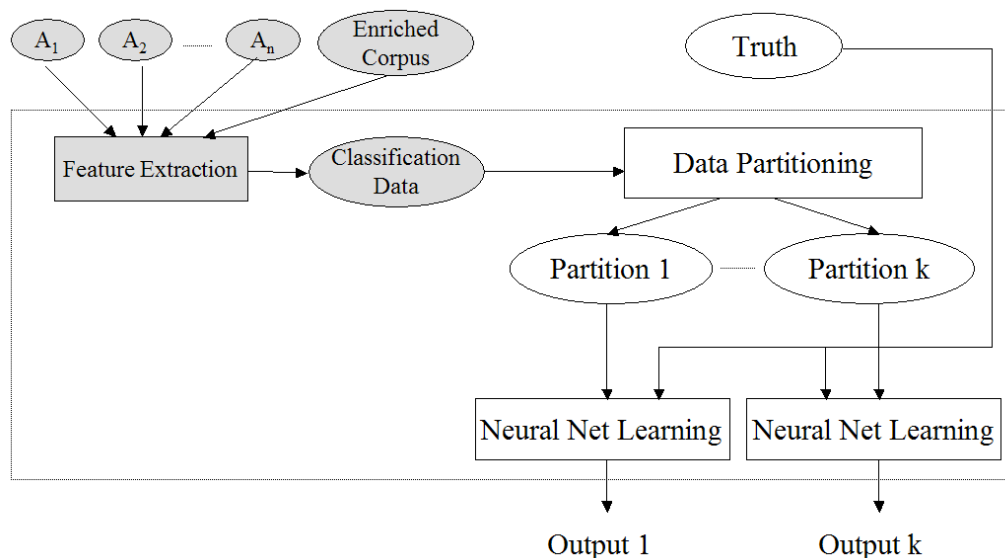


Figure 6.5: NeurAlign₂—Alignment Combination with Partitioning

to POS tags.⁷ The error rate is relatively low in cases where both words have the same POS tag. Except for verbs, the lowest error rate is obtained when both words have the same POS tag (the error rates on the diagonal). On the other hand, the error rates are high in several other cases—as high as 100%, e.g., when the Spanish word is a determiner or a preposition.⁸ This suggests that dividing the training data according to POS tag, and training neural networks on each subset separately might be better than training on the entire data at once.

Figure 6.5 illustrates the combination approach with neural nets after partitioning the data into disjoint subsets (NeurAlign₂). Similar to NeurAlign₁, the outputs of individual alignment systems, as well as the original corpus, are passed

⁷Only POS pairs that occurred at least 10 times are shown.

⁸The same analysis was done for the other direction and resulted in similar distribution of error rates.

to the feature extraction module. Then the training data is split into disjoint subsets using a subset of the available features for partitioning. Different neural nets are learned for each partition, and then the outputs of the individual nets are merged. The advantage of this is that it results in different generalizations for each partition and that it uses different subsets of the feature space for each net.

6.4 Evaluation

This section presents the data that is used to train neural networks and evaluate NeurAlign, the settings for neural networks, and description of the features that are employed in NeurAlign.

6.4.1 Training and Evaluation Data

NeurAlign was evaluated using 5-fold cross validation on four different data sets:

1. A set of 199 English-Spanish sentence pairs (nearly 5K words on each side) from a mixed corpus (UN + Bible + FBIS),
2. A set of 491 English-Chinese sentence pairs (nearly 13K words on each side) from 2002 NIST MT evaluation test set,
3. A set of 450 English-Arabic sentence pairs (11K words in Arabic and 13K words in English) from 2003 NIST MT evaluation test set (Ittycheriah and Roukos, 2005), and

4. A set of 248 English-Romanian sentence pairs (nearly 5.5K words on each side) from HLT’2003 Word Alignment Workshop (Mihalcea and Pedersen, 2003).

The details about the English-Spanish and English-Chinese data sets can be found in Section 4.4. For the other two data sets, the manual annotation was done by a bilingual speaker. Every link in the gold standard is considered a sure alignment link (i.e., $P = S$) during evaluation.

The experiments with NeurAlign used two existing word-alignment systems to generate initial alignments:

1. A statistical alignment system based on IBM models, i.e., GIZA++ (Och, 2000), and
2. A Syntax-Aware alignment system based on HMMs, i.e., SAHMM (Lopez and Resnik, 2005).

The details of how the English-Spanish and English-Chinese alignments were generated were provided in Section 4.4. An additional 44K sentence pairs (nearly 1.4M English and 1M Arabic words) were used for English-Arabic, and an additional 48K sentence pairs (nearly 1M words on each side) were used for English-Romanian to train GIZA++. For both data sets, GIZA++ was trained using 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4.

Three alignment combination heuristics were evaluated for the NeurAlign experiments:

1. Intersection of both directions (which will be represented as *Aligner*(int),
2. The union of both directions (which will be represented as *Aligner*(union)),
3. A heuristic combination approach called *grow-diag-final* (Koehn et al., 2003) (which will be represented as *Aligner*(gdf)).

For ease of comparison, Tables 6.3 and 6.4 summarize the precision, recall and alignment error rate values (in percentages) for all data sets.⁹ The best results for each metric on each data set are highlighted in boldface. For both systems, using the intersection of both directions yields the highest precision. The highest recall, on the other hand, is obtained by the union of both directions. For GIZA++, the lowest alignment error rate is obtained by the heuristic method grow-diag-final for Spanish, Chinese and Romanian while GIZA++($a \rightarrow e$) yields the lowest AER on Arabic data set. For SAHMM, the lowest alignment error rate is obtained by SAHMM($e \rightarrow c$) on English-Chinese data set.

The evaluations were performed using k -fold cross validation and 3 evaluation metrics, as described in previous chapters.

⁹SAHMM alignments were available for only English-Chinese. Therefore, other language pairs were evaluated using only GIZA++ alignments. NeurAlign is able to handle the outputs of other alignment system outputs easily once they are available.

Alignments	English-Spanish			English-Chinese		
	Pr	Rc	AER	Pr	Rc	AER
GIZA++($e \rightarrow f$)	87.0	67.0	24.3	70.4	68.3	30.7
GIZA++($f \rightarrow e$)	88.0	67.5	23.6	66.0	69.8	32.2
GIZA++(int)	98.2	59.6	25.9	94.8	53.6	31.2
GIZA++(union)	80.6	74.9	22.3	58.3	84.5	31.6
GIZA++(gdf)	83.8	74.4	21.2	61.9	82.6	29.7

Alignments	English-Arabic			English-Romanian		
	Pr	Rc	AER	Pr	Rc	AER
GIZA++($e \rightarrow f$)	66.4	64.7	34.5	72.9	62.4	32.7
GIZA++($f \rightarrow e$)	68.1	76.5	27.9	74.9	65.7	30.0
GIZA++(int)	96.1	57.1	28.4	94.2	52.4	32.7
GIZA++(union)	56.0	84.1	32.8	64.3	75.7	30.5
GIZA++(gdf)	60.2	83.0	30.2	68.0	74.6	28.8

Table 6.3: GIZA++ Results (on English-Spanish, English-Chinese, English-Arabic and English-Romanian)

Alignments	English-Chinese		
	Pr	Rc	AER
SAHMM($e \rightarrow f$)	73.2	73.9	26.5
SAHMM($f \rightarrow e$)	65.8	72.2	31.3
SAHMM(int)	93.4	58.8	27.6
SAHMM(union)	59.8	87.3	29.8
SAHMM(gdf)	62.4	85.2	28.6

Table 6.4: SAHMM Results (on English-Chinese)

6.4.2 Neural Network Settings

In the NeurAlign experiments, a multi-layer perceptron (MLP) consisting of 1 input layer, 1 hidden layer, and 1 output layer was used. The hidden layer consists of 10 units, and the output layer consists of 1 unit. All units in the hidden layer are fully connected to the units in the input layer, and the output unit is fully connected to all the units in the hidden layer. The hyperbolic tangent sigmoid function is used as the activation function between the input layer and the hidden

layer. The output unit is computed by first using the tangent sigmoid function and then mapping its output to either -1 or 1 using the sign function.

One of the potential pitfalls of the neural-net approach is the tendency for overfitting as the number of iterations increases. To address this, an *early stopping with validation set* method was used, where a portion of the training data is held out as the validation set. After each update of the weights during training, the error on the validation set is computed. In the early iterations of training, the error on the validation set decreases as the error on the training set decreases. However, when the network begins to overfit the data, the error on the validation set will typically rise. If the error on the validation set increases beyond a certain number of iterations, the training is stopped, and the weights at the minimum of the validation error are returned. In the experiments below, a randomly-selected portion (1/4 of the training set) was held out as the validation set.

Neural nets are sensitive to the initial weights. To reduce the effect of initialization, 5 runs of learning were performed for each training set. The final output for each training run was obtained by a majority voting over 5 runs.

6.4.3 Description of Classification Data

The following additional features were used, as well as the outputs of individual aligners, for an instance (i, j) (the the set of features 2–7 below were generated separately for each input alignment A_k):

1. $posE_i, posF_j, relE_i$: POS tags and dependency relation for e_i and f_j .
2. $neigh(i, j)$: 8 features indicating whether a neighboring link exists in A_k .
3. $fertE_i, fertF_j$: 2 features indicating the fertility of e_i and f_j in A_k .
4. $NC(i, j)$: Total number of existing links in $N(i, j)$ in A_k .
5. $TP(i, j)$: Translation probability $p(f_j|e_i)$ in A_k .
6. $NghTP(i, j)$: 8 features indicating the translation probability $p(f_y|e_x)$ for each $(x, y) \in N(i, j)$ in A_k .
7. $AvTP(i, j)$: Average translation probability of the neighbors of (i, j) in A_k .

POS tags were generated using the MXPOST tagger (Ratnaparkhi, 1996) for English and Chinese, and Connexor for Spanish. Dependency relations were produced using a version of the Collins parser (Collins, 1997) that has been adapted for building dependencies. To address the issue of data sparseness, the set of original POS tags and dependency relations were transformed into a more generalized set of 10 POS tags and 6 relations, as described in Section 3.3.3. The final set of POS tags consists of *Adjective*, *Adverb*, *Complementizer*, *Conjunction*, *Determiner*, *Noun*, *Particle*, *Preposition*, *Punctuation* and *Verb*. The final set of relations consists of *Modifier (Mod)*, *Direct Object (Obj)*, *objects with prepositional phrases (PObj)*, *Predicative (Pred)*, *Sentence (S)* and *Subject (Subj)*.

An important limitation of neural networks is the restriction of features to real-valued attributes. As a result, handling categorial (non-numeric) attributes is

problematic. Two standard techniques to overcome this problem are 1) mapping the values of categorial attributes into integers, 2) employing a binary feature for each possible value of a categorial attribute and setting only one of them to 1. In this thesis, in order to keep the number of features at a minimum, the values of categorial attributes are mapped to integers.¹⁰

6.5 Experimental Results

This section presents various experiments to test the usefulness of NeurAlign. Instead of presenting all possible combinations of parameters, the experiments below build on the previous experiments, i.e., they use the parameters that yielded the best performance in previous experiments. The experiments serve to illuminate the effects of different settings for NeurAlign:

1. Different feature sets for combining alignments for NeurAlign₁ (on English-Spanish),
2. Different feature sets for partitioning the training data (on English-Spanish and English-Chinese),
3. Number of input alignments in NeurAlign₂ (on English-Chinese),
4. Size of training data for NeurAlign₂ (on English-Chinese),

¹⁰In order to reduce effects of ranges of different attributes on the final output, the values of all attributes are scaled to real numbers between 0 and 1 in a preprocessing step.

5. Size of training data for generating input alignments (on English-Chinese),
6. Number of folds in k -fold cross validation (on English-Chinese), and
7. Lack of resources in one of the languages (on English-Arabic and English-Romanian).

The choice of a particular language pair in each experiment is influenced by the availability of different input alignments, the quantity of annotated data and parallel corpora, and the resources on the FL.

Statistical significance tests were performed on all values of error rates reported in this section using two-tailed paired t-tests. Unless otherwise indicated, the differences between NeurAlign and other alignment systems, as well as the differences among NeurAlign variations themselves, were statistically significant within the 95% confidence interval.

6.5.1 Effects of Using Different Features for NeurAlign₁

Table 6.5 presents the results of training neural nets using the entire data with different subsets of the feature space for the English-Spanish data set. These experiments were conducted using only GIZA++ alignments as input since SAHMM alignments were not available on English-Spanish data.

NeurAlign₁ performed worse than GIZA++(gdf) when POS tags and the dependency relation were used as features (22.5% vs. 21.2% AER). Using the neighboring links as the feature set gave slightly (but not significantly) better results

Features	Pr	Rc	AER
GIZA++(gdf)	83.8	74.4	21.2
$posE_i, posF_j, relE_i$	90.6	67.7	22.5
$neigh(i, j)$	91.3	69.5	21.1
$posE_i, posF_j, relE_i, neigh(i, j),$	91.7	70.2	20.5
$posE_i, posF_j, relE_i, fertE_i, fertF_j$	91.4	71.1	20.0
$posE_i, posF_j, relE_i, neigh(i, j),$ $NC(i, j), fertE_i, fertF_j$	89.5	76.3	17.6
$neigh(i, j), NC(i, j), fertE_i, fertF_j$	89.7	75.7	17.9
$posE_i, posF_j, relE_i, fertE_i, fertF_j,$ $neigh(i, j), NC(i, j), TP(i, j), AvTP(i, j)$	90.0	75.7	17.9

Table 6.5: NeurAlign₁: Effects of Feature Set for Combining Alignments (on English-Spanish)

than GIZA++(gdf) (21.1% vs. 21.2% AER). Using POS tags, dependency relations, and neighboring links also resulted in better performance than GIZA++(gdf) (20.5% vs. 21.2%) but the difference was not statistically significant.

When fertilities were used along with the POS tags and dependency relations, the AER was 20.0%—a significant relative error reduction of 5.7% over GIZA++(gdf). Adding the neighboring links to the previous feature set resulted in an AER of 17.6%—a significant relative error reduction of 17% over GIZA++(gdf). Interestingly, when the POS tags and dependency relations were removed from this feature set, there was no significant change in the AER, which indicates that the improvement is mainly due to the neighboring links. This supports the initial claim in this thesis about the clustering of alignment links, i.e., when there is an alignment link, usually there is another link in its neighborhood. Finally, translation probabilities were tested as a part of the feature set: these resulted in AER rates that were no better than the case where they were not used. This may be

the case because the translation probability $p(f_j|e_i)$ has a unique value for each pair of e_i and f_j ; therefore it is not useful to distinguish between alignment links with the same words.

The lowest AER was obtained by using the set of all features except for the translation probabilities. Thus, the NeurAlign experiments below use POS tags, dependency relations, neighborhood features, and fertilities as the set of features for combining alignments (i.e., the last row of Table 6.5).

6.5.2 Effects of Partitioning Data (NeurAlign₁ vs. NeurAlign₂)

In order to train on partitioned data (NeurAlign₂), an appropriate set of features needs to be established for partitioning the training data. Table 6.6 presents the evaluation results for NeurAlign₁ (i.e., no partitioning) and NeurAlign₂ with different features for partitioning (English POS tag, Spanish POS tag, and POS tags on both sides). The results for input alignments (uni-directional GIZA++ alignments) and the best combination method (GIZA++(gdf)) are listed for ease of comparison. For training on each partition, the feature space included POS tags (e.g., Spanish POS tag in the case where partitioning is based on English POS tag only), dependency relations, neighborhood features, and fertilities.

The results indicated that partitioning based on POS tags on one side reduced the AER to 17.4% and 17.1%, respectively. Using POS tags on *both* sides reduced the error rate to 16.9%—a significant relative error reduction of 5.6% over no partitioning. All four variations of NeurAlign (including no partitioning) yielded

Alignment	Pr	Rc	AER
GIZA++($e \rightarrow s$)	87.0	67.0	24.3
GIZA++($s \rightarrow e$)	88.0	67.5	23.6
GIZA++(gdf)	83.8	74.4	21.2
NeurAlign ₁	89.7	75.7	17.9
NeurAlign ₂ [$posE_i$]	91.1	75.4	17.4
NeurAlign ₂ [$posF_j$]	91.2	76.0	17.1
NeurAlign ₂ [$posE_i, posF_j$]	91.6	76.0	16.9

Table 6.6: NeurAlign₁: Effects of Feature Selection for Partitioning (on English-Spanish)

Features	Pr	Rc	AER
GIZA++($e \rightarrow s$)	87.0	67.0	24.3
GIZA++($s \rightarrow e$)	88.0	67.5	23.6
GIZA++(gdf)	83.8	74.4	21.2
$relE_i, fertE_i, fertF_j,$ $TP(i, j), AvTP(i, j), NghTP(i, j)$	91.9	73.0	18.6
$neigh(i, j)$	90.3	74.0	18.7
$relE_i, fertE_i, fertF_j, neigh(i, j), NC(i, j)$	91.6	76.0	16.9
$relE_i, fertE_i, fertF_j, neigh(i, j), NC(i, j),$ $TP(i, j), AvTP(i, j)$	91.4	76.1	16.9

Table 6.7: NeurAlign₂: Effects of Feature Set for Combining Alignments (on English-Spanish)

statistically significant error reductions over GIZA++(gdf)—a relative reduction of 20.3% in AER in the best case (16.9% vs. 21.2%). The best scores were obtained by using both POS tags for partitioning, and the rest of the features for combination. Thus, the NeurAlign₂ experiments below use partitioning based on this setting (i.e., the last row of Table 6.6).

Once it was determined that partitioning by POS tags on both sides brought about the biggest gain, NeurAlign₂ was run using this partitioning, but with different feature sets for the combination step. Table 6.7 shows the results of this experiment. Using dependency relations, word fertilities and translation probabili-

ties (both for the link in question and the neighboring links) yielded a significantly lower AER (18.6%)—a relative error reduction of 12.3% over GIZA++(gdf). Using only the neighboring links resulted in a slightly (not significantly) lower AER (18.7%) when compared to the previous feature set, which indicates that neighboring links help to find the correct alignments more than other features. When the feature set consisted of dependency relations, word fertilities, and neighboring links, the AER was reduced to 16.9%—a 20.3% relative error reduction over GIZA++(gdf). The addition of translation probabilities to this feature set was also tested, but as in the case of NeurAlign_1 , this did not improve the alignments.

In the best case, NeurAlign_2 achieved significant reductions in AER over the input alignment systems: a 28.4% relative error reduction over $\text{GIZA++}(s \rightarrow e)$ and a 30.5% relative error reduction over $\text{GIZA++}(e \rightarrow s)$. When compared to $\text{GIZA++}(\text{gdf})$, NeurAlign_2 yielded a significant relative error reduction of 20.3%.

The effects of partitioning data on language pairs other than English-Spanish data was also tested. A similar experiment was conducted for English-Chinese, using different features for partitioning. As in the case of English-Spanish experiments, 4 different settings were used for partitioning the data, and the POS tags, dependency relations, word fertilities, and neighborhood links were used as combination features.

Table 6.8 shows the results of the input alignments to NeurAlign (either GIZA++ or SAHMM alignments in two different directions), NeurAlign_1 (no partitioning) and variations of NeurAlign_2 (with different features for partitioning,

Alignments	GIZA++			SAHMM		
	Pr	Rc	AER	Pr	Rc	AER
<i>Aligner</i> ($e \rightarrow c$)	70.4	68.3	30.7	73.2	73.9	26.5
<i>Aligner</i> ($c \rightarrow e$)	66.0	69.8	32.2	65.8	72.2	31.3
<i>Aligner</i> (gdf)	61.9	82.6	29.7	62.4	85.2	28.6
NeurAlign ₁	85.0	71.4	22.2	85.6	70.9	22.2
NeurAlign ₂ [$posE_i$]	85.7	74.6	20.0	85.2	75.3	19.9
NeurAlign ₂ [$posF_j$]	85.7	73.2	20.8	84.7	74.0	20.9
NeurAlign ₂ [$posE_i, posF_j$]	86.3	74.7	19.7	85.4	77.2	18.8

Table 6.8: NeurAlign: Effects of Feature Selection for Partitioning Using GIZA++ and SAHMM Alignments (on English-Chinese)

e.g., English POS tag, Chinese POS tag, and POS tags on both sides). For comparison purposes, the results for GIZA++(gdf) and SAHMM(gdf) are also included in the table.

Using uni-directional GIZA++ alignments as input, without any partitioning, NeurAlign achieves an alignment error rate of 22.2%—a significant relative error reduction of 25.3% over GIZA++(gdf). Partitioning the data according to POS tags results in significantly better results over no partitioning. When the data is partitioned according to POS tags on only one side, NeurAlign achieves 20.0% and 20.8% AER, respectively. When the data is partitioned according to both POS tags, NeurAlign reduces AER to 19.7%—a significant relative error reduction of 33.7% over GIZA++(gdf) (19.7% vs. 29.7%). Compared to the input alignments, the best version of NeurAlign, i.e., using POS tags on both sides for partitioning data, achieves a relative error reduction of 35.8% and 38.8%, respectively.

Similarly, using uni-directional SAHMM alignments as input (without any partitioning), NeurAlign achieves an alignment error rate of 22.2%—a significant

relative error reduction of 22.4% over SAHMM(gdf). As in the case of GIZA++ alignments as input, partitioning the data according to POS tags results in significantly better results over no partitioning. When the data are partitioned according to POS tags on only one side, NeurAlign achieves 19.9% and 20.9% AER, respectively. When the data are partitioned according to both POS tags, NeurAlign reduces AER to 18.8%—a significant relative error reduction of 34.2% over SAHMM(gdf) (18.8% vs. 28.6%).

For the remaining Chinese experiments, the best settings from this section were used, i.e., using POS tags on both sides for partitioning data, and all other features for combining alignments.

6.5.3 Effects of Number of Alignment Systems

One of the important research questions is how much NeurAlign can improve the word alignments as the number of input alignments increases. For this purpose, an experiment with 4 input alignments on English-Chinese data was conducted: GIZA++ in both directions and SAHMM in both directions.¹¹ Table 6.9 presents the results for these four input alignments, two combinations of uni-directional GIZA++ alignments and SAHMM alignments using grow-diag-final, and variations of NeurAlign with no partitioning and three different methods for partitioning. Combination features were comprised of POS tags, dependency relations, word

¹¹The remaining experiments will be conducted on only English-Chinese data because SAHMM and more annotated data were available for only this data set.

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow c$)	70.4	68.3	30.7
GIZA++($c \rightarrow e$)	66.0	69.8	32.2
SAHMM($e \rightarrow c$)	73.2	73.9	26.5
SAHMM($c \rightarrow e$)	65.8	72.2	31.3
GIZA++(gdf)	61.9	82.6	29.7
SAHMM(gdf)	62.4	85.2	28.6
NeurAlign ₁	86.8	79.1	17.1
NeurAlign ₂ [$posE_i$]	87.5	80.0	16.3
NeurAlign ₂ [$posF_j$]	87.0	79.2	16.9
NeurAlign ₂ [$posE_i, posF_j$]	87.9	80.3	15.9

Table 6.9: NeurAlign: Effects of Using a Higher Number of Input Alignments (on English-Chinese)

fertilities, and neighborhood links.

When NeurAlign was run on all data at once, i.e., no partitioning of data, the alignment error rate was reduced to 17.1%. Similar to previous experiments, partitioning the data yielded further improvements. When the data was partitioned according to POS tags on both sides, NeurAlign achieved an AER of 15.9%. Compared to the best input alignment (SAHMM($e \rightarrow c$)), NeurAlign achieved 40% relative error reduction (15.9% vs. 26.5%).

The NeurAlign results with two input alignments (Table 6.8) were compared to NeurAlign with four input alignments (Table 6.9). It was observed that the use of four input alignments yielded a relative error reduction of 15.4% and 19.3% over that of the case with two input alignments (see the last rows of Table 6.8 and 6.9). The major difference between the two is the significant difference in recall—80.3% as opposed to 74.7% and 77.2%. This significant relative improvement indicates that using more input alignments results in the addition of more correct

alignment links without an adverse impact on precision. This might be important for applications that put emphasis on recall rather than precision.

6.5.4 Effects of Training Size for NeurAlign

This section seeks to answer the question of how much training data is needed for NeurAlign to improve alignments, and how the performance is impacted by the size of the training data. For this purpose, an experiment was conducted where only a subset of the available data was used for training.

NeurAlign₂ was trained on 4 input alignments for English-Chinese: GIZA++ and SAHMM in each direction. The data was partitioned according to both POS tags, and the combination features included only dependency relation, word fertilities, and neighboring links. Ten-fold cross validation was performed for this experiment. For each fold, 1/10 of the available data was chosen as the test set, 50 sentences were randomly chosen as the validation set, and n sentences were randomly chosen as the training set—where $n \in \{50, 100, 150, 200, 250, 300, 350, 391\}$.¹²

Table 6.10 presents NeurAlign₂ results for 8 different sizes of training data. Even with 50 sentences for training, NeurAlign₂ achieves a significant relative error reduction over input alignments (17.7% vs. 26.5%). Using more training data, up to 150 sentences, reduces the AER to 16.2%. Interestingly, using more than 150 sentences does not result in any significant improvements over using only 150 sentences, i.e., the fluctuations in AER for 150 sentences and up to 391 sentences

¹²Using 391 sentences for training is equivalent to using all available data as training.

Training Size (sentences)	Pr	Rc	AER
50	86.8	78.8	17.7
100	86.9	79.5	16.8
150	87.2	80.3	16.2
200	86.9	80.4	16.3
250	87.2	80.8	16.0
300	87.2	80.6	16.1
350	87.2	81.0	15.8
391	87.9	80.3	15.9

Table 6.10: NeurAlign: Effects of Training Size (on English-Chinese)

are due to the initialization effects in neural net learning. The results indicate that having 200 sentences (150 sentences for training and 50 sentences for validation) is sufficient to obtain significant improvements over input alignments. Furthermore, even if the manually annotated data is limited to 100 sentences (50 for training and 50 for validation set), NeurAlign₂ performs significantly better than the input alignments or other combination methods.

6.5.5 Effects of Training Size for Input Aligners

Another important question is whether NeurAlign can improve the word alignments if the input alignments are better as a result of using more training data for initial aligners. To address this question, the size of training data was increased from 107K sentence pairs to 241K sentence pairs to obtain better GIZA++ alignments. The new training data consists of sentences from FBIS corpus, with nearly 9.2M English and 7.3M Chinese words. As before, GIZA++ was trained using 5 iterations of Model 1, 5 iterations of HMM, 3 iterations of Model 3, and 3 iterations of Model 4. The average sentence length for this training data is 38 words for

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow c$)	72.2	70.5	28.6
GIZA++($c \rightarrow e$)	67.5	71.6	30.6
GIZA++(gdf)	64.0	84.4	27.7
NeurAlign ₁	84.9	74.8	20.3
NeurAlign ₂ [$posE_i$]	86.0	76.4	18.9
NeurAlign ₂ [$posF_j$]	86.2	74.7	19.8
NeurAlign ₂ [$posE_i, posF_j$]	86.1	76.8	18.7

Table 6.11: NeurAlign: Effects of Using More Data to Train Input Alignments (on English-Chinese)

English and 30 words for Chinese. In comparison to the AER scores in Table 6.3, the scores for input alignments and grow-diag-final combination method increased 2 (absolute) points as a result of using more training data.

Table 6.11 presents the results for two uni-directional GIZA++ alignments and their combination using grow-diag-final method, and variations of NeurAlign with no partitioning and three different methods for partitioning.¹³ The POS tags, dependency relations, word fertilities, and neighborhood links were used as combination features.

Without any partitioning, NeurAlign achieves an alignment error rate of 20.3%—a significant relative error reduction of 26.7% over GIZA++(gdf). Partitioning the data according to POS tags results in significantly better results over no partitioning. When the data is partitioned according to POS tags on only one side, NeurAlign achieves 18.9% and 19.8% AER, respectively. When the data is partitioned according to both POS tags, NeurAlign reduces AER to 18.7%—a sig-

¹³Because of scalability issues, SAHMM output could not be generated when the training data was increased to 241K sentence pairs.

nificant relative error reduction of 32.5% over GIZA++(gdf) (18.7% vs. 27.7%). Compared to the input alignments, the best version of NeurAlign—using POS tags on both sides for partitioning data—achieves a relative error reduction of 34.6% and 38.9%, respectively.

When compared to results in Section 6.5.2, the relative improvements in AER are very close to the ones obtained using less training data (32.5% vs 33.7% relative error reduction over GIZA++(gdf)). This indicates that NeurAlign is still effective, even with better initial alignments.

6.5.6 Stability of Results

This section investigates the stability of NeurAlign, i.e., whether the improvements are specific to a particular set of sentences. To address this issue, the data was divided into 31 subsets, and a 31-fold cross-validation experiment was conducted on English-Chinese data using four input alignments generated by GIZA++ and SAHMM, keeping all other settings as before.¹⁴

Table 6.12 presents the statistics related to the alignment error rates, for GIZA++(gdf) and NeurAlign₂ for a 31-fold cross validation experiment on English-Chinese data. NeurAlign₂ was trained using neighborhood features and fertilities of the words after partitioning the data according to POS tags on both sides.

To compare the alignments generated by GIZA++(gdf), SAHMM(gdf) and NeurAlign₂, the following statistics were collected: the minimum, the maximum,

¹⁴The first 30 sets contain 16 sentences, and the last one contains 11 sentences.

	GIZA++(gdf)	NeurAlign₂	Absolute Improvement	Relative Improvement
Min	24.7	9.2	9.7	33.1
Max	35.4	19.9	16.9	64.2
Median	29.7	16.4	14.1	47.1
Mean	29.6	15.9	13.7	46.4
StdDev	2.7	2.2	2.0	5.7

	SAHMM(gdf)	NeurAlign₂	Absolute Improvement	Relative Improvement
Min	21.2	9.2	7.8	33.6
Max	34.3	19.9	16.7	62.2
Median	28.4	16.4	12.3	44.3
Mean	28.4	15.9	12.6	44.2
StdDev	3.2	2.2	2.0	5.0

Table 6.12: Stability of NeurAlign (on English-Chinese)

the mean, the median, and the standard deviation values for all 3 systems as well as for the absolute and relative improvements obtained by NeurAlign₂ over 31 subsets of the data. The lowest AER for NeurAlign₂ is 9.2% and the maximum AER is 19.9%, which is significantly better than the minimum AER obtained by GIZA++(gdf) and SAHMM(gdf). The average AER for GIZA++(gdf) and SAHMM(gdf) are 29.6% and 28.4% as opposed to the average AER of 15.9% for NeurAlign₂. The average absolute improvement achieved by NeurAlign₂ over GIZA++(gdf) and SAHMM(gdf) are 13.7% and 12.6%, which are equivalent to an average relative improvement of 46.4% and 44.2%, respectively. The standard deviations indicate that the improvements by NeurAlign₂ are statistically significant, and the improvements are not specific to a particular set of sentences but general enough for any data set.

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow a$)	66.4	64.7	34.5
GIZA++($a \rightarrow e$)	68.1	76.5	27.9
GIZA++(gdf)	60.2	83.0	30.2
NeurAlign ₁	88.7	67.3	23.5
NeurAlign ₂ [$pos E_i$]	88.2	72.4	20.5

Table 6.13: GIZA++ vs. NeurAlign (on English-Arabic)

6.5.7 Experiments on Languages With Scarce Resources

As shown in the previous experiments, NeurAlign performs best when the data is partitioned according to POS tags on both sides. The need for a POS tagger on the FL side might be problematic for some languages. This section describes additional experiments on languages for which no resources are available in order to investigate whether NeurAlign yields similar improvements when the resources are more limited.

Experiments were performed on two data sets: English-Arabic and English-Romanian. The only available information on the FL side was the parallel text, i.e., no POS tagger was used on the FL side. NeurAlign₁ was trained using POS tags for only English words, neighborhood features and fertilities of the words. For training NeurAlign₂, the data was partitioned according to English POS tags and the other features were used for combination.

On English-Arabic data, GIZA++(gdf) performed worse than one of the input alignments (30.2% vs 27.9%) because there was a significant discrepancy between the two input aligners (27.9% vs. 34.5%) and grow-diag-final weighs each input equally. NeurAlign₁, on the other hand, yielded an AER of 23.5%—a sig-

Alignments	Pr	Rc	AER
GIZA++($e \rightarrow r$)	72.9	62.4	32.7
GIZA++($r \rightarrow e$)	74.9	65.7	30.0
GIZA++(gdf)	68.0	74.6	28.8
NeurAlign ₁	86.3	66.0	25.2
NeurAlign ₂ [$pos E_i$]	88.0	69.0	22.7

Table 6.14: GIZA++ vs. NeurAlign (on English-Romanian)

nificant relative error reduction of 15.8% and 31.9%, respectively, over the input alignments. Partitioning the data according to English POS tags yields a significant boost over NeurAlign₁, achieving an AER of 20.5%. The improvements obtained by NeurAlign₂ over the input alignments are 26.5% and 40.6%, respectively. When compared to GIZA++(gdf), the relative error reduction is 22.2% for NeurAlign₁ and 32.2% for NeurAlign₂.

On English-Romanian data, NeurAlign₁ yields an AER of 25.2%. Compared to the input alignments, NeurAlign₁ achieves a relative error reduction of 16% and 23% over GIZA++($e \rightarrow r$) and GIZA++($r \rightarrow e$), respectively. When compared to GIZA++(gdf), this is equivalent to a relative error reduction of 12.5%. Partitioning the data according to English POS tags yields a significant boost over NeurAlign₁, achieving an AER of 22.7%. When compared to GIZA++(gdf), NeurAlign₂ yields a significant relative error reduction of 21.2%. The improvements over the input alignments are 24.3% and 30.6%, respectively.

In conclusion, NeurAlign yields a significant relative error reduction (26.5% on English-Arabic and 21.2% on English-Romanian) over the best alignment on both data sets. Thus, even without using any resources on the FL side, NeurAlign

achieves significant improvements on word alignments.

6.6 Summary

This chapter presented a new method, NeurAlign, to combine multiple word alignments into an improved alignment. Based on the Multi-Align framework presented in Chapter 5, a novel method that learns how to combine multiple alignments has been implemented using neural networks. This method was evaluated on four different language pairs and demonstrated to be successful even with very limited manually-annotated data. The results indicated that NeurAlign yielded at least 20% of relative error reduction over the input alignments and the best combination algorithm.

This chapter also demonstrated that partitioning the data according to features of the words and learning a different model for each partition performed better than learning only one model for the entire data. Using POS tags of the words on both sides of the corpus to partition the data, and learning a different neural network for each partition yielded significantly better alignments than applying neural networks to the entire data at once. The experiments on English-Arabic and English-Romanian data suggested that even using POS tags on only the English side was sufficient to achieve significant improvements over the input alignments and best combination techniques.

As discussed in Chapter 5, it is crucial to choose input alignments that are

accurate and diverse in order to use the ensemble approach effectively. More specifically, the input aligners should produce relatively good alignments and they should make errors that complement each other. One major advantage of the Multi-Align framework is that the combination technique can be tuned to produce a particular output. As a result, if the input-alignment errors can be identified in advance, the behavior of the combination algorithm can be changed easily by choosing a different combination function based on the types of errors induced by the input aligners.

NeurAlign reduces the effects of the biases of existing systems by taking advantage of multiple alignments. The shortcomings of different systems are overcome by the strengths of different alignment systems in the combination framework. Similarly, the problems related to lack of training data can be solved using other alignments that utilize different resources. The function words, rare words or words related to translation divergences can be aligned in a better way by employing multiple alignments that complement each other. Finally, NeurAlign enables the incorporation of linguistic knowledge either by using linguistic knowledge as features of the words during the combination step or by dividing alignment links into different classes based on linguistic features of the words.

Chapter 7

Analysis of Alignments and MT Evaluation

Thus far, this thesis has compared different word alignment systems using the community-standard alignment error rate (AER). However, AER itself is not sufficient to understand the ways in which an alignment is better or worse than another alignment. This chapter provides an extensive analysis of different alignments to elucidate the nature of the differences between two alignments. In addition, this chapter evaluates alignments externally by examining the impact of alignment improvements on the quality of MT output.

The evaluations in Chapters 4 and 6 show that ALP and NeurAlign achieve significantly better alignments than existing alignments and their combinations according to the AER metric. Table 7.1 summarizes the overall AER improvements for ALP and NeurAlign. Regardless of the foreign language, ALP achieves a relative reduction of 13-22% in AER over unidirectional GIZA++ alignments. When compared to GIZA++(int) and GIZA++(union), ALP yields a relative error reduction of 18-23%. Finally, ALP achieves a relative error reduction of 22-23% over

ALP

Foreign Language	$e \rightarrow f$	$f \rightarrow e$	int	union	gdf
Spanish	21	22	22	22	22
Chinese	13	19	18	23	23

NeurAlign

Foreign Language	$e \rightarrow f$	$f \rightarrow e$	int	union	gdf
Spanish	31	28	35	24	20
Chinese	36	29	37	38	34
Arabic	41	27	28	38	32
Romanian	31	24	31	26	21

Table 7.1: Relative Improvements in AER by ALP and NeurAlign over 5 Different Alignments (in Percentages)

GIZA++(gdf), which is the current-best alignment combination technique.

Similarly, NeurAlign yields significant improvements over the input alignments and heuristic-based combination techniques on 4 different language pairs. Over the input alignments, NeurAlign yields a relative error reduction of 24-41%. Similarly, the relative error reductions over GIZA++(int) and GIZA++(union) are in the range of 28-37% and 24-38%, respectively. When compared to the current-best alignment combination technique, i.e., GIZA++(gdf), NeurAlign reduces the alignment error rate by 20-34%.

Unfortunately, AER itself does not describe how an alignment is better or worse than another alignment. This chapter sheds light on the behavior of different alignment algorithms by comparing alignment outputs in terms of precision and recall, number of alignment links, and fertilities of the words. For different word alignment combination techniques, this chapter also investigates how ambiguities between the outputs of different aligners are resolved.

As discussed in Section 2.6, evaluating alignments in the context of another application is another form of alignment evaluation that addresses the question of whether AER improvements carry over to improvements in an external application. For this purpose, this chapter describes an evaluation of alignments in the context of phrase-based machine translation and analyzes how alignments affect the behavior of phrase-based MT systems.

The rest of this chapter is organized as follows: First, an analysis of improvements in alignments is presented. Next, a brief overview of phrase-based machine translation is described as well as the specifics of the MT system used in this thesis. Finally, an analysis of impact of word alignments on MT output is discussed.

7.1 Analysis of Improvements in Alignments

This section analyzes alignment improvement from four different points-of-views: Precision vs. recall, number of alignment links generated, number of fertilities of the words, and the resolution of ambiguous cases for alignment combination techniques.

7.1.1 Precision vs. Recall

Word alignments are used in several NLP applications. However, the importance of precision and recall might be different for different applications. AER may be viewed as an average of precision and recall; thus, the same AER may be ob-

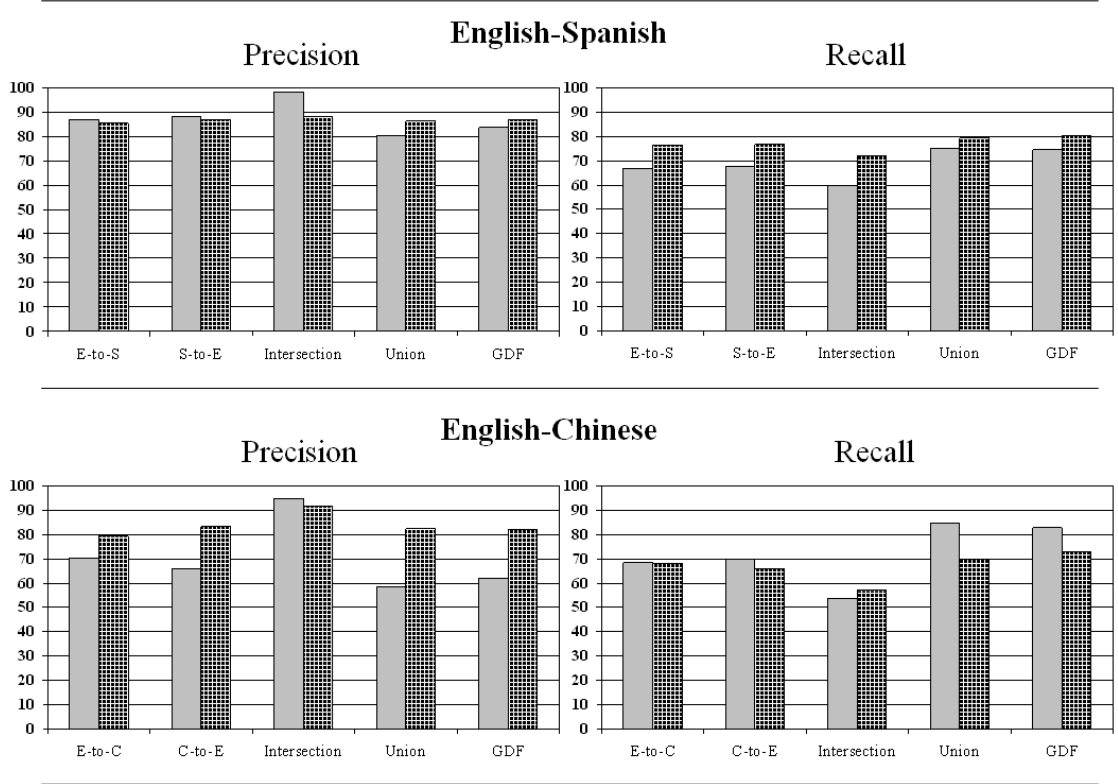


Figure 7.1: Precision and Recall for Initial Alignments and ALP

tained by increasing precision and decreasing recall accordingly, and vice versa. For instance, assuming the gold standard contains only sure alignment links, an alignment A_1 with 50% precision and 100% recall gives the same AER as an alignment A_2 with 100% precision and 50% recall. Moreover, another alignment A_3 with 100% precision, and 60% recall produces a lower AER, but this alignment might not be useful for an application that relies on recall. In such a case, the application might prefer A_1 over A_3 considering huge differences in recall values. Therefore, comparing precision and recall of two alignments is as important as comparing their AER.

Figure 7.1 present the precision and recall values for 5 initial alignments and ALP on two test sets that were used in evaluations in Chapters 4 and 6: English-Spanish and English-Chinese. In Figure 7.1, the first bar in each group represents an initial alignment and the second bar represents ALP with that initial alignment. On English-Spanish data, application of ALP to uni-directional GIZA++ alignments yields nearly the same precision values but significantly higher recall values. On English-Chinese data, however, ALP behaves exactly the opposite, yielding significant improvements in precision but comparable recall values. When GIZA++(int) is used as initial alignment, ALP yields lower precision but higher recall on both data sets (90% precision on both data sets with 55% recall on English-Chinese and 70% recall on English-Spanish). GIZA++(union) and GIZA++(gdf) behave similar to each other but their behavior is different on different languages. On English-Spanish data, both methods achieve nearly 80% precision, and 75% recall. When they are used as initial alignments, ALP increase precision to nearly 85%, and the recall to 80% percent. On English-Chinese data, GIZA++(union) and GIZA++(gdf) achieve nearly 60% precision and 80% recall. When they are used as initial alignments, ALP brings significant improvements in precision but at the expense of a reduced recall, with an overall 80% precision and 70% recall.

Figure 7.2 presents precision and recall values for four combined alignments (GIZA++(int), GIZA(union), GIZA++(gdf), and NeurAlign₂ using unidirectional GIZA++ alignments as input alignments) on four different language pairs. On English-Spanish, NeurAlign yields higher precision and recall than GIZA++(gdf)

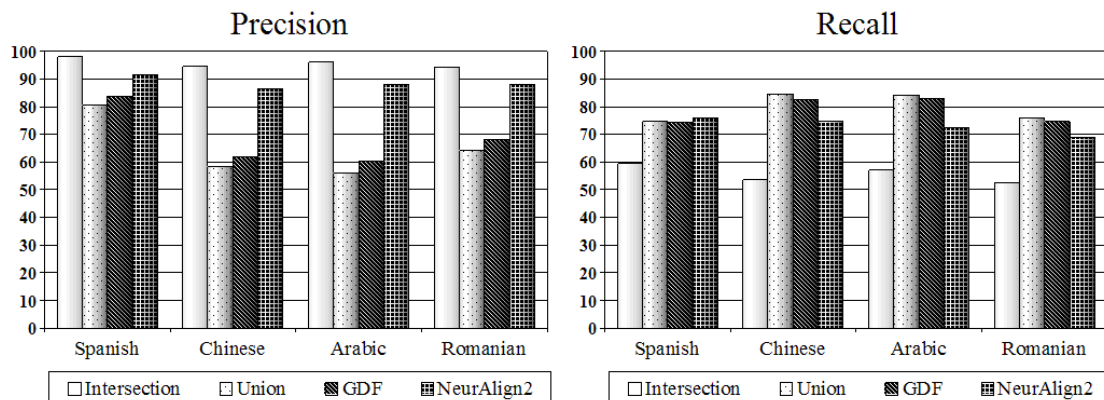


Figure 7.2: Precision and Recall for Initial Alignments and NeurAlign

and GIZA++(union). When compared to the intersection method, NeurAlign yields slightly lower precision but significantly higher recall. For the other 3 data sets, the ranking of the combined alignments according to their precision and recall values are the same. GIZA++(int) yields the highest precision (nearly 95%) but the lowest recall (in the range of 50-60%). The behaviors of the union and grow-diag-final methods are similar on all 3 data sets. Both methods achieve a low precision (in the range of 60-65%) but high recall (in the range of 75-85%). NeurAlign, on the other hand, yields significantly higher precision (nearly 85%) but lower recall (in the range of 70-75%).

In conclusion, when compared to existing alignments and their combinations, ALP and NeurAlign yield improved alignments by increasing the precision despite reductions in recall values. This clearly indicates that both methods are very conservative in adding links to the final alignment: they both leave out out links where there is not sufficient evidence in initial alignments or input alignments.

The impact of adding links conservatively is explored further in the next section.

7.1.2 Comparison of Number of Alignment Links

The precision and recall analysis shows that the alignment improvement techniques in this thesis yield alignments with higher precision but lower recall than the union and grow-diag-final method. However, this analysis is not very informative on its own because one might obtain alignments with high precision and low recall by including only a small number of links (as in the case of intersection method), or alignments with low precision and high recall by including many links (as in the case of union method). Therefore, precision and recall measures are more informative if the number of alignment links is also taken into account. This section compares the number of links in different alignments and investigates which alignment is closest to human judgment.

Table 7.2 lists the number of alignment links for two subsets of manual alignments (with only sure links, and with both sure and probable links), two GIZA++ alignments, 5 combined alignments, and an improved alignment using ALP on English-Chinese test data. For ease of comparison, the rows corresponding to automatically generated alignments are sorted according to the number of alignment links in the test data. The number of English words is 13,651 and the number of the FL words is 12,236 in test data. As shown in the “Test Data” column of Table 7.2, the highest number of alignment links (18,135 links) is obtained by the union method while the intersection produces the lowest number of alignment

Alignment	Test Data			Training Data		
	Total links	Links per e_i	Links per f_j	Total links	Links per e_i	Links per f_j
Human (sure)	11326	0.83	0.93	N/A	N/A	N/A
Human (sure+prob)	14467	1.06	1.18	N/A	N/A	N/A
GIZA++(int)	6592	0.48	0.54	1991723	0.49	0.61
ALP[GIZA++(gdf)]	10584	0.78	0.86	2914340	0.71	0.89
NeurAlign[2 inputs]	10863	0.80	0.89	3026894	0.74	0.92
NeurAlign[4 inputs]	11675	0.86	0.95	3206264	0.78	0.98
GIZA++($e \rightarrow c$)	11739	0.86	0.96	3136849	0.77	0.96
GIZA++($c \rightarrow e$)	12988	0.95	1.06	3865443	0.94	1.18
GIZA++(gdf)	16690	1.23	1.36	4646485	1.13	1.42
GIZA++(union)	18135	1.33	1.48	5010569	1.22	1.53

Table 7.2: Number of Alignment Links for Different Alignments (on English-Chinese)

links (6,592 links). The numbers of alignment links generated by other alignments are between these two numbers. Among the combination and alignment improvement methods, ALP results in the smallest number of links while NeurAlign yields similar numbers. Grow-diag-final produces nearly as many alignment links as the union method.

The critical issue is to find out which alignment simulates the human behavior. When compared to human alignments with only sure alignment links, the number of links generated by NeurAlign (both with 2 and 4 input alignments) is closest to the number of links in human alignment. When probable alignments are taken into account, ALP and NeurAlign yield a lower number of alignment links when compared to human alignment.

To investigate whether the correlation between numbers of alignment links carries over to different data, the same analysis was done for the training data,

as shown in the “Training Data” column of Table 7.2. This set consisted of 107K sentence pairs, with 4,099,116 English words and 3,275,965 Chinese words. Because the manual alignment for this data set was not available, a comparison was made between the GIZA++ alignments and their combinations. The number of alignment links depends on the size of the parallel text; therefore, the numbers of alignment links over two different data sets are not necessarily comparable. One solution is to compare the number of links per English (or Chinese) word, instead of the actual number of links, across the two data sets.

Columns 2 and 5 of Table 7.2 indicate the number of alignment links per English word and columns 3 and 5 indicate the number of links per Chinese word. Since the number of Chinese words is much smaller than the number of English words, the number of links per Chinese word is higher than the number of links per English word. The ordering of the number of links per word is exactly the same on both the testing and training data, indicating that the algorithms behave similarly on different data sets. For each of 5 combined alignments and ALP, the numbers of links per Chinese word are very similar on both data sets, with an absolute difference of at most 0.07. The difference is higher for the numbers of links per English word, yet the ratios for two data sets are similar.

The number of links per English word on the test data is higher than the number of links per English word on the training data. On the other hand, the number of links per Chinese word on test data is lower than the number of links per Chinese word on training data. The main reason for this behavior is the difference

between the ratio of English words to Chinese words on the data sets (1.12 on test data and 1.24 on training data).

Finally, it is worth noting that on training data, the number of links per word generated by NeurAlign and ALP is very similar to the number of links per word in the human alignment with sure links. This suggests that both methods simulate human behavior on a larger data set in a way that is similar to that of the smaller test data.

7.1.3 Comparison of Fertilities

The results in the previous section clearly show that ALP and NeurAlign yield lower recall despite significantly higher precision when compared to union or grow-diag-final methods. One possible explanation is that ALP and NeurAlign leave more words unaligned because there is not sufficient evidence in the training data to include them in the final alignment. To investigate whether this hypothesis is true, this section presents an analysis of the word fertilities for various alignments.

Tables 7.3 and 7.4 indicates the distribution of words according to the number of words they are aligned to. Both tables show the number of words with different fertilities for both test data and training data on English-Chinese for two input alignments, five different combinations of them, and ALP using GIZA++(gdf) as initial alignment. The test data includes 491 sentence pairs while the training data includes 107K sentence pairs. Each row of Tables 7.3 and 7.4 present the percentage of words that are unaligned (i.e., the fertility is equal to 0), the percentage of words

Alignment	<i>Fert=0</i>		<i>Fert=1</i>		<i>Fert > 1</i>	
	Test	Train	Test	Train	Test	Train
Human(sure)	26	N/A	68	N/A	6	N/A
Human(sure+prob)	12	N/A	74	N/A	14	N/A
GIZA++($e \rightarrow c$)	40	42	49	49	11	9
GIZA++($c \rightarrow e$)	5	6	95	94	0	0
GIZA++(int)	52	51	48	49	0	0
GIZA++(union)	4	5	75	77	21	18
GIZA++(gdf)	4	5	80	82	16	12
ALP[GIZA++(gdf)]	33	36	60	58	7	6
NeurAlign[2 inputs]	29	32	64	63	7	5
NeurAlign[4 inputs]	25	29	66	64	9	7

Table 7.3: Percentage of English Words with Different Fertilities (on English-Chinese)

that are aligned to only one word (i.e, the fertility is 1), and the percentage of words that are aligned to two or more words (i.e., the fertility is greater than 1) for two different data sets (test and training). The first two rows show the distribution of words for human alignments: The first row considers only sure alignment links while the second row examines both sure and probable alignment links. The next two rows show the distribution for input alignments, and the next six rows show the results for ALP and 5 combination techniques.

The input alignments and the intersection method are constrained by the types of multi-word alignments. For GIZA++($e \rightarrow c$), each Chinese word can be aligned to at most one English word. Similarly, GIZA++($c \rightarrow e$) allows each English word to be aligned to at most one Chinese word. This results in a high number of unaligned Chinese words (40% for test data and 42% for training data) for GIZA++($e \rightarrow c$), and a high number of unaligned English words (27% for test

Alignment	<i>Fert=0</i>		<i>Fert=1</i>		<i>Fert > 1</i>	
	Test	Train	Test	Train	Test	Train
Human(sure)	18	N/A	74	N/A	8	N/A
Human(sure+prob)	8	N/A	72	N/A	20	N/A
GIZA++($e \rightarrow c$)	4	4	96	96	0	0
GIZA++($c \rightarrow e$)	27	19	55	59	18	22
GIZA++(int)	46	39	54	61	0	0
GIZA++(union)	1	1	70	65	29	34
GIZA++(gdf)	1	1	77	72	22	27
ALP[GIZA++(gdf)]	24	24	67	65	9	11
NeurAlign[2 inputs]	24	23	64	63	12	14
NeurAlign[4 inputs]	19	21	68	64	13	15

Table 7.4: Percentage of Chinese Words with Different Fertilities (on English-Chinese)

data and 19% for training data) for GIZA++($c \rightarrow e$). The intersection method is limited to one-to-one alignments by nature; thus, the percentage of unaligned words is quite high (between 39-52%).

When the aligners are allowed one-to-many alignments, the percentage of unaligned words is very low. For instance, GIZA++($c \rightarrow e$) allows one English word to be aligned to many Chinese words; thus, only 5-6% of the English words are left unaligned. For similar reasons, GIZA++($e \rightarrow c$) leaves 4% of the Chinese words unaligned.

The union and grow-diag-final do not constrain the types of allowed alignments. Both methods include all or most of the links in cases where the alignment is ambiguous, i.e., where the input alignments generate different outputs.¹ As a result, the percentage of unaligned words is very low (4-5% for English words and

¹Section 7.1.4 discusses the resolution of ambiguous cases in more detail.

1% for Chinese words) in comparison to other alignments.

NeurAlign strikes a balance between the intersection and the union. Using two input alignments, NeurAlign leaves 29-32% of English words and 23-24% of Chinese words unaligned for both data sets. Using 4 input alignments reduces the percentage of unaligned words significantly. NeurAlign with 4 input alignments leaves 25-29% of English words and 19-21% of Chinese words unaligned. ALP performs similar to NeurAlign with 2 input alignments, leaving 33-36% of English words and 24% of Chinese words unaligned.

The crucial question is which alignment best reflects the text-alignment results of a human. This is answered by comparing the percentage of unaligned words in the human alignment with the percentage of unaligned words for a given alignment. Considering only sure alignment links, NeurAlign with 4 input alignments is the closest to human alignment on the test data (26% vs. 25% of English words, and 18% vs. 19% of Chinese words). However, when possible alignment links are also taken into account, the use of union or grow-diag-final provides a better approximation to human behavior.

7.1.4 Resolution of Ambiguous Links

In the process of combining multiple alignments, it is usually easy to predict the true alignment when all input alignments agree on the decision for a given alignment link. In most of the combination schemes, the common strategy is to take the decision of the input alignments as the final output if all agree on this decision.

In other words, for a given alignment link, if all input alignments agree on the existence of the link, the combiner produces 1, implying the existence of the link. Analysis of outputs of different combination algorithms show that this is not a bad decision because this type of combination produces an alignment error rate in the range of 1-4% for those cases.

Similarly, if all input alignments exclude a given alignment link, most of the combination algorithms (e.g., intersection, union, majority voting, and grow-diagonal) simply choose to exclude it in the final output as well. As supported by empirical evidence, this approach is unsuccessful because the true alignment usually includes many alignment links that are missed by all input alignments. As a result, the combination algorithms yield 100% alignment error in these cases. The alignment improvement techniques presented in this thesis attempt to find alignment links that are missed in the initial alignments by examining the neighboring links, fertilities of the words, and linguistic features associated with the words. All three methods achieve the most success on the English-Spanish data but the improvement for these cases is low. The major difficulty is that it is not possible to resolve ambiguous cases without any additional knowledge and it is not clear what kind of additional knowledge is required for handling these cases properly.

A more tractable task is to resolve the ambiguities, i.e., the cases where input alignments generate different outputs—henceforth called *ambiguous cases*. Table 7.5 summarizes the behavior of the 4 combination methods in the resolution of ambiguous cases when GIZA++($e \rightarrow c$) and GIZA++($c \rightarrow e$) are used as

Alignment A	$A=-1$ $H=-1$	$A=1$ $H=-1$	$A=-1$ $H=1$	$A=1$ $H=1$	Pr	Rc	AER
GIZA++(int)	7220	0	4323	0	0	0	100
GIZA++(union)	0	7220	0	4323	37	100	45
GIZA++(gdf)	1196	6024	249	4074	40	94	43
NeurAlign[2 inputs]	6155	1065	1507	2816	72	65	31

Table 7.5: Resolution of Ambiguous Cases Among 2 Alignments (on English-Chinese)

input alignments. In Table 7.5, A corresponds to the combined alignments, H corresponds to manual alignment, $A = 1$ ($H = 1$) corresponds to the existence of a link in A (H), and $A = -1$ ($H = -1$) corresponds to the absence of a link in A (H). The four columns containing A and H values specify the number of alignment links for four cases:

1. Both the combination method and the human alignment yield -1 ($A = H = -1$),
2. The combination method yields 1 but the human alignment yields -1 ($A = 1; H = -1$),
3. The combination method yields -1 but the human alignment yields 1 ($A = -1; H = 1$), and
4. Both the combination method and the human alignment yield 1 ($A = H = 1$).

The last 3 columns show the precision, recall, and alignment error rate for ambiguous cases. The intersection approach follows a leave-all-out strategy while

the union approach follows a take-them-all strategy. As a result, the intersection method yields zero precision and recall and 100% AER. Union, on the other hand, achieves 37% precision and 100% recall, which results in 45% AER. Grow-diag-final (gdf) method produces very similar results to the union method, yielding an AER of 43%. NeurAlign is more conservative in adding alignment links for ambiguous cases than the union and grow-diag-final methods, achieving 72% precision, 65% recall, and 31% AER. When compared to grow-diag-final, NeurAlign achieves a relative error reduction of 27%.

The major difference between grow-diag-final and NeurAlign is that the latter is more conservative in adding alignment links. This behavior reflects the distribution of alignment links in the human alignment. NeurAlign is a supervised learning method; thus, it attempts to learn human alignment. Among $7,223 + 4,323 = 11,523$ alignment links where the input aligners differ, only 4,323 (37%) of those links are labeled as valid links by humans. Grow-diag-final labels 10,098 of those links as valid links while NeurAlign labels only 3,881 links as valid links. As a result, grow-diag-final yields a very high recall (94%) but very low precision (40%). These values are more balanced for NeurAlign, i.e., NeurAlign sacrifices recall for an increased precision.

As presented in Chapter 6, using a higher number of input alignment systems yields lower AER. Table 7.6 presents a similar analysis of ambiguous cases where 4 input alignments (GIZA++ and SAHMM in both directions) are used as input to NeurAlign, following the same notation in Table 7.5. The first row corresponds to

Number of + Classifications	A=-1 H=-1	A=1 H=-1	A=-1 H=1	A=1 H=1	Pr	Rc	AER
1	9145	341	1238	516	60	29	60
2	1665	714	590	2142	75	78	23
3	186	184	57	1777	90	96	6
1 or 2 or 3	10996	1239	1885	4435	78	70	26

Table 7.6: Resolution of Ambiguous Cases Among 4 Alignments (on English-Chinese)

the alignment links where only one of the input alignments yield a 1. The second (third) row corresponds to cases where two (three) of the input alignments label the link as a valid link. The fourth row is an aggregation of the first three rows, i.e., cases where at least one aligner differs from the rest. There are 3 important results:

1. As the number of aligners that yield 1 increases, the agreement of aligners tends to drop, resulting in fewer alignment links. For instance, the first row includes 11,240 alignment links while the second and third row include 5,111 and 2,204 links, respectively.
2. As the number of aligners that yield 1 increases, the precision, the recall and AER improve significantly. For instance, the AER goes down from 60% to 6% as the number of aligners that yield a 1 increases from 1 to 3.
3. Using four input alignments yields better results than using only 2 input alignments (see the last rows in Table 7.5 and 7.6). For all ambiguous cases, the precision goes up from 72% to 78%, the recall goes up from 65% to 70%, and the AER goes down from 31% to 26%.

Input Alignments	Pr	Rc	AER
NeurAlign[GIZA++]	86.3	74.7	19.7
Upper Bound(GIZA++)	96.9	84.5	9.4
NeurAlign[SAHMM]	85.4	77.2	18.8
Upper Bound(SAHMM)	95.8	87.3	8.4
NeurAlign[GIZA++ and SAHMM]	87.9	80.3	15.9
Upper Bound(GIZA++ and SAHMM)	99.0	91.5	4.7

Table 7.7: Upper Bounds (Assuming Perfect Resolution of Ambiguous Cases) and NeurAlign Results for 3 Sets of Input Alignments (on English-Chinese)

One critical question is the upper bound on the alignment error rate, i.e., if ambiguous cases are resolved perfectly within alignment combination framework. In order to compute such an upper bound, the following heuristics are used to compute the *perfect* ensemble output:

1. If all aligners label an alignment link as an invalid link, this particular link is labeled as an invalid link in the ensemble output because there is no evidence in the input alignments to label it otherwise.
2. If all aligners label an alignment link as a valid link, this particular link is labeled as a valid link in the ensemble output. A preliminary analysis suggests that at least 96% of the links where two input aligners label a link as a valid link are also included in the gold standard.
3. If the input aligners can not agree on the decision for a given link, an *oracle* chooses the correct label. For this purpose, the gold standard is used as the oracle.

Table 7.7 presents NeurAlign scores and upper bounds for precision, recall,

and AER values assuming that the ambiguous cases are resolved perfectly as described above for the following 3 sets of input alignments on English-Chinese data:

1. 2 GIZA++ alignments: $\text{GIZA++}(e \rightarrow c)$ and $\text{GIZA++}(c \rightarrow e)$,
2. 2 SAHMM alignments: $\text{SAHMM}(e \rightarrow c)$ and $\text{SAHMM}(c \rightarrow e)$,
3. 4 input alignments: $\text{GIZA++}(e \rightarrow c)$ and $\text{GIZA++}(c \rightarrow e)$, $\text{SAHMM}(e \rightarrow c)$ and $\text{SAHMM}(c \rightarrow e)$,

The results indicate that the upper bound for AER when only two input alignments are used are 9.4% and 8.4% for GIZA++ and SAHMM alignments as input, respectively. When the number of aligners is increased to 4, the upper bound for AER is 4.7%—an absolute error reduction of 4.7% and 3.7% over using only 2 input alignments. Using 4 input alignments instead of 2 increases recall to 91.5% percent—an absolute increase of 7% and 4.2% over using only 2 input alignments. This clearly indicates that using more input alignments helps to improve word alignments by reducing the number of links for which there is no evidence in the input alignments. It is worth noting that this strongly relies on the selection of input alignments and the diversity of errors that are made by the input alignments. As discussed in Section 5.1, the best strategy is to choose ensemble members that complement each other.

A comparison of these upper bound values and the best results that are obtained by NeurAlign suggests that NeurAlign cannot completely resolve ambiguous cases and there is still room for improvement. For all 3 sets of input alignments,

NeurAlign yields nearly 10% lower precision, 10% lower recall, and 10% higher AER than the upper bounds. Chapter 8 discusses some future work to improve NeurAlign.

7.2 Phrase-based Machine Translation

To evaluate the impact of alignment improvements in an application, this thesis investigates the use of improved alignment in Pharaoh (Koehn, 2004), a phrase-based machine translation system. As background for the results of this investigation, a description of phrase-based MT and the Pharaoh implementation is presented below.

The early studies in phrase-based MT used the noisy channel model as word-based translation models. In a noisy channel model, the translation \mathbf{e} of a given foreign sentence \mathbf{f} is given by:

$$\operatorname{argmax}_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = \operatorname{argmax}_{\mathbf{e}} p(\mathbf{f}|\mathbf{e}) \cdot p(\mathbf{e})$$

This formulation allows the translation to be modeled using two components: A translation model $p(\mathbf{f}|\mathbf{e})$ and a language model $p(\mathbf{e})$. Reordering of the English phrases is modeled by a relative distortion probability distribution $p_D(f, e)$. An additional model can be used to optimize the output length by introducing a factor ω for each generated English word. Thus, the probability of a translation \mathbf{e} for a given foreign sentence \mathbf{f} is decomposed into 4 components:

1. **Phrase translation:** Controls whether the English and FL phrases are good

translations of each other.

2. **Language model:** Controls whether the output is fluent English.
3. **Distortion model:** Allows reordering of the English phrases.
4. **Word penalty:** Controls the length of the output English sentence.

Given these four components, $p(\mathbf{e}|\mathbf{f})$ may be written as follows:

$$\begin{aligned}
 p(\mathbf{e}|\mathbf{f}) &= p_{TM}(\mathbf{f}|\mathbf{e}) && \text{phrase translation} \\
 &\times p_{LM}(\mathbf{e}) && \text{language model} \\
 &\times p_D(\mathbf{f}, \mathbf{e}) && \text{distortion model} \\
 &\times \omega^I && \text{word penalty}
 \end{aligned}$$

Each of these components may be given a weight, i.e., λ_{TM} , λ_{LM} , λ_D and λ_W , in order to increase or decrease its effects on the final translation. Thus, the final translation probability can be written as:

$$p(\mathbf{e}|\mathbf{f}) = p_{TM}(\mathbf{f}|\mathbf{e})^{\lambda_{TM}} \times p_{LM}(\mathbf{e})^{\lambda_{LM}} \times p_D(\mathbf{f}, \mathbf{e})^{\lambda_D} \times \omega^{I \cdot \lambda_W}$$

In phrase based translation, it is assumed that the foreign sentence \mathbf{f} is segmented into a sequence of k phrases $\bar{f}_1^k = \bar{f}_1, \dots, \bar{f}_k$ during decoding. Each phrase \bar{f}_i is translated into an English phrase \bar{e}_i . Based on a given segmentation, the translation probability can be written as follows:

$$p_{TM}(\mathbf{f}|\mathbf{e}) = \prod_{i=1}^k p(\bar{f}_i|\bar{e}_i)$$

The distortion probability is $d(x_i - y_{i-1})$, where x_i denotes the start position of the foreign phrase that was translated into \bar{e}_i , and y_{i-1} denotes the end position of

the foreign phrase translated into \bar{e}_{i-1} . The language model is usually computed over a large English corpus using trigrams or more.

Och and Ney (2002) showed that modeling translation in the inverse direction is not always the best method. Instead, the translation can be modeled as a log-linear combination of several feature functions, including the translation probabilities in both directions. The recent implementation of Pharaoh also followed this methodology, i.e., it utilized various sources of translation probabilities. In this maximum-entropy framework, there are M feature functions h_i between \mathbf{e} and \mathbf{f} , and the probability that \mathbf{e} and \mathbf{f} are translations of each other is given as follows:

$$p(\mathbf{e}, \mathbf{f}) = \exp\left(\sum_{i=1}^M \lambda_i h_i(\mathbf{e}, \mathbf{f})\right)$$

The feature functions are typically the language model, reordering model, word penalty and various translation models such as phrase translation probabilities and lexical translation probabilities. In addition to the language model $p_{LM}(\mathbf{e})$, the distortion model $p_D(\mathbf{e}, \mathbf{f})$, and word penalty ω^I , Pharaoh uses five translation scores between \mathbf{e} and \mathbf{f} . The remainder of this section describes how Pharaoh extracts, and scores, phrases from a given alignment.

Phrases can be extracted and scored directly from parallel corpora using a phrase-based joint probability model (Marcu and Wong, 2002). Another strategy is to extract phrases from word-aligned parallel texts (Och and Ney, 2002; Koehn et al., 2003; Tillmann, 2003; Venugopal et al., 2003; Vogel et al., 2003; Zhang et al., 2003) because of the existence of several good word-alignment systems.

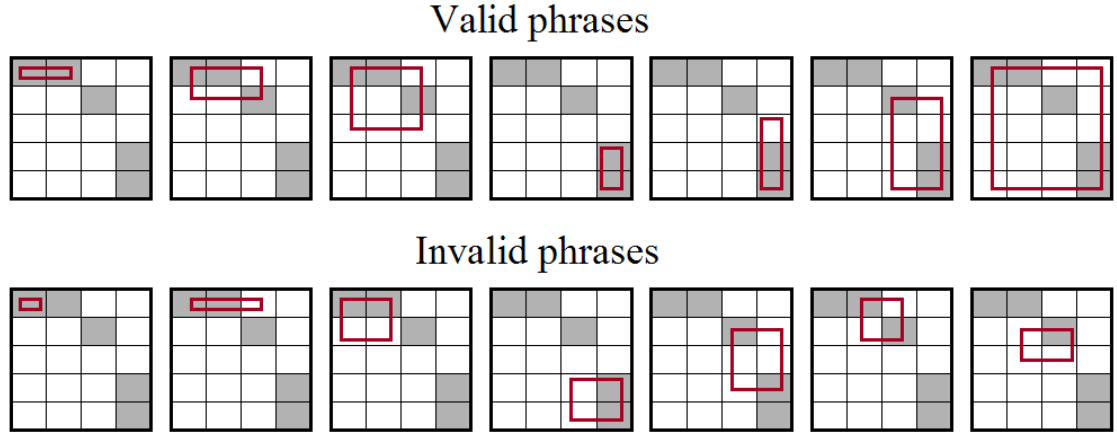


Figure 7.3: Examples of Valid and Invalid Phrase Pairs

Pharaoh assumes that the parallel corpus is first word-aligned using an existing word alignment system such as GIZA++. The next step is, for each sentence, to build a list of phrase pairs that are *consistent* with the given alignment. A given phrase pair (\bar{f}, \bar{e}) is consistent with a given alignment A if and only if each English word $e_i \in \bar{e}$ is aligned to only some word $f_j \in \bar{f}$ in A , and vice versa. In other words, for each $e_i \in \bar{e}$ and for all alignment links $(i, k) \in A$, $f_k \in \bar{f}$, and vice versa. Formally, a pair of phrases \bar{f} and \bar{e} is consistent with an alignment A if the following condition is satisfied:

$$\begin{aligned} & \forall e_i \in \bar{e} : (i, j) \in A \rightarrow f_j \in \bar{f} \\ \text{AND } & \forall f_j \in \bar{f} : (i, j) \in A \rightarrow e_i \in \bar{e} \end{aligned}$$

Figure 7.3 illustrates a word alignment matrix with 5 English words and 4 FL words, where alignment links are shown with gray boxes. The first 7 examples (the first row) are some of the valid phrase pairs, i.e., consistent with the given alignment, while the second row illustrates some of the invalid phrase pairs. For

a phrase to be valid, there should not be any alignment link above or below or to the right or to the left of the box. Note that all the words in the phrase pairs need not to be aligned to each other, i.e., the phrases may be expanded to include a word that is unaligned (examples 3 and 5 in Figure 7.3).

After extracting all phrase pairs that are consistent with the initial alignment, the next step is to score phrase pairs. Pharaoh computes five translation scores for each phrase pair (\bar{f}, \bar{e}) for a given alignment a :

1. Phrase translation probability $\phi(\bar{e}|\bar{f})$,
2. Lexical weighting $lex(\bar{e}|\bar{f})$, and
3. Phrase translation probability $\phi(\bar{f}|\bar{e})$,
4. Lexical weighting $lex(\bar{f}|\bar{e})$,
5. Phrase penalty (equal to $exp(1) = 2.718$).

Phrase translation probabilities $\phi(\bar{f}|\bar{e})$ and $\phi(\bar{e}|\bar{f})$ are estimated by relative frequency:

$$\phi(\bar{f}|\bar{e}) = \frac{count(\bar{f}, \bar{e})}{\sum_{\bar{f}'} count(\bar{f}', \bar{e})}$$

$$\phi(\bar{e}|\bar{f}) = \frac{count(\bar{f}, \bar{e})}{\sum_{\bar{e}'} count(\bar{f}, \bar{e}')}$$

The motivation behind lexical weighting is to validate the *goodness* of the phrase pairs by checking how well the words in the phrase translate to each other. For this purpose, one needs lexical translation probability distributions $w(e_i|f_j)$

and $w(f_j|e_i)$, which are computed exactly the same way phrase translation probabilities are computed. Thus, for a given phrase pair (\bar{f}, \bar{e}) and an alignment a between the foreign word positions $j = 1, \dots, J$ and the English word positions $i = 1, \dots, I$, the lexical weighting is computed as follows:

$$\begin{aligned} \text{lex}(\bar{f}|\bar{e}, a) &= \prod_{j=1}^J \frac{1}{|\{i|(i, j) \in a\}|} \sum_{\forall (i,j) \in a} w(f_j|e_i) \\ \text{lex}(\bar{e}|\bar{f}, a) &= \prod_{i=1}^I \frac{1}{|\{j|(i, j) \in a\}|} \sum_{\forall (i,j) \in a} w(e_i|f_j) \end{aligned}$$

If there are multiple alignments for a given phrase pair (\bar{f}, \bar{e}) , the highest lexical weight is selected for that pair:

$$\text{lex}(\bar{f}|\bar{e}) = \max_a \text{lex}(\bar{f}|\bar{e}, a)$$

$$\text{lex}(\bar{e}|\bar{f}) = \max_a \text{lex}(\bar{e}|\bar{f}, a)$$

7.3 MT Evaluation: Results and Analysis

This section investigates how the improved alignments affect the quality of the off-the-shelf phrase-based MT system described above, Pharaoh (Koehn, 2004).

Pharaoh can be mainly divided into 2 components:

1. Generating a phrase table from a word-level aligned parallel corpora,
2. Decoding a set of FL sentences using the generated phrase table and a language model.

The objective of the MT evaluation in this thesis is to measure the impact of word-alignment improvement on the quality of MT system. For this purpose, all components of the MT system are kept the same except the component where the phrase table is generated from a given alignment. A different phrase table is generated for each alignment, and based on the generated phrase table, a decoder is run on the same set of sentences. MT quality is evaluated using the standard MT evaluation metric BLEU (Papineni et al., 2002). The differences between the BLEU scores for different runs measure the impact of word alignments on the MT output.

The initial alignments are generated using two systems in two different directions (GIZA++ and SAHMM) on a parallel text of 107K sentence pairs. Based on these initial alignments, six different word alignments are generated as follows:

1. GIZA++(int): Intersection of GIZA++($e \rightarrow c$) and GIZA++($c \rightarrow e$).
2. GIZA++(union): Union of GIZA++($c \rightarrow e$) and GIZA++($c \rightarrow e$).
3. GIZA++(gdf): Grow-diag-final method with 2 inputs: GIZA++($e \rightarrow c$) and GIZA++($c \rightarrow e$).
4. ALP[GIZA++(gdf)]: Alignment link projection using GIZA++(gdf) as initial alignment, deletion and multi-word templates, generalized instantiation, and alignment error rate as the best rule selection method.
5. NeurAlign[2 inputs]: NeurAlign₂ with GIZA++($e \rightarrow c$) and GIZA++($c \rightarrow e$)

as input alignments. The data is partitioned according to POS tags in both languages.

6. `NeurAlign[4 inputs]`: `NeurAlign2` with `GIZA++($e \rightarrow c$)`, `GIZA++($c \rightarrow e$)`, `SAHMM($e \rightarrow c$)` and `SAHMM($c \rightarrow e$)` as input alignments. The data is partitioned according to POS tags in both languages.

Both ALP and `NeurAlign` are supervised learning methods; thus, they require a manually annotated corpus to learn the transformation rules and the neural networks. The same set that is used to evaluate ALP and `NeurAlign` in Chapters 4 and 6 is used to learn the transformation rules and neural networks. ALP learns the transformation rules on the entire set of 491 sentence pairs, and then applies the learned transformation rules to 107K sentence pairs to generate the final alignment for 107K sentence pairs. `NeurAlign` uses 200 sentence pairs as the validation set and the remaining 291 sentence pairs for training neural networks. Once it learns the weights associated with each neural network, it applies the learned neural networks to 107K sentence pairs to generate a combined alignment.

After generating a phrase table for each of the six alignments, the only remaining task is to run a decoder using the generated phrase tables and evaluate the output against human reference translations. As discussed in Section 7.2, Pharaoh employs 8 different parameters for modeling translation. The weights associated with each parameter may be set manually. Alternatively, these weights may be optimized to maximize BLEU score on a held-out development set where human

reference translations are available. In this chapter, the weights are optimized using minimum-error-rate training (Och, 2003) on MTEval'02 data set.

The final step is to run the decoder on the test set (MTEval'03) using the weights learned by minimum-error-rate training and then evaluate the translations against human translations.

For the language model, the SRI Language Modeling Toolkit was used to train a trigram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998) on 155M words of English newswire text, mostly from the Xinhua portion of the Gigaword corpus. During decoding, the number of English phrases per FL phrase is limited to 100 and the distortion of phrases is limited by 4.

The rest of this section discusses the differences between MT runs with different alignments in terms of the size of the generated phrase tables, the BLEU scores, and the weights associated with the Pharaoh parameters.

7.3.1 Phrase Table Analysis

Pharaoh allows the generation of phrases up to certain lengths. Koehn et al. (2003) showed that using phrases longer than 3 words did not significantly change the BLEU scores. Moreover, using longer phrases is computationally expensive, especially when the number of alignment links is small, due to larger phrase tables. Therefore, in this chapter, sentences are translated using phrases up to 3 words.

Table 7.8 presents the number of alignment links, the size of the phrase table file and the number of phrase pairs that are generated.

Alignment	# of Links	Size of file	# of Phrase Pairs
GIZA++(int)	2.0M	599MB	8,020,199
GIZA++(union)	5.0M	45MB	618,007
GIZA++(gdf)	4.6M	62MB	849,872
ALP[GIZA++(gdf)]	2.9M	362MB	4,875,836
NeurAlign[2 inputs]	3.0M	338MB	4,596,680
NeurAlign[4 inputs]	3.2M	300MB	4,079,744

Table 7.8: Size of Phrase Tables Generated by Different Alignments (on English-Chinese)

A smaller number of alignment links results in larger phrase tables because the phrases are expanded until there is an alignment link that violates the phrase. As a result, the largest phrase table (nearly 8M phrase pairs) is generated using GIZA++(int) while the smallest phrase table (only 618K phrase pairs) is generated using GIZA++(union). ALP and NeurAlign generate 2.9M-3.2M alignment links and 4.1-4.9M phrase pairs. When compared to GIZA++(gdf), ALP and NeurAlign generate significantly larger phrase tables (with nearly a 1:5 ratio).

7.3.2 MT Evaluation using BLEU

MT output was evaluated using BLEU (Papineni et al., 2002), as calculated by the NIST script (version 11a) with its default settings (case-insensitive matching of n -grams up to $n = 4$, and the shortest reference sentence for the brevity penalty).

Table 7.9 presents word alignment error rate on MTEval’02 data and the BLEU scores on both MTEval’02 and MTEval’03 data for 6 different Pharaoh runs, one for each word alignment.² The systems are sorted according to the

²Following the common practice in Chinese MT evaluations (Chiang, 2005), minimum-error-

Alignment	AER on MTEval'02	BLEU Score on MTEval'02	BLEU Score on MTEval'03
GIZA++(union)	31.6	0.2247	0.2278
GIZA++(gdf)	29.7	0.2431	0.2358
GIZA++(int)	31.2	0.2470	0.2377
ALP[GIZA++(gdf)]	22.7	0.2516	0.2368
NeurAlign[2 inputs]	19.7	0.2527	0.2396
NeurAlign[4 inputs]	15.9	0.2541	0.2463

Table 7.9: Evaluation of Pharaoh with Different Initial Alignments (on English-Chinese)

BLEU scores on MTEval'02 data. GIZA++(union) and GIZA++(gdf) yield the worst BLEU scores on both MTEval'02 and MTEval'03 data. Both GIZA++(int) and ALP perform better than GIZA++(union) and GIZA++(gdf). The best scores are obtained by NeurAlign on both data sets. Using 4 input alignments, NeurAlign achieves the highest BLEU scores of 0.2541 and 0.2463 on MTEval'02 and MTEval'03, respectively.

Except for GIZA++(int), the ranking of the systems according to AER is exactly the same as the ranking according to BLEU scores. This suggests that improving word alignment leads to improvements in MT quality. On the other hand, the results indicate that there is not a strict correlation between AER and BLEU scores. First, the BLEU score improvement is relatively small compared to AER improvement. Second, the alignments with higher AER might yield higher BLEU scores than alignments with lower AER. For instance, GIZA++(int) yields

rate training was run on MTEval'02 data to learn parameter weights that maximize the BLEU scores on MTEval'02 data. Pharaoh runs on MTEval'03 data use the parameter weights that are optimized on MTEval'02 data.

a higher AER than GIZA++(gdf) but its BLEU scores are also higher than the BLEU scores obtained by GIZA++(gdf) on both MTEval'02 and MTEval'03 data. This clearly indicates that further investigations are needed to observe the true effects of alignments on MT quality.

NeurAlign with 4 input alignments yields the highest BLEU scores, and the differences between NeurAlign and 3 GIZA++ variants are statistically significant, using a significance test with bootstrap resampling (Zhang et al., 2004). Although the BLEU score improvement is relatively small compared to AER improvement, the potential for additional alignment improvements is high enough to support a more substantial impact on the BLEU score. (Chapter 8 discusses some possible future directions related to this point.)

7.3.3 Parameters

For a better understanding of why a MT system behaves differently when phrases are generated from different alignments, Table 7.10 presents 8 parameters of the MT system that are learned on the development set (i.e., MTEval'02 data set). The first column corresponds to distortion model, the second column corresponds to the language model, the next 5 columns correspond to translation models where *PP* is the phrase penalty, and the last column corresponds to the word penalty.³

³For space reasons, Table 7.10 uses abbreviations for the names of alignments. ALP represents alignment link projection with GIZA++(gdf) as the initial alignment, NA[2] represents NeurAlign with 2 input alignments and NA[4] represents NeurAlign with 4 input alignments.

			TM					
Align.	D	LM	$p(e f)$	$lex(e f)$	$p(f e)$	$lex(f e)$	PP	WP
Union	0.109	0.234	0.059	0.072	0.072	0.095	0.040	-0.319
gdf	0.103	0.232	0.135	0.017	0.074	0.128	0.024	-0.288
ALP	0.073	0.124	0.053	-0.005	0.104	0.068	0.406	-0.163
Int	0.066	0.112	0.069	-0.068	0.119	0.071	0.384	-0.112
NA[2]	0.061	0.126	0.068	-0.004	0.111	0.042	0.452	-0.136
NA[4]	0.073	0.165	0.065	-0.005	0.171	0.010	0.348	-0.163

Table 7.10: MT Parameters (on English-Chinese)

The MT system parameter values learned from the development set are comparable for the first two alignments (union and grow-diag-final). Similarly, the values learned for the other four alignments are comparable yet completely different from the ones learned for union and grow-diag-final. Interestingly, the major difference between union and grow-diag-final and the other four alignments is the number of alignment links and the number of generated phrases (see Table 7.8). Thus, the crucial question is how the number of alignment links and the size of the phrase table affect the values of MT parameters. The most important differences between the parameter values for the first two alignments and the next four alignments in Table 7.10 can be summarized as follows:

1. The distortion model is more important for union and grow-diag-final than for the other four alignments.
2. The language model is more important for union and grow-diag-final than for the other four alignments.
3. The lexical probabilities are more important for union and grow-diag-final

than for the other four alignments.

4. The effect of the phrase penalty is insignificant for union and grow-diag-final while the phrase penalty is the most dominant factor for the other four alignments.
5. The word penalty is more important for union and grow-diag-final, which results in shorter translations than those generated by the other four alignments.

7.4 Summary

This chapter presented an extensive analysis of word alignments in order to elucidate the nature of the improvements obtained by ALP and NeurAlign. The alignments were compared using their precision and recall values, the number of alignment links generated, the fertilities of the words, and resolution of ambiguous cases when there are multiple input alignments. An external evaluation of improved alignments was performed using an off-the-shelf phrase-based MT system, revealing that modest BLEU score improvements are possible with a reduced AER. Finally, the impact of different word alignments on the behavior of the MT system was investigated.

Chapter 8

Conclusions

This thesis has introduced two new frameworks to improve existing word alignments using supervised learning techniques. In the first framework, called Alignment Link Projection (ALP), the frequent errors made by an initial alignment system have been learned using transformation-based learning, and alignments have been improved by correcting these errors. The second framework, Multi-Align, is an alignment combination framework that improves word alignments by combining evidence coming from existing word alignment systems. A neural-network based implementation of the Multi-Align framework, NeurAlign, was also presented.

Word alignments were evaluated using alignment error rate. The evaluations showed that both ALP and NeurAlign yield significant improvements over existing alignments on four different language pairs. Both methods require manually aligned data to learn transformation rules and neural networks, respectively. However, it has been shown that both approaches were quite effective even when the annotated data was very limited. The results also proved that using a higher

number of input alignments increased the improvement rate even further.

The improved alignments were also evaluated inside another NLP application, specifically phrase-based MT. Both ALP and NeurAlign yielded higher BLEU scores than existing alignments although the improvements in BLEU scores were modest in comparison to AER improvements.

The rest of this chapter describes the contributions of this thesis, limitations, and problems of the work presented in this thesis, and directions for future work.

8.1 Contributions

The following contributions have been made by this thesis:

- Error analysis on word alignments based on linguistic properties of the words, such as POS tags, dependency relations, and semantic-based classes.
- Introduction and implementation of a word alignment improvement module that targets translation divergences.
- A novel word alignment correction system that characterizes alignment errors systematically using machine learning and improves word alignments by correcting frequently occurring errors.
- The first use of classifier ensembles on word alignment problem and introduction of a new framework for combining different word alignments, which might take as many aligners as possible as input, regardless of their under-

lying model and resources they employ.

- The easy integration of linguistic knowledge into statistical models without the need for large modifications to existing word alignment systems.
- The use of neural networks to implement an instantiation of the alignment combination framework to improve word alignments within the combination framework.
- The analysis of alignments that elucidates the sources of improvements by one alignment over another.
- The investigation of the impact of improved word alignment on machine translation (specifically on phrase-based MT).

Although this thesis focuses on improving word alignments, it also contributes to other fields in Computer Science by:

1. Demonstrating that error-driven learning techniques are useful for improving existing systems and that they can be used easily to eliminate the need for handling each system separately.
2. Introducing an application of classifier ensembles in a new field and demonstrating that classifier ensembles perform better than individual classifiers in this new field.
3. Demonstrating that neural networks can be successfully and easily applied to a new domain.

4. Demonstrating that linguistic knowledge can be easily integrated into existing systems without changing the internals of existing systems.

8.2 Limitations

The alignment improvement techniques presented in this thesis have the following limitations:

- DUSter requires for humans to identify translation divergences between two languages, and create rules that handle them properly.
- Both ALP and NeurAlign are supervised learning methods; thus, they need manually aligned data. Both approaches are sensitive to noise (i.e., inconsistencies) in manual annotation.
- Both ALP and NeurAlign make the assumption that the test and training data exhibit the same characteristics, i.e., alignment decisions made on the test data are comparable to those made on the training data.
- ALP relies on the existence of some form of linguistic knowledge to instantiate the templates. Thus, it requires additional tools to obtain linguistic knowledge in both languages. This is not a bottleneck for English; however, those tools may not be available for other languages. NeurAlign has been shown to be effective with English-only resources although additional resources in the other languages might be helpful.

- During the process of learning frequently alignment errors, ALP learns only those rules that adhere to the structure of a prespecified set of hand-generated templates. Although it is possible to design a larger set of templates to overcome this limitation, this is a computationally expensive solution. Therefore, one needs to choose the set of templates carefully by taking differences between languages into account.
- In order to make reasonable generalizations, both ALP and NeurAlign rely on the correct labeling of POS tags, dependency relations, etc. If the tools that generate these labels produce errors, the resulting alignments will be less reliable.
- NeurAlign treats each pair of words in two sentences as a separate instance to be classified. This leads to an explosion in the number of classification instances. This might not be a problem for learning neural networks since these instances are generated only once on a small data set for each language pair. However, scaling is an important issue for the application of learned neural networks to larger data.

8.3 Future Work

This section describes possible directions for future work on the improvement of word alignments and better integration in other NLP applications.

Employing Additional Features: This thesis has described alignment improvement approaches that incorporate both the linguistic features of words and the features of word alignment outputs. However, the linguistic knowledge has been limited to POS tags and dependency relations. Other researchers have successfully used additional resources, such as bilingual dictionaries, Wordnet entries, and morphological analyzers, to improve word alignments (Liu et al., 2005; Ittycheriah and Roukos, 2005). Incorporation of such resources might provide additional improvements (both in precision and recall) in ALP and Multi-Align frameworks.

Increasing Recall: As discussed in Chapter 7, the alignment combination techniques yield alignments with very high precision but smaller recall values. The primary reasons for this outcome are: (1) the lack of sufficient evidence when all input aligners label a particular alignment link as invalid and (2) poor resolution of ambiguous links.

One way of overcoming the first problem is to use additional word alignments as input. Fortunately, there exist several word alignment systems that achieve reasonable alignments. In the Multi-Align framework, it is easy to take advantage of all available systems. However, as discussed in Chapter 5, one has to make sure that the input alignments are complementary, i.e., using similar systems as inputs might not provide further improvements. Employing additional features, such as additional linguistic knowledge, might be helpful for inferring new alignment links even when all input aligners label them invalid.

Another approach to improving recall is better handling of ambiguous cases (pairs of words for which the input aligners generate different outputs). One way to achieve this is to filter training data according to the outputs of input aligners. An alternative approach is to apply `NeurAlign` repeatedly in order to minimize the number of unaligned words, using a smaller subset of the initial training data in successive iterations.

Recall is deemed an important factor in MT quality (Och and Ney, 2003). The MT evaluations in Chapter 7 showed that ALP and `NeurAlign` increased the BLEU scores; however, the improvements were relatively low with respect to improvements in AER. Therefore, improving recall for ALP and `NeurAlign` might provide further boosts in MT quality.

Using ALP and `NeurAlign` Together: It is worth investigating whether the two alignment improvement frameworks presented in this thesis can be used together to obtain further improvements and to determine the extent to which they complement each other. For testing such a hypothesis, ALP can be applied to any number of existing alignments, and then the resulting alignments can be fed into `NeurAlign` as input. Similarly, several aligners can be given as input to `NeurAlign`, and then the output of `NeurAlign` can be passed to ALP for additional error correction.

Implementation of the Multi-Align Using Other Approaches: Chapters 5 and 6 presented two implementations of the Multi-Align framework, using

single-layered and multi-layered perceptrons. There are other supervised learning techniques that might be used in future instantiations of the Multi-Align framework, e.g., support-vector machines (Vapnik, 1995) and maximum entropy models (Berger et al., 1996), which have been used in several other applications successfully.

Improving Efficiency: Although both ALP and NeurAlign are parallelizable, both methods still require a lot of time to be applied to large data sets. The major issue with ALP is that all possible sequences of words that match each template are evaluated against the gold standard during template instantiation in every iteration. This problem might be overcome by taking advantage of faster implementations of TBL, such as fnTBL (Ngai and Florian, 2001). The major issue with NeurAlign is that all pairs of words in the two sentences are treated as separate instances during neural net application, and nearly 94% of these word pairs are invalid alignment links. Based on the observations in Section 7.1.4, NeurAlign can be made faster by considering only the word pairs that are aligned by at least one input aligner at the expense of a slightly lower recall.

Additional External Evaluations: Chapter 7 presented an example of evaluation of word alignments inside another application, i.e., phrase-based MT. There are several other NLP applications whose performance has been shown to be influenced by the quality of word alignments significantly. These applications include projection of resources between languages, e.g., POS tags or parse trees, and bilin-

gual lexicon construction. Previous research demonstrates that the precision of alignments is more important than recall for these two applications (Hwa et al., 2002). The word alignment improvement techniques presented in this thesis achieve 85-90% precision with reasonably good recall (70-80%). Thus, the use of improved alignments for such applications is an area worthy of further study.

Investigation of Correlations between Alignments and MT: This thesis presented significant improvements in word alignments in terms of alignment error rate, yet the impact of these improved alignments on an external application, e.g., MT quality, is surprisingly small. Chapter 7 provided an analysis of how different alignments change the parameters of an MT system; however, it is not yet clear which characteristics of an alignment bring about these parametric changes. To achieve better MT output, further investigation is needed to understand the connection between alignment and different parameters of the MT modeling.

In summary, this thesis introduced two new frameworks for improving word alignments: (1) An error-driven learning approach that identifies frequent errors made by existing systems and corrects them, and (2) an alignment combination framework that takes advantage of different word-alignment systems. Both approaches have been shown to yield significant improvements over existing word alignments on different language pairs. The improvements were analyzed using various measures, including the impact of improved word alignments in an external application—phrase-based machine translation.

Appendix A

DUSTer Parameters

This chapter presents a complete list of semantic word classes (parameters) that are employed in DUSTer.

English Parameters:

- **Aspect Verbs:** begin, cease, commence, complete, continue, discontinue, end, finish, forbid, halt, happen, initiate, keep, let, proceed, prevent, prohibit, quit, repeat, remain, resume, start, stay, stop, terminate
- **Change of State Verbs:** abase, abate, abbreviate, abrade, abridge, accelerate, accentuate, acetify, acidify, activate, adjust, adulterate, advance, aestivate, africanize, age, agglomerate, air, alkalify, alter, ameliorate, americanize, amplify, amputate, anesthetize, anglicize, animate, apostatize, arabicize, atomize, atrophy, attenuate, augment, authenticate, authorize, awaken, balance, ballast, beatify, beautify, become, bedew, befriend, bifurcate, bisect, blacken, blast, bleach, blind, bloody, blunt, blur, bolshevize, botanize, botch,

brighten, broaden, brown, brutalize, bungle, burglarize, burn, burst, calcify, calm, capacitate, capsize, caramelize, carbonify, carbonize, castrate, categorize, catholicize, cauterize, centralize, change, char, cheapen, chill, chlorinate, christianize, cicatrize, circulate, circumcise, civilize, clean, clear, clog, close, clouded, coagulate, coarsen, coke, collapse, collect, colonize, commercialize, complicate, compound, compress, concentrate, conciliate, condense, congeal, constrict, constringe, contextualize, contract, convalesce, cool, correct, corrode, corrugate, corrupt, counteract, cremate, crimson, crisp, cross-pollinate, crumble, crystallize, curdle, dampen, darken, darn, deafen, debilitate, decelerate, decentralize, decompose, decrease, deepen, deescalate, deflate, defrost, degenerate, degrade, dehumidify, dehydrate, demagnetize, demarcate, demobilize, democratize, depressurize, deputize, desalinate, desiccate, destabilize, deteriorate, detonate, devalue, diffract, diffuse, dilute, dim, diminish, dirty, disintegrate, dislocate, disorganize, disperse, dissipate, dissolve, distend, diversify, divide, domesticate, double, drain, dry, dull, dwindle, ease, eclipse, economize, effectuate, effeminate, elapse, elegize, emaciate, emasculate, embalm, embitter, embrocate, emphasize, empower, empty, emulsify, energize, enhance, enlarge, enslave, entitle, equalize, equilibrate, escalate, eternalize, europeanize, evangelize, evaporate, even, exaggerate, exfoliate, expand, explode, extemporate, extemporize, facilitate, fade, fame, famish, fan, fanaticize, fatten, federate, feminize, fertilize, fictionalize, fill, finalize, firm, flatten, flood, focus, foliate, foment, foreshadow, fortify, fossilize, foster,

fraternize, fray, freeze, freshen, frost, fructify, fumigate, fuse, gasify, gelatinize, germanize, gladden, glutenize, gradate, granulate, gray, green, grow, halt, harden, harmonize, hasten, heal, heat, hebraize, heighten, hellenize, hibernate, highlight, hospitalize, humidify, hush, hybridize, hydrogenate, ignite, immobilize, immunize, impoverish, improve, inaugurate, incapacitate, incinerate, increase, incubate, index, individualize, industrialize, inebriate, infiltrate, inflate, inoculate, intellectualize, intensify, invert, iodize, ionize, irrigate, kindle, kipper, laminate, legalize, lengthen, lessen, level, levitate, light, lighten, lignify, liquefy, loop, loose, loosen, macadamize, macerate, magnetize, magnify, masturbate, mature, mechanize, mellow, melt, mend, menstruate, metabolize, mineralize, mobilize, moderate, modernize, modify, modulate, moisten, monopolize, moroccanize, motorize, muddy, muffle, multiply, mute, narcoticize, narcotize, narrow, nasalize, nationalize, naturalize, neaten, neutralize, nitrify, normalize, nourish, nurture, obfuscate, objectify, objectivate, obscure, open, operate, ossify, overhaul, overshadow, overthrow, overturn, overwork, oxidize, oxygenate, paginate, pale, palliate, palpitate, paralyze, pasteurize, pendulate, people, perfect, petrify, philosophize, phosphoresce, pickle, polarize, pollinate, polymerize, pop, popularize, populate, preserve, procreate, progress, proliferate, propagate, publicize, pulverize, purify, purple, pustulate, putrefy, quadruple, quicken, quiet, quieten, rarefy, rationalize, rear, rectify, redden, reduce, refine, reform, refract, refrigerate, regionalize, regularize, regulate, rekindle, relapse, renovate, reopen, replicate,

reproduce, reshape, resurface, resurge, retard, retouch, retrogress, reupholster, reverse, revive, revolutionize, ripen, roughen, round, rupture, saponify, satirize, scarp, scorch, sear, secularize, self-eternalize, service, sharpen, short-circuit, shorten, shrink, shrivel, shut, sicken, silence, silicify, silver, simplify, singe, sink, slack, slacken, slim, slow, slur, smarten, smelt, smooth, soak, sober, soften, solidify, sour, sovietize, specialize, splay, sprout, stabilize, standardize, steady, steep, steepen, sterilize, stiffen, straighten, stratify, strengthen, stretch, submerge, submerse, subside, sudanize, sulfurize, summarize, sunburn, sweeten, symbolize, sympathize, tame, tan, taper, tauten, temporize, tense, thaw, thicken, thin, tidy, tighten, tilt, tire, topple, toughen, tousle, tranquilize, transpose, treble, triple, tune, turkify, typify, tyrannize, ulcerate, underline, undermine, unfold, unfurl, uniform, unionize, unroll, unwind, urbanize, valorize, vaporize, variegate, vary, ventilate, versify, vibrate, vitrify, vocalize, volatilize, vulcanize, waken, warm, warp, weaken, westernize, wet, whiten, widen, worsen, yellow

- **Complementizers:** after, although, and, as, as soon as, because, before, but, but rather, for, just, provided that, since, so, so as to, so that, than, that, till, to, unless, until, when, where, which, while, without, yet
- **Directional Prepositions:** across, against, along, at, from, in, into, onto, to, up, down, away from, toward, towards

- **Direction Verbs:** abdicate, advance, alight, approach, arrive, ascend, come, climb, cross, defect, depart, descend, disembark, ebb, emigrate, encroach, enter, escape, exit, fall, flee, follow, go, graduate, gravitate, head, infest, infringe, invade, leave, mount, plunge, progress, recede, retire, return, rise, tumble, vacate
- **Functional Determiners:** a, an, a lot, all, another, each, every, how many, how much, many, much, my, no, one, other, our, several, some, that, the, their, these, this, those, what, which, your
- **Functional Nouns:** a lot, all, another, each, every, he, her, hers, herself, him, himself, his, how many, how much, i, it, its, itself, many, me, mine, much, myself, no, none, noone, one, other, ours, ourselves, self, several, she, some, someone, that, theirs, them, themselves, these, they, this, those, us, we, what, which, you, yours, yourself, yourselves, thou, ye, thee, thyself, thy, thine
- **Light Verbs:** be, do, give, take, put, have, make
- **Location Verbs:** abut, adjoin, antecede, arrange, attend, be, belong, besiege, bestride, blanket, border, bound, bracket, bridge, bury, cap, circle, comprise, confine, consist, constrain, contain, corner, cover, cross, delimit, deposit, disarrange, displace, dominate, edge, encircle, enclose, encompass, engulf, fence, fill, flank, follow, forego, frame, frequent, have, head, hit, hug, immerse, implant, include, insert, install, intersect, juxtapose, limit, line,

locate, lodge, meet, miss, mount, overcast, overflow, overhang, own, park, pertain, place, position, possess, precede, prefix, put, restrict, rim, ring, set, shade, siege, situate, skirt, sling, span, stash, stow, straddle, subjugate, succeed, superimpose, supersede, support, surround, top, touch, traverse, underlie, uphold

- **Modal Verbs:** can, can't, cannot, could, couldst, couldn't, did, didn't, didst, do, does, doesn't, doest, doeth, don't, done, dost, doth, had, hadn't, hadst, has, hasn't, hast, hath, have, haven't, having, may, mayest, mayn't, might, mightest, mightn't, must, mustn't, needn't, ought, oughtest, oughtn't, should, shouldest, shouldn't, used, usedn't, wast, wert, would, would'st, wouldest, wouldn't, wouldst
- **Motion Verbs:** amble, backpack, bob, bolt, bounce, bound, bow, bowl, buck, canter, caper, carom, cascade, cavort, charge, clamber, climb, clump, coast, crawl, creep, cut, dance, dandle, dart, dash, digress, dodder, drift, drop, eddy, falter, fidget, file, flap, flit, float, flutter, fly, frolic, gallop, gambol, glide, go, gosestep, gyrate, hasten, hike, hobble, hop, hopple, hover, hurry, hurtle, inch, jiggle, jog, joggle, journey, jump, kick, lead, leap, limp, lollop, lope, lumber, lurch, march, meander, migrate, mince, mosey, move, muck about, nip, oscillate, overtake, pad, parade, pass through, perambulate, plod, plunge, pounce, prance, promenade, prowl, pulsate, quake, quiver, race, ramble, range, reel, revolve, roam, rock, roll, romp, rotate, route, rove,

run, rush, sashay, saunter, scamper, scoot, scam, scramble, scud, scurry, scutter, scuttle, shake, shamble, shuffle, sidle, skedaddle, skip, skitter, skulk, sleepwalk, slide, slink, slither, slog, slouch, sneak, somersault, speed, squirm, stagger, step, stir, stomp, straggle, stray, streak, stretch, stride, stroll, strut, stumble, stump, swagger, sway, sweep, swerve, swim, swirl, tack, take a step, tear, teeter, throb, tiptoe, toddle, toil, totter, tour, traipse, tramp, travel, trek, tremble, trip, troop, trot, trudge, trundle, tumble, turn, twist, twitch, undulate, usurp, vacillate, vault, veer, venture, vibrate, waddle, wade, waft, waggle, walk, wander, wave, waver, weave, whiz, wiggle, wind, wobble, wriggle, writhe, zigzag, zoom

- **Negation Words:** no, not, none
- **Obliques:** about, above, across, after, against, ahead, along, among, around, at, away, back, before, behind, below, beneath, beside, between, beyond, by, down, during, following, for, from, in, inside, into, near, nearby, next, of, off, on, onto, out, outside, over, past, through, throughout, to, toward, towards, under, underneath, up, upon, via, with, within, without
- **Pleonastics:** it, there,
- **Psych Verbs:** abhor, ache, admire, adore, advocate, affirm, appreciate, back, bear, bewail, bother, cherish, covet, crave, deify, deplore, desire, despise, detest, disbelieve, disdain, dislike, distrust, dread, enjoy, envy, esteem, exalt, execrate, fail, fancy, favor, fear, grudge, hate, hurt, idolize, implicate,

importune, indulge, invoke, involve, itch, know, lack, lament, like, loathe, love, mean, miss, mourn, need, pain, pamper, pity, prefer, prize, reaffirm, regret, relish, resent, respect, revenge, revere, rue, savor, stand, support, tolerate, treasure, trust, try, value, venerate, vindicate, want, worship

- **Tense Verbs:** shall, shalt, shan't, will, wilt, won't

Spanish Parameters:

- **Aspect Verbs:** abandonar (*abandon*), abarcar (*undertake*), abstenerse (*abstain*), abstenerse (*abstain, refrain*), acabar (*terminate*), acabar (*cease, end, end up, finish, terminate*), acabarse (*end*), actuar (*proceed*), actuar (*proceed*), cesar (*cease*), comenzar (*begin, commence, start*), completar (*complete*), concluir (*conclude*), concluir (*terminate*), concluir (*conclude, end, terminate*), continuar (*proceed*), continuar (*continue, keep, proceed, resume*), cumplir (*complete, finish*), dejar (*quit*), discontinuar (*discontinue*), detener (*halt, stop*), empezar (*begin, commence, proceed, start*), ensayar (*repeat*), ensayar (*repeat*), extinguir (*extinguish*), finalizar (*cease, end, end up, finish, terminate*), fingir (*pretend*), inicializar (*initiate*), iniciar (*begin, commence, initiate, start*), interrumpir (*interrupt*), interrumpir (*discontinue*), ocurrir (*occur*), ocurrir (*occur, happen*), originar (*start*), parar (*cease, halt, stop*), pasar (*happen*), principiar (*begin*), proceder (*proceed*), proseguir (*continue, proceed*), quedar (*end, stop*), reanudar (*resume*), reasumir (*reassume*), reasumir (*reassume, resume*), recitar (*repeat*), recomenzar (*recommence*), re-

comenzar (*begin again*), reiterar (*repeat*), reiterar (*repeat*), rematar (*end up*), repetir (*repeat*), repetir (*repeat*), resumir (*resume*), retocar (*finish*), seguir (*continue, resume*), suceder (*occur, transpire, succeed*), suceder (*happen, occur, transpire*), suprimir (*abolish*), suspender (*discontinue*), terminar (*terminate*), terminar (*finish*), transcurrir (*transpire*), transpirar (*transpire*), zanjar (*conclude, finish*)

- **Change Of State Verbs:** abalanzar (*balance, hurl, impel, rush, swoop down*), abanar (*ventilate with a hanging fan*), abanicar (*fan, fan oneself*), abaratar (*cheapen, lower price of, become cheap, fall in price*), abatir (*deject, depress, flatten, lower, swoop, dishearten*), abigarrar (*variegate*), ablandar (*soften*), abrasar (*scorch, sizzle, burn*), abreviar (*condense, shorten*), brillantar (*brighten*), abrir (*bore, open, spread, unfasten, unfold, unlatch, unlock*), aburrir (*bore, moped, tire, weary*), acallar (*hush*), acaparar (*engross, hoard, monopolize*), acelerar (*accelerate, hasten, hurry, quicken, speed, speed up, urge*), acentuar (*stress, accentuate, emphasize*), acetificar (*acetify*), acharar (*flatten*), achicar (*bail, lessen*), acidificar (*acidify*), aclarar (*brighten, clear, clarify, become clear, enlighten, explain, explicate, illuminate, thin*), acometer (*offend, thrust, tilt*), acompasar (*measure with a compass, regulate, bar, divide into measures*), acortar (*contract, shorten, shrink*), acrecentar (*increase*), acrecer (*increase*), activar (*activate, energize, sound*), adaptar (*adapt, adjust, become adapted*), adelantar (*advance, hurry along*), adel-

gazar (*slim, taper, thin*), adormecer (*blunt, drowse, lull*), adulterar (*adulterate, adulterous*), afamar (*fame, make famous, give fame*), afianzar (*secure, steady, make firm, make fast, prop, strengthen*), afilar (*edge, point, sharpen, strap*), aflojar (*loose, loosen, relax, slack, slacken, unfasten, untie*), aglomerar (*agglomerate*), agotar (*deplete, distress, drain, exhaust, harass*), agradar (*gladden, please, like, soothe*), agrandar (*distend, enlarge, increase*), agremiar (*unionize*), agriarse (*sour*), aguar (*thin*), aguzar (*sharpen*), ahumar (*fume, smoke, kipper*), ahusar (*taper*), airear (*air, rifle*), ajustar (*fit, adapt, adjust, conform, tighten*), alargar (*expand, lengthen, reach, stretch*), alcalizar (*alkalify*), alegrar (*brighten, cheer, enliven, exhilarate, gladden, rejoice*), aliar (*federate*), aligerar (*lighten*), alisar (*dub, neaten, smooth*), aliviar (*assuage, comfort, disencumber, dull, ease, exonerate, relieve, solace, soothe, unburden*), allanar (*flatten, level*), alterar (*affect, alter, unsettle*), alumbrar (*enlighten, illuminate, light, lighten*), amainar (*subside*), amarillear (*yellow*), americanizar (*americanize*), amortiguar (*dampen, soften*), ampliar (*broaden, enlarge, expand, extend, heap, mushroom*), amplificar (*magnify, amplify*), anegar (*flood, submerge*), angostar (*narrow*), animar (*brighten, cheer, comfort, embolden, encourage, enliven, excite, gladden, hearten, revive, animate*), anular (*clear, delete, void, annul, cancel, repeal, invalidate*), apaciguar (*appease, assuage, pacify, placate, smooth, soothe, allay, calm*), aplanar (*flatten, smooth*), aplastar (*clobber, crush, flatten, smash, overwhelm, squash, squelch, squish, stub*), apocar (*lessen, contract, restrict, hum-*

ble, belittle), *apoderar* (*seize, take possession of, empower*), *apresurar* (*speed up, make haste, accelerate, hurry, speed, hasten*), *apretar* (*clench, compress, crush, grip, jam, pinch, ram, squash, squeeze, squish, strain, tighten, touch, crowd together*), *apurar* (*press, worry, hurry, drain*), *aquietar* (*hush, quiet, sober*), *arder* (*blaze, burn, flame, kindle*), *armonizar* (*chime, harmonize*), *arreglar* (*arrange, fix, hire, manicure, neaten, rank, smarten, straighten, trim, repair, get ready, improve*), *arrugar* (*crinkle, crumple, line, pucker, shrivel, wrinkle*), *arruinar* (*blast, ruin, sink, smash, wreck*), *asear* (*clean, spruce*), *asurarse* (*burn, parch, worry, harass*), *atenuar* (*attenuate*), *atiesar* (*stiffen, tighten*), *atrofiar* (*atrophy*), *aumentar* (*accumulate, add, enlarge, heighten, increase, magnify, swell, wax*), *autorizar* (*allow, warrant, authorize, give permission*), *avanzar* (*advance, push, move forward*), *avivar* (*kindle, quicken*), *bailar* (*bop, dance, prance*), *balancear* (*balance, rock, swing*), *blanquear* (*blanch, silver, whiten, whitewash*), *bonificar* (*improve, credit*), *borrar* (*delete, erase, excise, expunge, obliterate, remove, wipe, scratch, blur, annul, blank, cancel, cross out, fade, disappear, be erased*), *broncear* (*bronze, brown, tan*), *brotar* (*emanate, germinate, gush, spout, sprout, spurt*), *caer* (*drop, fall, topple, tumble*), *calcificar* (*calcify*), *calentar* (*heat, bask, fire, scald, warm*), *callar* (*hush, quieten, stifle, silence, suppress, be quiet, remain silent, shut up*), *calmar* (*appease, assuage, calm, hush, lull, pacify, quiet, sober, soothe, calm, pacify, quiet, quieten, sober, soothe, subside, calm down, let up*), *cambiar* (*alter, barter, change, exchange, shift, swap, trade,*

vary), cansar (*harass, sicken, weary, tire, make weary, fatigue*), capar (*cas-
trate*), caramelizar (*caramelize*), carbonificar (*carbonify*), carbonizar (*car-
bonize, char*), catolizar (*catholicize*), cauterizar (*cauterize*), cobar (*bait, fat-
ten, feed*), cepillar (*brush, curry, smooth*), cerrar (*clasp, close, dam, lock,
shut, stop up*), chafar (*crumple, flatten, crush, rumple, squash, cut short,
silence*), chamuscar (*scorch, sear, singe*), circular (*pass, circulate*), civi-
lizar (*civilize*), clarar (*clear, make clear*), coagular (*coagulate*), cobrar (*cash,
charge, collect*), coleccionar (*collect*), colonizar (*colonize, people, settle*), com-
padecer (*sympathize*), compasar (*measure with a compass, regulate, bar, di-
vide into measures*), complicar (*complicate, get complicated, make difficult,
thicken*), componer (*compose, compromise, compound, repair, mend, make,
spruce*), comprimir (*compress, pack, squeeze*), concentrar (*concentrate, ma-
jor*), concordar (*agree, harmonize, hire*), condensar (*compress, condense,
pack, thicken*), confirmar (*approve, confirm, strengthen, sanction, ratify, cor-
roborate*), conformar (*conform, adapt, adjust, shape, resign, comply*), con-
gelar (*freeze*), congeniar (*harmonize*), congregar (*assemble, collect, congre-
gate*), conservar (*preserve, conserve, keep, maintain, retain*), consolidar (*ce-
ment, consolidate, solidify, strengthen*), contraer (*contract, catch, shrink,
make smaller*), convalecer (*convalesce, be convalescent, recover*), convertir
(*change, convert, transform, be converted*), corregir (*castigate, chasten, chas-
tise, correct, reform, mend*), corroer (*corrode*), corromper (*corrupt, putrefy,
taint*), crecer (*grow, sprout, thrive, wax*), criar (*raise, rear, breed, bring up,*

grow), *cristalizar* (*crystallize*), *cuadruplicarse* (*quadruple*), *curar* (*cure, heal, treat, be treated, get well, heal up, recover*), *curtir* (*harden, tan*), *debilitar* (*fade, soften, weaken*), *decelerar* (*decelerate*), *deformar* (*warp, deform, mutate, be deformed*), *dejar* (*bequeath, desert, lay, abandon, allow, let, permit*), *depreciar* (*cheapen, depreciate*), *depurar* (*clean, debug, purge*), *derretir* (*fuse, melt, thaw*), *derribar* (*demolish, knock down, lay, overturn, topple, tumble*), *derrocar* (*overturn*), *derrumbarse* (*collapse, tumble*), *desacelerar* (*decelerate*), *desaguar* (*drain, empty*), *desalentar* (*chill, dampen, discourage, dispirit*), *desaparecer* (*disappear, dissolve, fade, cause disappear, duck, vanish*), *desarrollar* (*cultivate, develop, evolve, germinate, grow, improve, unroll, unwind, unfurl, expound, unfold, work out, uncoil, progress*), *desarticular* (*disarticulate, put out of joint, take apart, disconnect, disjoint, disorganize*), *desatar* (*detach, loose, loosen, unfasten, unleash, untie*), *desbordarse* (*flood*), *descalzar* (*take off shoes, take off stockings, remove wedges from, dig under, undermine*), *descargar* (*acquit, discharge, dump, empty, exonerate, lighten, unburden, unload*), *descentralizar* (*decentralize*), *descoger* (*unfold, extend, spread*), *descolorar* (*fade*), *descubrir* (*bare, detect, discover, ferret, open, reveal, spot, unfold, unlock, uncover, disclose, expose, be discovered*), *desdoblar* (*unfold*), *desecar* (*desiccate, dry*), *desempolvar* (*clean, degrit, dust*), *desengrasar* (*clean, defat, degrease*), *desescalar* (*deescalate*), *desfallecer* (*faint, weaken*), *desgastar* (*erode, fray, fret, scud*), *deshacer* (*crumble, dissolve, loose*), *deshelar* (*defrost, deice, thaw*), *desimantar* (*demagnetize*), *desinte-*

grar (*disintegrate*), deslustrar (*deglaze, deluster, dull, frost, tarnish*), des-
 menuzar (*crumble, mince*), desmoronarse (*crumble*), desocupar (*empty, evac-*
uate), desorganizar (*disorganize*), despejarse (*brighten*), despertar (*arouse,*
awake, awaken, wake), desplegar (*evolve, splay, spread, unfold*), desplomarse
 (*collapse, tumble*), despuntar (*blunt*), desteñir (*fade*), destruir (*blast, crash,*
demolish, destroy, ravage, ruin, shatter, smash, wipe, wreck), desunir (*de-*
tach, loosen, sever), desviar (*shunt, slice, divert, deviate, digress, vary, stray,*
wander, go off course, keep clear of), detener (*detain, arrest, stop, halt,*
retard, stem), devenir (*happen, become*), difamar (*defame, malign*), difer-
 enciar (*differ, vary*), dilatar (*distend, expand, puff, stretch, widen*), dirigir
 (*beam, boss, direct, funnel, lead, manacle, operate, pilot, refer, manage, gov-*
ern, address, conduct, guide, dedicate, go, be writing), discrepar (*differ, dis-*
agree, vary), disipar (*dissipate, evaporate, exhale, vanish*), disminuir (*abate,*
decline, decrease, diminish, lessen, lower, shrink, slacken, taper, weaken),
 disolver (*break, dismiss, dissolve*), dispersar (*dissipate, scatter*), distender
 (*sprain*), dividir (*cleave, divide, partition, slice, slit, sunder, split, separate*),
 doblar (*bend, buckle, crease, double, flex, fold, knell, loop, toll, turn*), do-
 mar (*tame*), domesticar (*tame*), dulcificar (*sweeten*), duplicar (*double*), echar
 (*bounce, chuck, dash, expel, flash, fling, flop, pop, pour, spout, sprout, toss,*
send, throw), educar (*educate, rear*), edulcorar (*sweeten*), elevar (*heighten,*
hoist, lift, levitate, raise), emblanquecer (*whiten*), emborrachar (*soak*), em-
 botar (*blunt, dull*), embrutecerse (*coarsen*), empapar (*drench, imbue, im-*

pregnate, soak, steep), empastar (*fill*), empañar (*blur, fog, mist, tarnish*),
 empeorar (*worsen*), empequeñecer (*diminish*), empezar (*arise, begin, com-
 mence, open, start*), empinar (*steepen*), empollar (*cram, incubate, sit*), emul-
 sionar (*emulsify*), enardecer (*fall in love, court, kindle*), encanecer (*gray*),
 encender (*turn on, catch fire, fire, flame, flash, fluster, ignite, kindle, light,
 strike*), encoger (*shrink, contract, shrivel*), encrespar (*crimp, crisp*), en-
 derezar (*flatten, straighten*), endulzar (*suffuse, sugar, sweeten*), endurecer
 (*harden, stiffen, toughen*), enfermar (*sicken*), enfriar (*cool, cool down, get
 cold, chill*), engordar (*fatten*), engrandar (*enlarge, increase*), engrasar (*fat-
 ten, grease, oil*), enjugar (*dry, wipe*), enlodar (*muddy, slop*), enmagrecer
 (*grow lean, lose flesh*), enmudecer (*hush, mute*), ennegrecer (*blacken*), en-
 ranciar (*sour*), enrarecerse (*thin*), enrojecer (*crimson, redden*), ensanchar
 (*broaden, distend, enlarge, expand, extend, fill, stretch, widen*), ensombre-
 cer (*darken*), ensuciar (*dirty, muddy, slop, soil*), entorpecer (*dull, numb,
 stupefy*), entristecer (*sadden, be sad, grieve, darken*), enturbiar (*muddy*), en-
 vejecer (*age*), equilibrar (*equalize*), equilibrar (*balance*), escarchar (*frost*), es-
 carpar (*scarp, steepen*), escurrir (*drain, ooze, trickle, wring*), esparcir (*strew,
 spread, dissipate, intersperse, scatter, bestrew*), espesar (*stiffen, thicken*), es-
 tabilizar (*fix, peg, stabilize, steady*), estallar (*break, burst, crack, crash, deto-
 nate, explode, pop*), estimular (*energize, exhilarate, fuel, jog, prod, stimulate*),
 estirar (*distend, groove, strain, stretch, tauten, tighten*), estratificar (*strat-
 ify*), estrechar (*take in, jam, narrow, taper, tighten*), evacuar (*empty, evacu-*

ate, void), evaporar (*evaporate, exhale, vaporize*), exaltar (*elate, exalt, extol, heighten*), exhibir (*air*), explotar (*blast, explode*), extender (*extend, spread, enlarge, expand, overspread, reach, splay, sprawl, stretch, widen*), facturar (*facilitate, bill, invoice, check*), fanatizar (*fanaticize*), fatigar (*tire, fatigue, get tired, distress, exhaust, harass, strain*), federar (*federate*), fermentar (*ferment, sour*), fertilizar (*enrich, fatten*), firmar (*sign, firm*), flaquear (*weaken*), fomentar (*develop*), fortalecer (*arm, confirm, invigorate, strengthen*), fosilizarse (*fossilize*), frisar (*freeze*), fructificar (*fructify*), funcionar (*operate, perform, run, work*), gasificar (*gasify*), fundir (*coalesce, fuse, melt, overwhelm*), gelatinizar (*gelatinize*), germinar (*germinate, sprout*), glutenizar (*glutenize*), granular (*granulate*), hacer (*do, become, be made, brew, construct, create, fabricate, fashion, get, make, perform*), hambrear (*famish, starve*), hartar (*gorge, sicken*), hastiar (*jade, bore, tire, disgust*), helar (*chill, freeze, frost*), henchir (*fill*), hermosear (*embellish, smarten*), hinchar (*fill, lump, puff, swell, inflate*), humectar (*moisten*), humedecer (*dampen, humidify, moisten, splash*), hundir (*collapse, immerse, plunge, sink, slump, subside*), igualar (*equal, equalize, even, level*), iluminar (*light*), imantar (*magnetize*), inclinar (*incline, slant, droop, bow, lean, move, slope, stoop, tilt, tip*), incrementar (*increase, zoom*), incubar (*hatch, incubate*), infamar (*defame*), inflamar (*ignite, kindle*), inflar (*distend, inflate, puff, swell*), iniciar (*begin, commence, initiate, open, start*), intensificar (*deepen, intensify*), intercambiar (*exchange, interchange, swap*), interrumpir (*break, discontinue, halt,*

suspend), inundar (*deluge, drown, flood, inundate, submerge, overwhelm*),
 invertir (*invert*), ionizar (*ionize*), juntar (*join, meet, amass, assemble, col-
 lect, congregate, conjoin, gather, heap, unite, yoke*), ladear (*tilt, tip*), laminar
 (*lamine, roll*), lavar (*wash, clean, wipe*), levantar (*lift, raise, elevate, stand,
 get up, arise, heft, heighten, perk*), libertar (*deliver, emancipate, liberate,
 loosen, rescue*), licuar (*liquefy*), lignificar (*lignify*), limitar (*bound, narrow*),
 limpiar (*tidy, clean, cleanse, preen, swab, sweep, weed, wipe*), llagar (*ulcerate,
 make sore, hurt, wound*), llenar (*fill, replenish, be filled, become filled, stuff,
 throng*), lustrar (*brighten, glaze, polish, shine*), macerar (*macerate*), madu-
 rar (*mature, mellow, ripen*), magnetizar (*magnetize, mesmerize*), magnificar
 (*magnify*), manchar (*tarnish, dirty, blot, blur, dapple, darken, fleck, pollute,
 slurp, smear, smudge, soil, speckle, splash, splotch, spot, stain*), manejar
 (*manage, govern, drive, handle, manacle, operate, ply, thumb, wield*), man-
 tener (*maintain, keep, hold, support, preserve, stay, sustain, feed*), marchitar
 (*fade, sear, shrivel, wilt, wither*), mejorar (*ameliorate, brighten, develop, im-
 prove*), menguar (*decrease, subside*), mermar (*shrink*), mezclar (*blend, com-
 mingle, dash, melt, mix, shuffle*), minar (*mine, undermine*), minorar (*de-
 grade, lessen*), mitigar (*mitigate, alleviate, ease, soften, palliate, soothe, as-
 suage*), moderar (*ease, moderate, slacken*), modificar (*alter, change*), modu-
 lar (*modulate*), mojar (*dampen, dip, douse, drench, water, wet, whack*), mov-
 ilizar (*mobilize*), mudar (*change, move*), multiplicar (*multiply, proliferate*),
 nacer (*originate, grow, spring, hatch, engender, awaken, love, sprout, bud,*

come up, rise, rise, spring up, appear, begin to flow, begin, start, be born, begin, originate, have its origin in), nacionalizar (*naturalize, nationalize*), narcotizar (*narcotize, drug, dope*), nasalizar (*nasalize*), naturalizar (*naturalize*), naufragar (*be wrecked, sink, be shipwrecked, fail, miscarry, suffer a disaster*), nausear (*nauseate, sicken*), negativizar (*neutralize*), neutralizar (*neutralize, counteract, saturate*), nevar (*snow, cover with snow, whiten*), nitrificar (*nitrify*), nivelar (*even, level, level out, grade, level up, equalize, even out, even up, make even, balance against, adjust deficit to cover, deal with*), nobilizar (*enhance, dignify, ennoble*), normalizar (*normalize, restore to normal, standardize*), novelar (*make a novel out of, tell in novel form, fictionalize, write novels*), nublar (*darken, blurb, obscure, cloud, disturb, affect, cloud, destroy*), nuclear (*bring together, combine, concentrate, lead*), nucleizar (*bring together, combine, concentrate, provide a focus for, act as a forum for, lead*), nutrir (*nourish, feed, nurture, strengthen, support, foment, encourage*), obcecar (*obfuscate, blind*), objetivar (*objectify, objectivate*), obscurecer (*darken, shade, overcast*), obstruir (*obstruct, bar, blockade, clog, gag, impede, shut, stop up*), occidentalizar (*westernize*), ocupar (*occupy, busy, fill, deal with, be in charge of, employ*), ofuscar (*obfuscate, bewilder, confuse mentally, dazzle*), operar (*operate*), ordenar (*arrange, decree, neat, ordain, order, rank, tidy, will*), oscurecer (*blacken, blur, darken, dim, shadow, obscure*), osificarse (*ossify*), oxidar (*oxidize, rust*), pactar (*contract*), paginar (*page, paginate*), palidecer (*blanch, fade, pale*), palpitar (*palpitate, beat, throb, flutter, heave,*

throb), parar (*stop, cease, halt, still, stall, stand up*), parchar (*mend, patch*),
 perfeccionar (*improve, polish*), petrificar (*fossilize, petrify*), platear (*plate, silver*), poblar (*people, populate*), polarizar (*polarize*), poner (*become, put*),
 potar (*drink, correct and mark*), preservar (*preserve, keep, save*), progresar (*progress, improve, gain*), proliferar (*proliferate*), prolongar (*continue, extend, lengthen*), promover (*advance*), propagar (*spread, be conveyed, over-spread, propagate*), pudrir (*decompose, putrefy, rot*), pulir (*shine, become polished, buff, dub, polish, rub, smooth*), pulverizar (*powder, pulverize*), purificar (*clean, cleanse, filter, purify*), purpurar (*purple*), quebrar (*rupture*), quemar (*burn, parch, scorch, tan*), raer (*bark, debark, fray, strip*), reabrir (*reopen*), reavivar (*rekindle*), rebajar (*abate, reduce, discount, be lowered, stoop to, descend to, decry, degrade, depreciate*), reblandecer (*soften*), recaer (*relapse*), recoger (*pick up, collect, gather, pick, reap, recollect*), recolectar (*collect, gather*), reconcentrar (*concentrate, become absorbed in thought*), recopilar (*collect, compile*), redondear (*fill, firm, round*), reducir (*reduce, discount, be reduced, abate, compress, condense, decrease, deplete, depress, dock, lessen, lower, narrow, prune, shorten, slack*), reemplazar (*change, substitute*), reencender (*rekindle*), refinar (*refine*), reformar (*reform, reshape, reclaim*), reforzar (*firm, intensify*), refrescar (*brush, cool, freshen*), regar (*irrigate, shower, sprinkle, strew, water*), regocijar (*elate, exhilarate, gladden, rejoice*), regularizar (*equalize, regularize*), relajar (*loosen, relax, slack*), relimpiar (*clean again, make very clean*), rellenar (*caulk, cram, fill, pad, replen-*

ish, stuff, wad), *remediar* (*remedy, relieve, heal*), *remendar* (*patch*), *remojar* (*drench, soak*), *reparar* (*repair, mend, service, heal*), *reproducir* (*reproduce, replicate*), *requemar* (*burn again, overcook, overbake, roast to excess, sunburn*), *resumir* (*abridge, summarize, recapitulate, abstract, resume*), *resurgir* (*resurge, revive, spring up again*), *retajar* (*cut round, trim the nib of, circumcise*), *retardar* (*retard, slacken, slow*), *retocar* (*touch up, finish, retouch*), *retoñar* (*sprout*), *retrasarse* (*defer, delay, retard, slacken, slow*), *retrocesar* (*retrogress*), *reunir* (*reunite, come together, meet together, assemble, collect, flock, gather, join, meet*), *revelar* (*blab, blat, reveal, tell, unfold, unlock*), *revenirse* (*shrink, waste away*), *reventar* (*blast, burst, crack, explode*), *rizar* (*ripple, crimp, crinkle, crisp, curl, ruffle*), *romper* (*batter, break, crack, destroy, rupture, shatter, smash, sunder, wear out*), *ruborizarse* (*blush, color, crimson, flush, redden*), *sanar* (*heal*), *satirizar* (*satirize, pillory*), *sazonar* (*mellow, relish, savor, season, spice*), *secar* (*dry off, dry up, blot, dry, parch, towel, wipe*), *serenar* (*settle, sober*), *silenciar* (*muffle, silence*), *simbolizar* (*symbolize, typify, foreshadow*), *simplificar* (*simplify*), *sindicar* (*unionize*), *socarrar* (*singe*), *solidificar* (*firm, make concrete, solidify*), *soltar* (*release, free, come unfastened, come untied, drop, loose, loosen, unclasp, unfasten, unhitch, unleash, unpin, untie*), *sonrojar* (*blush, crimson*), *sostener* (*sustain, uphold, bear, prop, allege, contend, hold, steady, support*), *suavizar* (*assuage, smooth, soften, sweeten*), *subdividir* (*subdivide*), *sublimar* (*heighten*), *subrayar* (*underline, emphasize*), *sumergir* (*dip, drown, flood, immerse, plunge,*

sink, submerge), *sumir* (*immerse, sink*), *sustituir* (*change, double, substitute*), *tapar* (*close, hide, blanket, cap, close, conceal, cover, plug, shut, stop up, stopper*), *tensor* (*tauten, tense*), *tirar* (*stretch, bowl, chuck, dart, draw, fling, haul, hurl, pelt, pitch, pluck, pull, shoot, snipe, throw, toss*), *titular* (*title, entitle, be titled, be called*), *torcer* (*bend, distort, become bent, crook, sprain, strain, twist, warp, wreath, wrench*), *tornar* (*give back, return, turn, change, alter, change into, come back*), *tostar* (*parch, roast, sunburn, tan, toast*), *tranquilizar* (*tranquilize, calm, quiet, quieten, reassure*), *transformar* (*transform, change*), *transponer* (*transpose, transplant, disappear behind, turn around, go around*), *triplicar* (*triple*), *ulcerar* (*ulcerate*), *unificar* (*unify, standardize, become unified*), *uniformar* (*standardize, uniformize, make wear a uniform, uniform*), *urdir* (*warp, plot, contrive*), *vaciar* (*deplete, drain, dump, empty, evacuate, flush, void*), *valorear* (*increase the value of, valorize*), *vaporizar* (*evaporate, vaporize*), *variar* (*vary, change, modify*), *velicar* (*lance, open*), *vendar* (*bandage, blindfold*), *ventilar* (*air*), *verdecer* (*green*), *versificar* (*versify*), *verter* (*dump, empty, pour, shed, spill, stream*), *vibrar* (*jar, oscillate, pulsate, quiver, throb, vibrate*), *vigorizar* (*energize, invigorate*), *vitricar* (*vitricify*), *vivificar* (*enliven, quicken*), *vocalizar* (*vocalize*), *volatilizar* (*volatilize*), *volcar* (*capsize, dump, flip, overturn, tip, topple, upset*), *volver* (*return, revert, turn, turn over*), *vulgarizar* (*coarsen*), *yodurar* (*iodize*), *zurcir* (*mend, darn*), *zurrar* (*spank, tan*)

- **Complements:** a (*to*), aún que (*yet*), a fin de (*so that*), a fin de que (*so that*), a menos que (*unless*), antes de (*before, previous to, prior*), antes de que (*before, previous to, prior*), antes que (*before, previous to, prior*), así que (*as soon as*), aunque (*although*), como (*as*), con tal de (*provided that*), con tal que (*that*), cuando (*when*), de manera que (*so that*), de modo que (*so that*), desde (*since*), desde que (*since*), después de (*after*), después de que (*after*), después que (*after*), donde (*where*), en caso de (*provided that*), en caso de que (*provided that*), en cuanto (*as soon as*), en el momento que (*just*), hasta (*until*), hasta que (*until*), lo mismo que (*just as*), luego que (*as soon as*), mientras (*as, while*), mientras que (*as, while*), para que (*so that, so, for*), pero (*but*), porque (*because*), puesto que (*as*), que (*than, that*), sin (*without*), sin que (*without*), sino (*but rather, but*), tan pronto como (*as soon as*), todavía (*yet*), y (*and*)
- **Directional Prepositions:** a (*at, by, into, out at, through, to*), abajo (*down*), arriba de (*above*), bajo (*below, beneath, under*), contra (*against, up against*), de (*about, by, from, in, of, out*), debajo de (*under, underneath*), en (*about, across, along, at, by, in, into, on, over, upon, within*), encima (*above, on, over*), encima de (*above, on, over*), encontra de (*against*), hacia (*onto, toward, towards, unto, pending, till, until, unto, up until*), hacia (*toward*), hacia abajo (*down*), hacia adentro (*into*)

- **Direction Verbs:** abandonar (*desert, leave*), abdicar (*abdicate*), acercarse (*come*), acudir (*go*), adelantar (*advance*), adelantarse (*advance*), advenir (*arrive, come*), alejarse (*recede*), alzar (*rise*), alzarse (*rise*), apartarse (*recede*), aproximarse (*come*), arrimarse (*come*), ascender (*ascend*), ascender (*ascend, rise*), atravesar (*cross*), avanzar (*advance*), avanzar (*advance, move forward*), bajar (*descend, descend to, fall*), buscar (*chase*), caer (*fall*), caerse (*fall, tumble*), chapuzar (*plunge*), cruzar (*cross*), dejar (*leave*), descender (*fall*), desertar (*desert*), desistir (*recede*), dirigirse (*go*), elevar (*rise*), elevarse (*ascend, climb, rise*), entrar (*enter*), entrarse (*enter*), escalar (*climb*), escapar (*escape, flee*), escaparse (*escape, flee*), evadirse (*escape, flee*), fracasar (*fall*), fugarse (*escape, flee*), graduar (*graduate*), huir (*escape, flee*), huirse (*flee*), hundir (*plunge*), hundirse (*plunge*), internarse (*advance*), introducir (*enter*), introducirse (*enter*), ir (*depart, go, leave*), irse (*depart, escape, exit, go, leave*), largarse (*escape, flee, leave*), levantar (*rise*), levantarse (*rise*), llegar (*arrive, come*), marcharse (*depart, go, leave*), partir (*depart, leave*), perseguir (*chase, pursue, tail*), perseguir (*follow, pursue*), persignar (*cross*), plagar (*infest*), posar (*alight*), precipitarse (*plunge*), promover (*advance*), regresar (*return*), responder (*return*), resucitar (*rise*), retirarse (*leave, recede*), retornar (*return*), retroceder (*recede*), revertir (*revert*), rodar (*tumble*), rular (*tumble*), salir (*depart, exit, go out, leave*), santiguar (*cross*), seguir (*track, trail*), seguir (*follow, pursue, track, trail*), subir (*ascend, climb, rise*), sumergir (*plunge*), sumergirse (*plunge*), tornar (*return*), trepar (*climb*), vacar (*va-*

cate), vacar (*vacate*), venir (*come, go*), voltear (*tumble*), volver (*return*), zambullir (*plunge*), zambullirse (*plunge*), zampar (*enter*)

- **Functional Determiners:** algún (*some*), alguna (*some*), algunas (*some*), alguno (*some*), algunos (*some*), cada (*each*), cuál (*which*), cuáles (*which*), cuánta (*how many, how much*), cuántas (*how many, how much*), cuánto (*how many, how much*), cuántos (*how many, how much*), el (*the*), esa (*that*), esas (*those*), ese (*that*), esos (*those*), esta (*this*), estas (*these*), este (*this*), estos (*these*), l (*the*), la (*the*), las (*the*), los (*them*), mucha (*many, much*), muchas (*many, much*), mucho (*many, much*), muchos (*many, much*), ningún (*no, any*), ningunas (*no, any*), ningunos (*no, any*), nuestra (*our*), nuestras (*our*), nuestro (*our*), nuestros (*our*), otra (*other, another*), otras (*other, another*), otro (*other, another*), otros (*other, another*), su (*its, his, her, their, your*), suya (*its, his, her, hers, their, theirs, your, yours*), suyas (*its, his, her, hers, their, theirs, your, yours*), suyo (*its, his, her, hers, their, theirs, your, yours*), suyos (*its, his, her, hers, their, theirs, your, yours*), toda (*all*), todas (*all*), todo (*all*), todos (*all*), tuya (*your, yours*), tuyas (*your, yours*), tuyo (*your, yours*), tuyos (*your, yours*), un (*one*), una (*one*), unos (*some*), unas (*some*), varias (*several*), varios (*several*)
- **Functional Nouns:** él (*he, him, it*), ésa (*that*), ésas (*those*), ése (*that*), éso (*that*), ésos (*those*), ésta (*this*), éstas (*these*), éste (*this*), ésto (*this*), éstos (*these*), alguien (*someone*), algún (*some*), alguna (*some*), algunas (*some*),

alguno (*some*), algunos (*some*), cada (*each*), cuál (*which*), cuáles (*which*),
 cuánta (*how many, how much*), cuántas (*how many, how much*), cuánto
 (*how many, how much*), cuántos (*how many, how much*), la (*the*), las (*the*),
 le (*her, him, it, you*), les (*them, you*), lo (*him, it, you*), los (*them*), me
 (*me, myself*), mí (*me*), mío (*mine*), míos (*mine*), mía (*mine*), mías (*mine*),
 mucha (*many, much*), muchas (*many, much*), mucho (*many, much*), mu-
 chos (*many, much*), nadie (*noone*), ningún (*no, any*), ningunas (*no, any*),
 ninguno (*no, any*), ningunos (*no, any*), nos (*us, ourselves*), nuestra (*our*),
 nuestras (*our*), nuestro (*our*), nuestros (*our*), otra (*other, another*), otras
 (*other, another*), otro (*other, another*), otros (*other, another*), se (*yourself,*
yourselves, themselves, himself, herself, itself, her, him, them), su (*its, his,*
her, their, your), suya (*its, his, her, hers, their, theirs, your, yours*), suyas
 (*its, his, her, hers, their, theirs, your, yours*), suyo (*its, his, her, hers, their,*
theirs, your, yours), suyos (*its, his, her, hers, their, theirs, your, yours*), sí
 mismo (*yourself, yourselves, themselves, himself, herself, itself*), tú (*you*), te
 (*you, yourself*), toda (*all*), todas (*all*), todo (*all*), todos (*all*), tuya (*your,*
yours), tuyas (*your, yours*), tuyo (*your, yours*), tuyos (*your, yours*), uno
 (*one*), una (*one*), unos (*some*), unas (*some*), varias (*several*), varios (*sev-*
eral), yo (*i*)

- Light Verbs:** estar (*be*), ser (*be*), hacer (*make, do*), dar (*give*), tomar (*take*),
 poner (*put*), tener (*have*)

- **Location Verbs:** abarcar (*comprise, contain, include, span*), abrazar (*hug*), adjuntar (*enclose*), albergar (*lodge*), alcanzar (*span*), alojar (*lodge*), amenazar (*overhang*), anexar (*adjoin*), anteponer (*prefix*), apocar (*restrict*), apoyar (*abut, support*), apretar (*touch*), arreglar (*arrange*), arrojar (*sling*), asistir (*attend*), atravesar (*cross, traverse*), bordear (*flank, rim*), cercar (*circle, enclose, fence, ring, surround*), circundar (*circle, ring, surround*), circunscribir (*bound*), colgar (*overhang*), colindar (*adjoin*), colocar (*place, position*), conectar (*bridge*), confinar (*bound, confine*), confluir (*converge*), consistir (*consist*), constar (*consist*), constatar (*consist*), contener (*contain*), cruzar (*cross, intersect, meet*), cubrir (*cover*), cursar (*frequent*), delimitar (*bracket, delimit*), depositar (*deposit*), dirigir (*head*), doblar (*corner*), dominar (*dominate*), encabezar (*head*), encerrar (*contain, enclose, include*), encuadrar (*frame*), enterrar (*bury*), esparrancarse (*straddle*), estacionar (*park*), estar (*be*), evitar (*miss*), fallar (*miss*), fijar (*set*), golpear (*hit*), guardar (*stash, stow*), hospedar (*lodge*), hundir (*immerse*), incluir (*enclose, include*), instalar (*install*), intersecarse (*intersect*), juntar (*meet*), lanzar (*sling*), limitar (*limit*), lindar (*abut, border, line*), llenar (*fill*), localizar (*locate*), lograr (*succeed*), meter (*stow*), montar (*mount, straddle*), obscurecer (*shade*), omitir (*miss*), orillar (*border*), pegar (*hit*), perder (*miss*), perseguir (*follow*), pertenecer (*belong, pertain*), poner (*arrange, locate, place, position, put, set*), poseer (*have, own, possess*), preceder (*antecede, precede*), rebosar (*overflow*), recorrer (*traverse*), rellenar (*fill*), rematar (*top*), remontar (*mount*), residir

(*lodge*), reunir (*meet*), reñir (*surround*), rodear (*encircle, encompass, rim, surround*), seguir (*follow*), sitiar (*beseige, siege, locate, place, position, situate*), sobreponer (*superimpose*), sobresalir (*overhang*), sostener (*uphold*), subir (*mount*), suceder (*succeed*), sumergir (*immerse*), tapar (*cover*), tener (*have, possess*), tocar (*ring, touch*), ubicar (*locate, situate*), yuxtaponer (*juxtapose*)

- **Modal Verbs:** deber (*must*), haber (*have*), poder (*can*)
- **Motion Verbs:** abalanzar (*rush*), abalanzarse (*rush*), abarrotar (*lumber*), abombar (*pad*), abovedar (*vault*), acelerar (*hurry*), acelerarse (*hasten, hurry, speed*), acocear (*kick*), acolchar (*pad*), acortar (*cut*), acuciar (*hurry*), acunar (*rock*), agitar (*flap*), agitarse (*bob, flap, stir, wave*), agrupar (*clump*), agruparse (*clump*), alargarse (*stretch*), aletear (*flap, flutter*), amblar (*amble*), anadear (*waddle*), andar (*amble, pad, stride, walk*), apresurar (*hurry*), apresurarse (*hasten, hurry, speed*), arquearse (*bow*), arrastrar (*crawl, creep, slither*), arrastrarse (*crawl, creep, slither*), arremolinar (*eddy, swirl*), arrojar (*hurtle*), arrojarse (*hurtle*), aserrar (*lumber*), atacar (*charge*), atajar (*cut*), atreverse (*venture*), atropar (*troop*), atroparse (*troop*), avanzar lentamente (*crawl*), aventurarse (*venture*), bailar (*dance, prance*), bajar (*drop, tumble*), balancear (*rock*), balancearse (*bob, rock, teeter*), bambolearse (*reel, totter*), barajar (*shuffle*), barajarse (*shuffle*), barrenar (*scuttle*), batir (*flap*), barrer (*sweep*), bolear (*bowl*), botar (*bounce, bound*), boyar (*float*), bregar

(toil), brincar (*bound, gambol, hop, jump, leap, skip*), cabriolar (*prance*),
 caer (*tumble*), caminar (*trudge, walk*), cargar (*charge*), cercenar (*stump*),
 chapotear (*wade*), chocar (*carom*), chochea (*dodder*), chutar (*kick*), cocear
 (*kick*), cojear (*hobble, hop, limp*), columpiarse (*sway*), contonear (*swagger,*
waddle), contonearse (*swagger, sway, waddle*), contorcerse (*writhe*), contorn-
 earse (*wiggle*), correr (*jog, race, run, scoot, scramble, scuttle*), cortar (*lum-*
ber, stump), costear (*coast*), crispar (*twitch*), culebrear (*wiggle*), cumplir
 (*turn*), danzar (*dance*), deambular (*mosey, perambulate, shamble*), debatirse
 (*writhe*), derrumbarse (*tumble*), descaminar (*go astray, go the wrong way*),
 descaminarse (*go astray, go the wrong way*), desfilar (*parade*), desgarrar
 (*tear*), desgarrarse (*tear*), deslizar (*slide*), deslizarse (*creep, glide, scoot,*
slide, slither), despedir (*bounce*), desplomarse (*tumble*), desviar (*digress,*
stray, wander), desviarse (*digress, stray, wander*), devanar (*reel*), doblar
 (*turn*), echar (*bounce, dash*), echarse (*bounce, dash*), elevarse (*climb*), embe-
 stir (*charge*), embocar (*put through*), embolar (*tip a bull's horn with balls*),
 encabritarse (*prance*), encorvarse (*slouch*), escabullirse (*scurry, skeddadle,*
slide, slink, sneak), enfocar (*zoom*), ensancharse (*stretch*), escalar (*climb*),
 escaparse (*scamper*), escurrirse (*glide, slide, slink*), esforzarse (*buck*), es-
 trellar (*dash*), estrellarse (*dash*), estremecer (*rock*), estremecerse (*quake,*
quiver, rock, tremble), exhibirse (*parade*), extenderse (*straggle, sweep*), ex-
 traviarse (*straggle, stray*), flotar (*drift, float, hover, wave*), flotarse (*drift,*
float), fluctuar (*oscillate*), fluir (*run*), fluye (*run*), forrar (*pad*), galopar

(gallop), gatear (*crawl, creep*), girar (*gyrate, revolve, rotate, stir, swirl, turn*), girarse (*revolve, rotate*), golpear (*dash*), golpearse (*dash*), hacer girar (*roll*), hacer resbalar (*slide*), hacer rodar (*roll*), hollar (*tramp*), huir (*bolt, run*), inclinarse (*bow, sway*), inducir (*move*), infiltrarse (*hover*), ir (*go*), jacarear (*walk the streets singing and making noise*), jugar (*frolic, gambol, romp*), ladearse (*sway*), lanzar (*bounce, hurtle, jump*), lanzarse (*bounce, dart, dash, hurtle*), largarse (*sashay, scam*), latir (*pulsate, throb*), limar (*file*), limpiar (*sweep*), listar (*streak*), llanear (*coast*), llevarse (*sweep*), luchar (*scramble*), manear (*hobble, hopple*), manosear (*fidget*), marchar (*march*), mecer (*rock*), mecerse (*rock, waft*), menear (*bob, shake, waggle, wiggle*), menearse (*bob, stir, waggle, wiggle*), merodear (*prowl*), meterse (*plunge*), mezclar (*dash, shuffle*), mezclar (*dash, shuffle*), mover (*move, rock*), moverse (*move, rock, stir, travel*), mudarse (*move in*), nadar (*float, swim*), najarse (*beat it*), nifear (*muck about*), noctambular (*wander, wander about at night*), nomadear (*wander*), ondear (*flap, float, flutter, undulate, wave*), ondular (*undulate, wave*), oscilar (*flap, oscillate, sway, waver*), ostentarse (*parade*), palpitar (*flutter, throb*), pasar (*stride*), pasear (*amble, go for a walk, promenade, ramble, saunter, stroll, wander*), pasearse (*mosey, parade, perambulate, promenade, sashay, saunter, stroll*), patalear (*kick*), patear (*kick*), pavonearse (*strut*), pellizcar (*nip*), perderse (*stray*), pernear (*kick*), picar (*mince*), picotear (*nip*), pisar (*pad*), pisotear (*tramp*), planear (*glide*), precipitarse (*hurtle, plunge*), pulsar (*pulsate, throb*), rasgar (*tear*), rasgarse (*tear*),

rayar (*scud, streak*), rebobinar (*reel*), rebotar (*bounce, carom*), rebotarse (*bounce*), recorrer (*roam*), remolinear (*eddy, swirl*), renguear (*hobble*), resbalar (*glide, slide*), retemblar (*sway*), retorcer (*squirm, wriggle*), retorcerse (*squirm, writhe*), retozar (*frolic, gambol, romp*), revolotear (*flit, hover*), revolver (*shuffle*), revolvearse (*scramble, shuffle*), rodar (*roll, rotate, slither*), rondar (*prowl*), rotar (*gyrate, rotate*), rular (*roll, rotate*), sacudir (*jog*), saltar (*bounce, bound, hop, jump, leap, prance, skip, vault*), saltarse (*jump*), serpentear (*meander, ramble, squirm, twist, weave, wind*), subir (*climb, hike*), tambalearse (*reel, stagger, totter, wobble*), temblar (*flutter, quake, quiver, shake, stagger, sway, tremble*), temblequear (*dodder*), tirar (*bowl, dart*), titubear (*reel, stagger*), topar (*carom*), torcer (*twist*), tramar (*weave*), transferir (*move*), transponer (*disappear behind, go around*), traquetear (*joggle*), trasladar (*move*), trepar (*clamber, climb*), trepidar (*quake*), tropezar (*stumble*), trotar (*jog, trot*), vacilar (*falter, joggle, reel, stagger, teeter, vacillate, waver*), vadear (*wade*), vagabundear (*ramble, rove, tramp, wander*), vagar (*meander, ramble, roam, rove, tramp, wander*), versar (*turn*), viajar (*journey, travel, trek*), vibrar (*oscillate, pulsate, quiver, throb, vibrate*), virar (*tack, turn*), volar (*flap, fly*), voltear (*revolve, turn*), volver (*turn*), volverse (*turn*), zangolotar (*jiggle*), zangolotarse (*jiggle*), zapatear (*kick*), zigzaguear (*zigzag*)

- **Negation Words:** no (*no*), ningun (*no, nobody, none, noone*), ninguno (*no, nobody, none, noone*), ninguna (*no, nobody, none, noone*), ningunos (*no, nobody, none, noone*), ningunas (*no, nobody, none, noone*)
- **Obliques:** a (*at, by, into, out at, through, to*), a causa de (*at, because of, for, out*), a diferencia de (*unlike*), a excepción de (*except*), a falta de (*failing*), a lo largo de (*along, alongside, down, throughout*), a pesar de (*despite*), a través de (*across, through*), abajo (*down*), acerca de (*about, concerning*), además de (*besides*), adentro (*in, within*), al lado de (*beside, by, next to*), al otro lado de (*across, beyond, over*), allende de (*over*), alrededor de (*all around, around, about*), ante (*before*), antes de (*before, previous to, prior*), aparte de (*aside from*), arriba de (*above*), bajo (*below, beneath, under*), bajo la dirección de (*under*), causa de (*through, about, at, beside, by, near*), cerca de (*near, close to*), como (*as, for, like*), comparado con (*against*), con (*at, by, in, with*), con respecto a (*as for, regarding*), conducto de (*through*), considerado (*considering*), considerado que (*considering that*), contra (*against, up against*), de (*about, by, from, in, of, out*), de menos de (*under*), debajo de (*under, underneath*), delante (*before*), delante de (*ahead of*), dentro (*into*), dentro de (*in, inside, into, under, within*), dentro de los límites de (*within*), desde (*from, since*), después de (*after, since*), detrás (*behind*), detrás de (*after, beyond, in back of, behind*), durante (*during, for, in, pending, throughout*), en (*about, across, along, at, by, in, into, on, over, upon, within*), en caso de

que no (*failing*), en compañía de (*with*), en cuanto a (*as for*), en cumplimiento de (*pursuant*), en favor de (*for*), en frente de (*before*), en lugar de (*instead, instead of*), en medio de (*amid, among*), en todo (*throughout*), encima (*above, on, over*), encima de (*above, on, over*), en contra de (*against*), en contraste con (*against*), entre (*among, amid, between, betwixt and between, in between, in*), estando (*as*), excepto (*but, except, save*), faltando (*failing*), frente a (*opposite, facing, in front of*), fuera de (*off, outside, without*), hacia (*onto, toward, towards, unto, pending, till, until, unto, up until*), hacia abajo (*down*), hacia adentro (*into*), independientemente de (*irrespective*), junto a (*alongside of, by, next to*), junto con (*together with*), lejos de (*away from, off*), más allá (*past*), más allá de (*beyond, past*), más de (*prescindingo de, upward of*), menos (*except*), menos de (*under*), mientras (*pending*), no habiendo (*failing*), no obstante (*despite, in spite of*), para (*by, for, to*), por (*across, around, because, by, for, from, in, on, out, over, per, through*), por causa de (*owing*), por ciento (*per cent*), por encima de (*above, over*), por entre (*through*), por espacio de (*for*), por la vía de (*via*), por medio de (*through*), por todas partes de (*throughout*), porque (*for*), sólo que (*except that*), salvo (*except for, save*), según (*as per, by*), siendo (*as*), sin (*out, without*), sobre (*about, above, across, on, onto, over*), tal como (*as*), tocante a (*concerning, regarding*), tras (*after*), vestido de (*in*), via (*via*), visto que (*considering that*)

- **Psych Verbs:** abominar (*detest, execrate, loathe*), aborrecer (*abhor, detest, hate, loathe*), acariciar (*cherish*), admirar (*admire*), adorar (*adore, love, worship*), agraciar (*favor*), agradar (*like*), agradecer (*appreciate*), aguantar (*stand*), amar (*love*), ambicionar (*covet*), anhelar (*envy*), aparecerse (*appear*), apoyar (*support*), apreciar (*appreciate, cherish, esteem, prize, value*), aprobar (*support*), arrepentirse (*regret*), asistir (*support*), aspirar (*aspire*), atesorar (*treasure*), atreverse (*dare*), ayudar (*support*), buscar (*seek*), carecer (*lack*), codiciar (*envy*), compadecer (*pity*), compadecerse (*pity*), concernir (*involve*), condimentar (*relish*), confiar (*rely, rely on, trust*), conocer (*know*), considerar (*esteem*), consultar (*refer*), decidir (*decide*), declinar (*decline*), denegar (*refuse*), depender (*rely on*), deplorar (*deplore, lament, regret*), desconfiar (*distrust*), desdeñar (*disdain*), desear (*covet, desire, want*), despreciar (*despise*), detestar (*abhor, detest, dislike, loathe*), dirigir (*manage, refer*), disfrutar (*enjoy*), divertir (*enjoy*), divertirse (*enjoy*), doler (*ache, hurt, pain*), dudar (*doubt*), elegir (*elect*), encantar (*love*), envidiar (*envy*), errar (*miss*), escocer (*hurt, itch*), esforzar (*endeavor*), esperar (*expect, hope, wish*), estimar (*appreciate, esteem, prize, respect, value*), exaltar (*exalt*), execrar (*execrate*), extrañar (*miss*), favorecer (*favor*), fiar (*trust*), fiarse (*trust*), flechar (*inspire love*), fruir (*enjoy what has been wished for*), garantizar (*guarantee*), gobernar (*manage*), gozar (*enjoy*), gozarse (*enjoy*), gustar (*enjoy, fancy, like, relish*), herir (*hurt*), idolatrar (*worship*), idolatrar (*idolize, worship*), imaginar (*fancy*), implicar (*implicate*), intentar (*attempt*), intentar

(try), jurar (swear), lamentar (bewail, deplore, lament, mourn, regret, rue),
 lamentarse (lament, mourn), llagar (hurt), llorar (mourn), manejar (man-
 age), mantener (hold, maintain), menospreciar (despise), merecer (deign,
 deserve), mirar (watch), molestar (bother), necesitar (need, want), negar
 (refuse), notar (feel), nutrir (support), odiar (execrate, hate), olvidar (for-
 get), omitir (omit), omitir (miss), padecer (bear), paladear (relish), pare-
 cer (seem), pensar (intend, plan, think), pensar (fancy), perder (miss),
 perseguir (seek), picar (hurt, itch), portar (bear), preferir (favor, prefer),
 preponer (prefer, put before), probar (try), procurar (try), proponer (mean,
 propose), proponer (intend, mean, propose), quejarse (lament), querellarse
 (bewail, lament), querer (intend), querer (cherish, love, want), reconocer
 (recognize), referir (refer), referir (intend, refer), regalar (indulge), rehusar
 (refuse), reiterar (reaffirm), reivindicar (reaffirm), requerir (need), resentir
 (resent), resentirse (resent, take umbrage), respetar (respect), reverenciar
 (revere, venerate), rezar (worship), saber (know), saborear (relish, savor),
 sentir (feel), sentir (regret), soler (tend), soportar (afford, support), sopor-
 tar (bear, support), sospechar (suspect), sostener (hold), sostener (support),
 sufrir (bear), sustentar (support), temer (be afraid to, dread, fear), tolerar
 (stand, tolerate), tratar (try), valorar (appreciate, treasure, value), valorizar
 (appreciate), valuar (appreciate, prize), venerar (revere, venerate, worship),
 vindicar (revenge, vindicate)

Appendix B

DUSTer Universal Rules for English-Spanish

This chapter presents the current set of rules that are used in DUSTer for English-Spanish, using the notation described in Section 3.3.2.

0.A.X [English{2 1} Spanish{1}]

[Verb<1,i> [ModalV<2,Mod,Verb,C:i>]] <-->

[Verb<1,i>]

0.AVar.X [English{2 1} Spanish{1}]

[Verb<1,i> [TenseV<2,Mod,Verb,C:i>]] <-->

[Verb<1,i>]

0.C.X [English{2 1} Spanish{1}]

[Verb<1,i> [Complement<2,Comp,Mod,Child:~S,C:i>]] <-->

[Verb<1,i>]

1.A.X [English{1} Spanish{1 2 3}]

[PsychV<1,i,CatVar:V_N,Verb,Child:~Obj>] <-->

[LightVB<1,Verb,C:i> [Oblique<2,Pred,Prep,C:i> [Noun<3,i,PObj>]]]

1.B.X [English{2 1 3} Spanish{2 1 3 4 5}]

[PsychV<1,i,CatVar:V_N,Verb> [Noun<2,j,Subj>] [Noun<3,k,Obj>]] <-->

[LightVB<1,Verb,C:i> [Noun<2,j,Subj>] [Noun<3,i,Obj>]

[Oblique<4,Pred,Prep,C:i> [Noun<5,k,PObj>]]]

1.BVar.X [English{1} Spanish{1 2 3}]

[PsychV<1,i,CatVar:V_N,Verb>] <-->

[LightVB<1,Verb,C:i> [FuncN<2,Obj,Noun,C:i>] [Noun<3,i,Obj>]]

1.C.X [English{1} Spanish{1 3 2}]

[MotionV<1,i,CatVar:V_N,Verb,Child:~Obj>] <-->

[LightVB<1,Verb,C:i> [Noun<2,i,Obj> [FuncDet<3,Mod,Det,C:i>]]]

1.D.X [English{2 1 4 3} Spanish{2 1}]

[Verb<1,C:i> [LightVB<2,i,Verb>] [Noun<3,j,CatVar:N_V,Obj>

[FuncDet<4,Mod,Det,C:j>]]] <-->

[Verb<1,j> [LightVB<2,i,Verb>]]

1.HVar1A.X [English{1 2 3 4} Arabic{1 2} Chinese{1 2} Spanish{1 2}]

[LightVB<1,Verb,C:i> [LocationV<2,i,CatVar:V_AJ,Verb>

[Oblique<3,Pred,Prep,C:j> [Noun<4,j,PObj>]]]] <-->

[LocationV<1,i,Verb> [Noun<2,j,Obj>]]

1.HVar2.X [English{1 2} Arabic{1} Chinese{1} Spanish{1}]

[LightVB<2,i,Verb> [Pleonastic<1,Subj,Noun,C:i>]] <-->

[LightVB<1,i,Verb>]

2.A.X [English{1} Spanish{1 2}]

[Verb<1,i>] <-->

[MotionV<1,Verb,C:i> [Verb<2,i,Mod>]]

2.A.XX [English{1} Spanish{1 2}]

[Verb<1,i>] <-->

[AspectV<1,Verb,C:i> [Verb<2,i,Mod>]]

2.B.X [English{1 2} Spanish{2 1 3}]

[LightVB<1,i,Verb> [Verb<2,j,CatVar:V_AJ>]] <-->

[MotionV<1,i,Verb> [FuncN<2,Obj,Noun,C:i>] [Verb<3,j,Mod>]]

2.B.XX [English{1 2} Spanish{2 1 3}]

[LightVB<1,i,Verb> [Verb<2,j,CatVar:V_AJ>]] <-->

[AspectV<1,i,Verb> [FuncN<2,Obj,Noun,C:i>] [Verb<3,j,Mod>]]

3.A.X [English{1 2} Spanish{1 2}]

[MotionV<1,i,Verb> [DirectionP<2,j,Pred,Prep>]] <-->

[DirectionV<1,j,Verb> [MotionV<2,i,Verb,Mod>]]

3.A.XX [English{1 2} Spanish{1 2}]

[MotionV<1,i,Verb> [DirectionP<2,j,Pred,Prep>]] <-->

[MotionV<1,j,Verb> [DirectionV<2,i,Verb,Mod>]]

4.A.X [English{1 2} Spanish{1 2}]

[Verb<1,i> [Adv<2,j,CatVar:AV_V,Mod>]] <-->

[Verb<1,j> [Verb<2,i,Obj>]]

5.A.X [English{1 2 3 4} Spanish{1 2}]

[LightVB<1,Verb,C:j> [Adj<2,j,CatVar:AJ_V,Mod>]

[Oblique<3,Pred,Prep,C:m> [Noun<4,m,PObj>]]] <-->

[PsychV<1,j,Verb> [Noun<2,m,Obj>]]

5.A.XX [English{2 1 3} Arabic{2 1 3} Spanish{2 1 3}]

[LightVB<1,Verb,C:j> [Noun<2,i,Subj>]

[Adj<3,j,CatVar:AJ_V,Mod>]] <-->
 [PsychV<1,j,Verb> [Noun<2,Subj>] [Noun<3,i,Obj>]]

 5.BVar.X [English{2 1 3} Spanish{2 3 1 4}]
 [PsychV<1,j,Verb> [Noun<2,i,Subj>] [Noun<3,k,Obj>]] <-->
 [PsychV<1,j,Verb> [Oblique<2,Pred,Prep,C:i>
 [Noun<3,i,PObj>]] [Noun<4,k,Subj>]]

 5.BVar.XX [English{2 1} Spanish{2 3 1}]
 [PsychV<1,j,Verb> [Noun<2,i,Subj>]] <-->
 [PsychV<1,j,Verb> [Oblique<2,Pred,Prep,C:i> [Noun<3,i,PObj>]]]

 5.BVar.XXX [English{2 1 3} Spanish{2 1 3 4}]
 [PsychV<1,j,Verb> [Noun<2,i,Subj>] [Noun<3,k,Obj>]] <-->
 [PsychV<1,j,Verb> [Noun<2,k,Subj>] [Oblique<3,Pred,Prep,C:i>
 [Noun<4,i,PObj>]]]

 5.BVar.XXXX [English{2 1} Spanish{1 2 3}]
 [PsychV<1,j,Verb> [Noun<2,i,Subj>]] <-->
 [PsychV<1,j,Verb> [Oblique<2,Pred,Prep,C:i> [Noun<3,i,PObj>]]]

 6.A.X [English{1 2 3} Spanish{1 2}]

[MotionV<1,j,Verb> [Oblique<2,Pred,Prep,C:k>

[Noun<3,k,CatVar:N_V,PObj>]]] <-->

[MotionV<1,j,Verb> [Verb<2,k,Mod>]]

6.A.XX [English{1 2 3} Spanish{1 2}]

[PsychV<1,j,Verb> [Oblique<2,Pred,Prep,C:k>

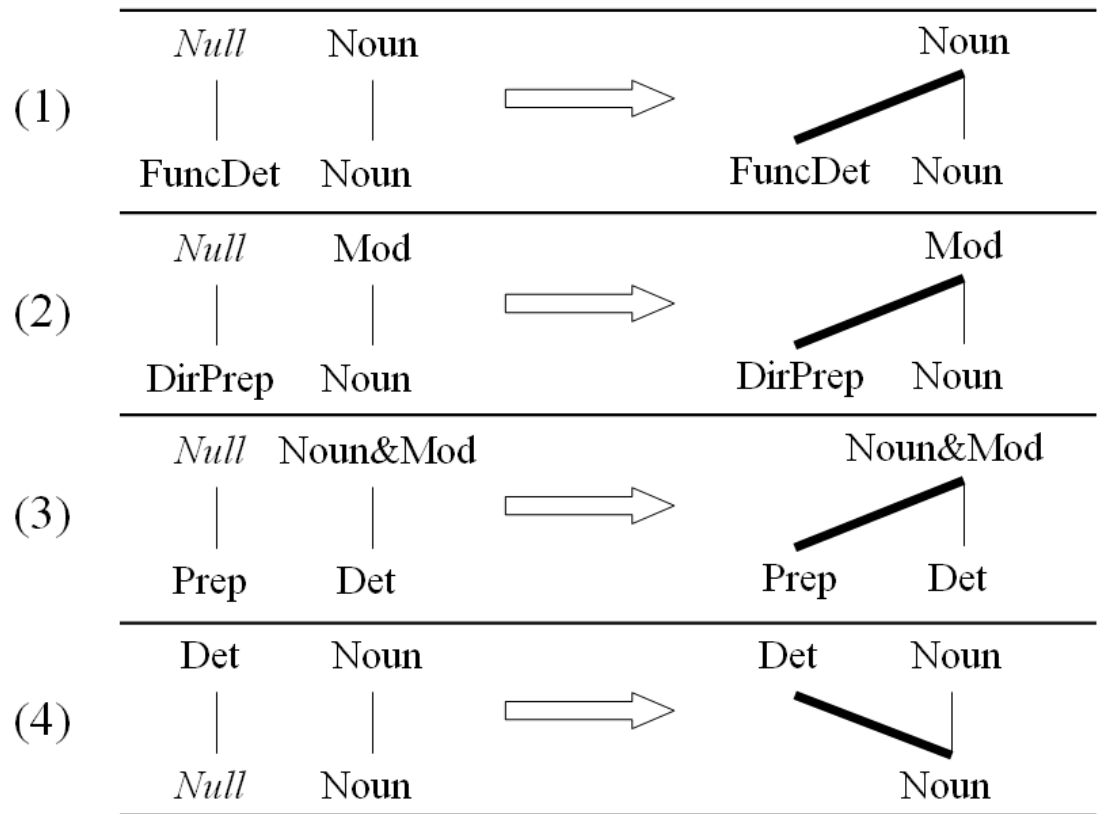
[Noun<3,k,CatVar:N_V,PObj>]]] <-->

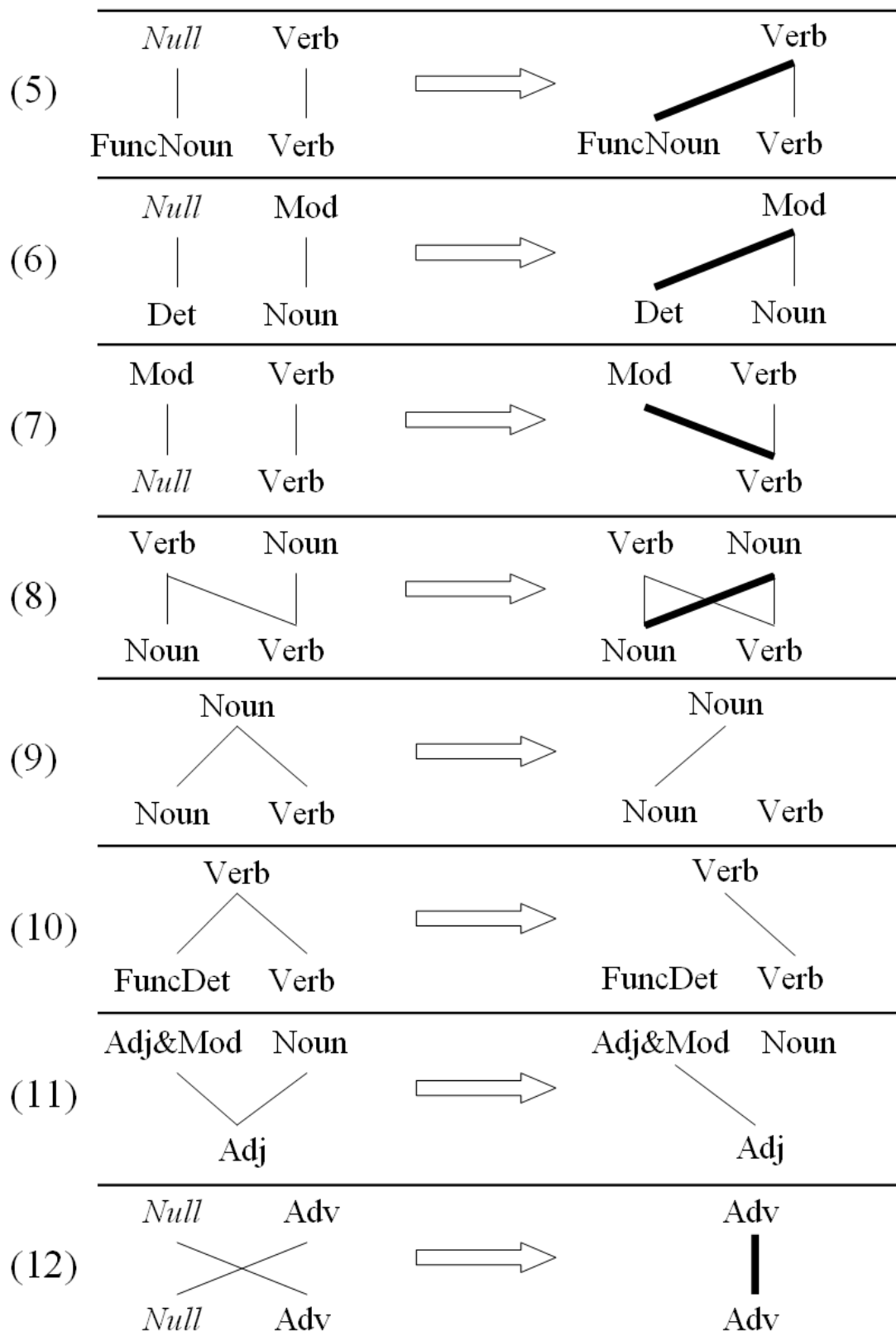
[PsychV<1,j,Verb> [Verb<2,k,Mod>]]

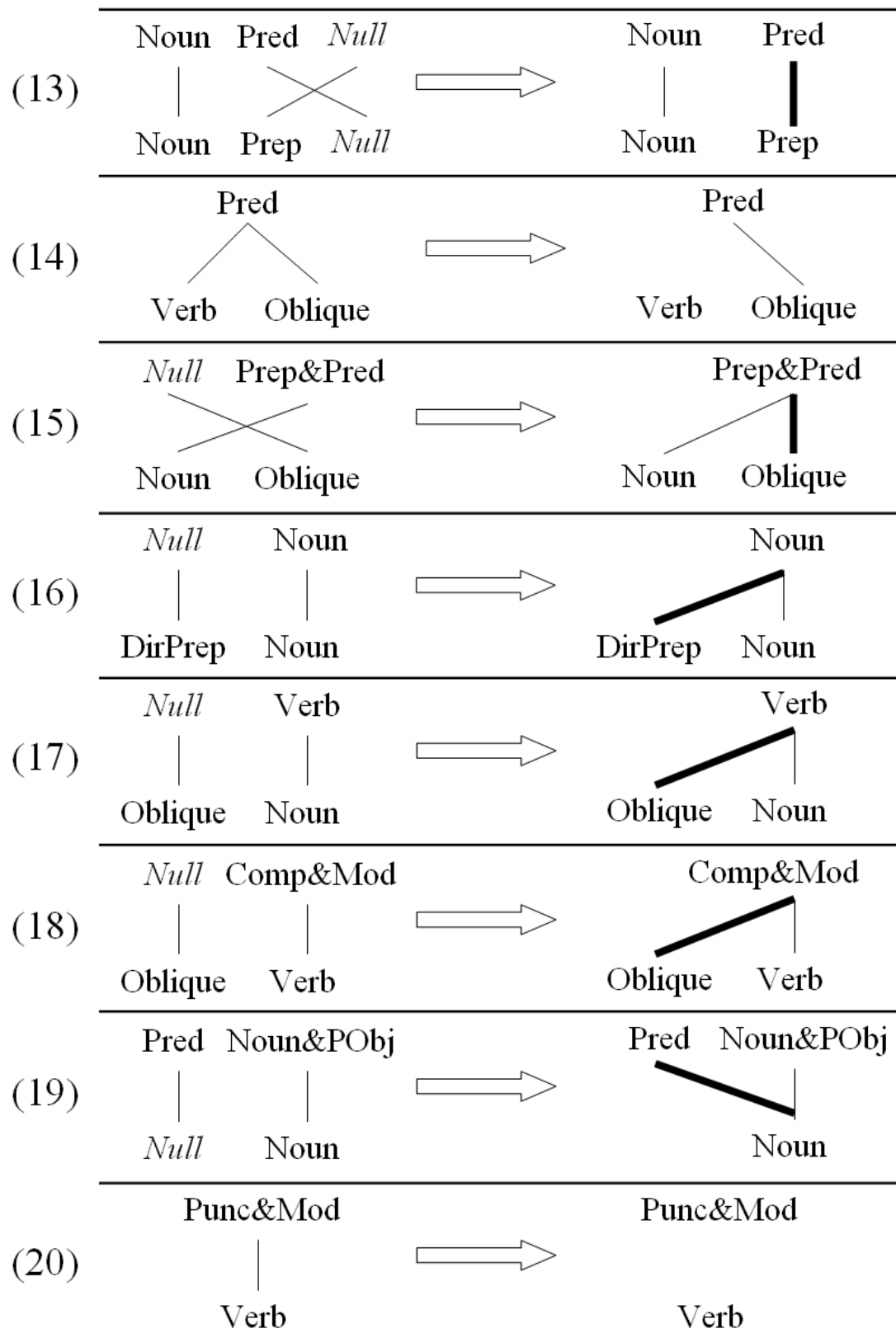
Appendix C

ALP Rules (on English-Spanish)

This chapter lists the first 20 transformation rules that are discovered by ALP on English-Spanish data, using the graphical representation described in Section 4.3.2.




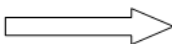
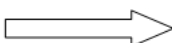
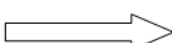



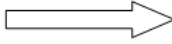
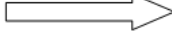







Appendix D

ALP Rules (on English-Chinese)

This chapter lists the first 20 transformation rules that are discovered by ALP on English-Chinese data, using the graphical representation described in Section 4.3.2.

(1)	Mod		Mod
	Particle		Particle
(2)	Complement		Complement
	Punc		Punc
(3)	Pred		Pred
	Noun		Noun
(4)	Complement		Complement
	Verb		Verb

(5)	Punc&Mod		Punc&Mod
	Noun		Noun
(6)	LightVB		LightVB
	Adv		Adv
(7)	Det		Det
	Prep		Prep
(8)	FuncDet		FuncDet
	Verb		Verb
(9)	Pred		Pred
	Particle		Particle
(10)	Det		Det
	Punc		Punc
(11)	Noun Verb		Noun Verb
	Noun		Noun
(12)	Verb		Verb
	Oblique		Oblique

(13)	<div>Complement</div> <div> </div> <div>Noun</div>	→	<div>Complement</div> <div>Noun</div>
(14)	<div>Noun</div> <div> <div> <div>Noun</div> <div>Prep</div> </div> </div>	→	<div>Noun</div> <div> <div> <div>Noun</div> <div>Prep</div> </div> </div>
(15)	<div>UNK&Mod</div> <div> </div> <div>Verb</div>	→	<div>UNK&Mod</div> <div>Verb</div>
(16)	<div>Verb</div> <div> </div> <div>Particle</div>	→	<div>Verb</div> <div>Particle</div>
(17)	<div>Comp</div> <div> </div> <div>Adv</div>	→	<div>Comp</div> <div>Adv</div>
(18)	<div>Oblique</div> <div> </div> <div>Punc</div>	→	<div>Oblique</div> <div>Punc</div>
(19)	<div>Noun</div> <div> </div> <div>Det</div>	→	<div>Noun</div> <div>Det</div>
(20)	<div>LightVB</div> <div> </div> <div>Noun</div>	→	<div>LightVB</div> <div>Noun</div>

BIBLIOGRAPHY

Steven Abney, Robert E. Schapire, and Yoram Singer. Boosting applied to tagging and PP attachment. In *Proceedings of 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC'99)*, pages 38–45, University of Maryland, College Park, MD, June 21-22 1999.

Lars Ahrenberg, Mikael Andersson, and Magnus Merkel. A simple hybrid aligner for generating lexical correspondences in parallel texts. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL/COLING'98)*, Montreal, Quebec, Canada, August 10-14 1998.

Lars Ahrenberg, Magnus Merkel, Anna Segvall Hein, and Jörg Tiedemann. Evaluation of word alignment systems. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC'00)*, volume III, pages 1255–1261, Athens, Greece, May 31-June 2 2000.

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John D. Lafferty, I. Dan Melamed, David Purdy, Franz J. Och, Noah A. Smith, and David Yarowsky.

- Statistical machine translation. Technical report, Johns Hopkins University, 1999. Final Report, WS99 Workshop.
- Kamal M. Ali and Michael J. Pazzani. Error reduction through learning multiple descriptions. *Machine Learning*, 24(3):173–202, 1996.
- Hiyan Alshawi, Shona Douglas, and Srinivas Bangalore. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60, 2000.
- E. L. Antworth. *PC-KIMMO: A Two-Level Processor for Morphological Analysis*. Dallas Summer Institute of Linguistics, 1990.
- Necip F. Ayan, Bonnie J. Dorr, and Nizar Habash. Multi-Align: Combining linguistic and statistical techniques to improve alignments for adaptable MT. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA’04)*, pages 17–26, Georgetown University, Washington DC, September 28–October 2 2004.
- Srinivas Bangalore and Giuseppe Riccardi. A finite-state approach to machine translation. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL’01)*, Carnegie Mellon University, Pittsburgh, PA, June 2-7 2001.
- Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the As-*

- sociation for Computational Linguistics and 10th Conference of the European Chapter (ACL/EACL'01)*, pages 26–33, Toulouse, France, July 9-11 2001.
- Stephen D. Bay. Combining nearest neighbor classifiers through multiple feature subsets. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, pages 37–45, Madison, Wisconsin, July 24-27 1998.
- Adam L. Berger, Stephan A. Della Pietra, and Vincent J. Della-Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1), 1996.
- J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. John Wiley, 1994.
- Indrajit Bhattacharya, Lise Getoor, and Yoshua Bengio. Unsupervised sense disambiguation using bilingual probabilistic models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, Barcelona, Spain, July 21-26 2004.
- Leo Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.
- Eric Brill. *A Corpus-based Approach to Language Learning*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1993.
- Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.

Eric Brill. Learning to parse with transformations. In *Recent Advances in Parsing Technology*. Kluwer Academic Publishers, 1996.

Eric Brill and Philip Resnik. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, August 5-9 1994.

Eric Brill and Jun Wu. Classifier combination for improved lexical disambiguation. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL/COLING'98)*, pages 191–195, Montreal, Quebec, Canada, August 10-14 1998.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. Word sense disambiguation using statistical methods. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL'91)*, pages 264–270, University of California, Berkeley, California, June 18-21 1991a.

Peter F. Brown, Jennifer C. Lai, and Robert L. Mercer. Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL'91)*, pages 169–176, University of California, Berkeley, California, June 18-21 1991b.

Peter F. Brown, Stephan A. Della Pietra, and Robert L. Mercer. The mathe-

- matics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.
- Chris Callison-Burch, David Talbot, and Miles Osborne. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, Barcelona, Spain, July 21-26 2004.
- Jaime Carbonell, Katharina Probst, Erik Peterson, Christian Monson, Alon Lavie, Ralf Brown, and Lori Levin. Automatic rule learning for resource-limited MT. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas (AMTA'02)*, Tiburon, CA, October 8-12 2002.
- Xavier Carreras, Lluís Marquez, and Lluís Padro. Named entity extraction using adaboost. In *Proceedings of the 6th Workshop on Computational Language Learning (CoNLL'02)*, Taipei, Taiwan, August 31-September 1 2002.
- John Carroll, Ted Briscoe, and Antonio Sanfilippo. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC'98)*, pages 447–454, Granada, Spain, May 28-30 1998.
- R. Catizone, G. Russell, and S. Warwick. Deriving translation data from bilingual texts. In *Proceedings of the First International Lexical Acquisition Workshop*, Detroit, MI, 1989.

- J. S. Chang and M. H. C. Chen. Using partial aligned parallel text and part-of-speech information in word alignment. In *Proceedings of the 1st Conference of the Association for Machine Translation in the Americas (AMTA'94)*, pages 16–23, Montreal, Quebec, Canada, October 5-8 1994.
- J. S. Chang, J. N. Chen, H. H. Sheng, and S. J. Ker. Combining machine readable lexical resources and bilingual corpora for broad word sense disambiguation. In *Proceedings of the 2nd Conference of the Association for Machine Translation in the Americas (AMTA'96)*, Columbia, MD, October 2-5 1996.
- Stanley F. Chen. Aligning sentences in bilingual corpora using lexical information. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL'93)*, Ohio State University, Columbus, Ohio, June 22-26 1993.
- Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.
- Colin Cherry and Dekang Lin. A probability model to improve word alignment. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, pages 88–95, Sapporo, Japan, July 7-12 2003.
- David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Com-*

putational Linguistics (ACL'05), University of Michigan, Ann Arbor, Michigan, June 25-30 2005.

Kenneth W. Church and W. A. Gale. Concordances for parallel text. In *Proceedings of the Seventh Annual Conference of the UW Centre for the New OED and Text Research*, pages 40–62, St. Catherine's College, Oxford, England, 1991.

Kennett W. Church. Char_Align: A program for aligning parallel texts at the character level. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pages 1–8, Ohio State University, Columbus, Ohio, June 22-26 1993.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*, pages 1–8, University of Pennsylvania, Philadelphia, PA, July 6-7 2002.

Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC'99)*, University of Maryland, College Park, MD, June 21-22 1999.

Micheal Collins. Three generative lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational*

Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97), Madrid, Spain, July 7-12 1997.

James R. Curran. Ensemble methods for automatic thesaurus extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*, pages 222–229, University of Pennsylvania, Philadelphia, PA, July 6-7 2002.

Ido Dagan and Kenneth W. Church. Termight: Identifying and translating technical terminology. In *Proceedings of the Applied Natural Language Processing Conference (ANLP'94)*, pages 34–40, University of Stuttgart, Germany, October 13-15 1994.

Ido Dagan, Kenneth W. Church, and William A. Gale. Robust bilingual word alignment for machine aided translation. In *Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives*, pages 1–8, 1993.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480, 1972.

Mark W. Davis, Ted E. Dunning, and William C. Ogden. Text alignment in the real world: Improving alignments of noisy translations using common lexical features, string matching , n-gram comparisons. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Lin-*

guistics (EACL'95), University College Dublin, Belfield, Dublin, Ireland, March 27-31 1995.

Herve Dejean, Eric Gaussier, Cyril Goutte, and Kenji Yamada. Reducing parameter space for word alignment. In *Proceedings of the HLT/NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 23–26, Sapporo, Japan, July 7-12 2003.

Yonggang Deng and William Byrne. HMM word and phrase alignment for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'05)*, Vancouver, B.C., Canada, October 6-8 2005.

Mona Diab and Philip Resnik. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, University of Pennsylvania, Philadelphia, PA, July 6-12 2002.

L. R. Dice. Measures of the amount of ecologic association between species. *Journal of Ecology*, 26:297–302, 1945.

Thomas G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.

Thomas G. Dietterich. Ensemble methods in machine learning. In J. Kittler and

- F. Roli, editors, *Multiple Classifier Systems*, volume LNCS Vol. 1857, pages 1–15. Springer, 2000.
- Bonnie J. Dorr. *Machine Translation: A View from the Lexicon*. The MIT Press, Cambridge, MA, 1993.
- Bonnie J. Dorr. Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4):597–633, 1994.
- Bonnie J. Dorr. LCS verb database. Technical Report Online Software Database, University of Maryland, College Park, MD, 2001. http://www.umiacs.umd.edu/~bonnie/LCS_Database_Documentation.html.
- Bonnie J. Dorr, Lisa Pearl, Rebecca Hwa, and Nizar Habash. DUSTER: A method for unraveling cross-language divergences for statistical word-level alignment. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas (AMTA '02)*, Tiburon, CA, October 8-12 2002.
- Bonnie J. Dorr, Necip Fazil Ayan, Nizar Habash, Nitin Madnani, and Rebecca Hwa. Rapid porting of DUSTER to hindi. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3), 2003.
- T. E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.
- Saso Dzeroski and Bernard Zenko. Is combining classifiers better than selecting

the best one? In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*, pages 123–130, Sydney, Australia, July 8-12 2002.

Gerard Escudero, Lluís Marquez, and German Rigau. Boosting applied to word sense disambiguation. In *Proceedings of the 11th European Conference on Machine Learning (ECML'00)*, pages 129–141, Barcelona, Spain, May 30-June 2 2000.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
<http://www.cogsci.princeton.edu/~wn>.

Radu Florian and David Yarowsky. Modeling consensus: classifier combination for word sense disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*, pages 25–32, University of Pennsylvania, Philadelphia, PA, July 6-7 2002.

Radu Florian, John Henderson, and Grace Ngai. Coaxing confidence from an old friend: Probabilistic classifications from transformation rule lists. In *Proceedings of 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC'00)*, Hong Kong, China, October 7-8 2000.

Christian Fluhr. Multilingual information retrieval. In R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, editors, *Survey of the State of the Art in Human Language Technology*, pages 391–405. Cambridge University Press,

- Center for Spoken Language Understanding, Oregon Graduate Institute, 1995.
- Available <http://www.cse.ogi.edu/CSLU/HLTsuryey/ch8node7.html>.
- Yakov Frayman, Bernard F. Rolfe, and Geoffrey I. Webb. Solving regression problems using competitive ensemble models. In *Proceedings of the 15th Australian Joint Conference on Artificial Intelligence (AI'02)*, pages 511–522, 2002.
- Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- P. Fung and K. W. Church. K-vec: A new approach for aligning parallel texts. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING'94)*, pages 1096–1102, Kyoto, Japan, August 5-9 1994.
- William A. Gale and Kennett W. Church. A program for aligning sentences in bilingual corpora. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL'91)*, pages 177–184, University of California, Berkeley, California, June 18-21 1991a.
- William A. Gale and Kennett W. Church. Identifying word correspondences in parallel texts. In *Proceedings of the 4th Speech and Natural Language Workshop*, pages 152–157. DARPA, Morgan Kaufmann, 1991b.
- William A. Gale, Kennett W. Church, and David Yarowsky. Using bilingual materials to develop word sense disambiguation methods. In *Proceedings of the 4th*

International Conference on Theoretical and Methodological Issues in Machine Translation (TMI'92), pages 101–112, Montreal, Quebec, Canada, June 1992.

William A. Gale, Kennett W. Church, and D. Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and Humanities*, 26:415–439, 1993.

Joao Gama. Discriminant trees. In *Proceedings of the 16th International Conference on Machine Learning (ICML'99)*, pages 134–142, Bled, Slovenia, June 27-30 1999.

Ismael García-Varea, Franz Josef Och, Hermann Ney, and Francisco Casacuberta. Improving alignment quality in statistical machine translation using context-dependent maximum entropy models. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, Taipei, Taiwan, August 24-September 1 2002.

E. Gaussier. Flow network models for word alignment and terminology extraction from bilingual corpora. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL/COLING'98)*, pages 444–450, Montreal, Quebec, Canada, August 10-14 1998.

Joydeep Ghosh. Multiclassifier systems: Back to the future. In *Proceedings of*

- the 3rd International Workshop on Multiple Classifier Systems (MCS'02)*, pages 1–15, Cagliari, Italy, June 24-26 2002.
- Daniel Gildea. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, Sapporo, Japan, July 7-12 2003.
- Daniel Gildea. Dependencies vs. constituents for tree-based alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'04)*, Barcelona, Spain, July 25-26 2004.
- Joshua Goodman. Parsing algorithm and metrics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, University of California, Santa Cruz, California, June 24-27 1996.
- Cyril Goutte, Kenji Yamada, and Eric Gaussier. Aligning words using matrix factorisation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, pages 502–509, Barcelona, Spain, July 21-26 2004.
- Nizar Habash and Bonnie J. Dorr. Handling translation divergences: Combining statistical and symbolic techniques in generation-heavy machine translation. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas (AMTA'02)*, Tiburon, CA, October 8-12 2002.
- Nizar Habash and Bonnie J. Dorr. A categorial variation database for english. In

Proceedings of the Human Language Technology and the Meeting of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL'03), pages 96–102, Edmonton, Canada, May 27-June 1 2003.

L. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001, 1990.

Masahiko Haruno and Takefumi Yamazaki. High-performance bilingual text alignment using statistical and dictionary information. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pages 131–138, University of California, Santa Cruz, California, June 24-27 1996.

John C. Henderson and Eric Brill. Bagging and boosting a treebank parser. In *Proceedings of the Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP/NAACL'00)*, pages 34–41, Seattle, Washington, April 29-May 3 2000.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 392–399, University of Pennsylvania, Philadelphia, PA, July 6-12 2002.

Chung hye Han, Benoit Lovie, Martha Palmer, Owen Rambow, Richard Kittredge, Tanya Korelsky, Narin Kim, and Meesook Kim. Handling structural divergences

- and recovering dropped arguments in a korean/english machine translation system. In *Proceedings of the 4th Conference of the Association for Machine Translation in the Americas (AMTA '98)*, Cuernavaca, Mexico, October 10-14 2000.
- P. Isabelle, M. Dymetman, G. Foster, J-M. Jutras, E. Macklovitch, F. Perrault, X. Ren, and M. Simard. Translation analysis and translation automation. In *Proceedings of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI'93)*, pages 15–22, Kyoto, Japan, 1993.
- Abraham Ittycheriah and Salim Roukos. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'05)*, Vancouver, B.C., Canada, October 6-8 2005.
- Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–97, 1991.
- D. Jones and M. Alexa. Towards automatically aligning german compounds with english word groups. In D. Jones and H. Somers, editors, *New Methods in Language Processing*, pages 220–230. UCL Press, London, 1997.
- Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.
- Hiroyuki Kaji, Yuuko Kida, and Yasutsugu Morimoto. Learning translation templates from bilingual text. In *Proceedings of the 14th International Conference*

- on *Computational Linguistics (COLING'92)*, pages 672–678, Nantes, France, August 23-28 1992.
- Laveen Kanal. Patterns in pattern recognition. *IEEE Transactions Information Theory*, IT-20:697–722, 1974.
- Martin Kay and Martin Roscheisen. Text-translation alignment. *Computational Linguistics*, 19(1), 1993.
- Sue J. Ker and Jason S. Chang. A class-based approach to word alignment. *Computational Linguistics*, 23(2):313–343, 1997.
- M. Kitamura and Y. Matsumoto. Extraction of word sequence correspondences in parallel corpora. In *Proceedings of the Fourth Annual Workshop on Very Large Corpora (WVLC'96)*, Copenhagen, 1996.
- Philipp Koehn. Europarl: A multilingual corpus for evaluation of machine translation. Technical report, University of Southern California, 2002. Available at <http://www.iccs.informatics.ed.ac.uk/~pkoehn/publications/europarl.ps>.
- Philipp Koehn. Pharaoh: A beam search decoder for phrase-based statistical machine translation. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas (AMTA'04)*, Georgetown University, Washington DC, September 28-October 2 2004.
- Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and the Meeting of*

the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL'03), Edmonton, Canada, May 27-June 1 2003.

Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 313–321, Tahoe City, California, July 9-12 1995.

A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. MIT Press, Cambridge, MA, 1995.

Philippe Langlais, Michel Simard, and Jean Véronis. Methods and practical issues in evaluating alignment techniques. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL/COLING'98)*, pages 711–717, Montreal, Quebec, Canada, August 10-14 1998.

Benoit Lavoie, Richard Kittredge, Tanya Korelsky, and Owen Rambow. A framework for MT and multilingual NLG systems based on uniform lexico-structural processing. In *Proceedings of the Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP/NAACL'00)*, Seattle, Washington, April 29-May 3 2000.

Geoffrey Leech, R. Barnett, and P. Kahrel. EAGLES recommendations for the syntactic annotation of corpora. Technical report, Department of Linguistics and Modern English Language, Lancaster University, Lancaster, United Kingdom, 1996. Technical Report EAG-TCWG-SASG/1.8, Version of 11th March 1996.

Beth Levin. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, IL, 1993.

Yang Liu, Qun Liu, and Shouxun Lin. Log-linear models for word alignment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, University of Michigan, Ann Arbor, Michigan, June 25-30 2005.

Adam Lopez and Philip Resnik. Improved HMM alignment models for languages with scarce resources. In *Proceedings of the ACL'05 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 83–86, University of Michigan, Ann Arbor, Michigan, June 25-30 2005.

E. Macklovitch and M. L. Hannan. Line 'em up: Advances in alignment technology and their impact on translation support tools. In *Proceedings of the 2nd Conference of the Association for Machine Translation in the Americas (AMTA'96)*, Columbia, MD, October 2-5 1996.

Catherine Macleod, Ralph Grishman, Adam Meyers, Leslie Barrett, and Ruth Reeves. Nomlex: A lexicon of nominalizations. In *Proceedings of the Conference*

of European Association for Lexicography (EURALEX'98), University of Liege, Liege, Belgium, 1998.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*, University of Pennsylvania, Philadelphia, PA, July 6-7 2002.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

L. Marquez, L. Padro, and H. Rodriguez. Improving tagging accuracy by using voting taggers. In *Proceedings of the 2nd Conference on Natural Language Processing and Industrial Applications (NLP+IA/TAL+AI'98)*, Moncton, New Brunswick, Canada, 1998.

L. Marquez, H. Rodriguez, J. Carmona, and J. Montolio. Improving pos tagging using machine learning techniques. In *Proceedings of 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC'99)*, University of Maryland, College Park, MD, June 21-22 1999.

Yuji Matsumoto, Hiroyuki Ishimoto, Takehito Utsuro, and Makoto Nagao. Structural matching of parallel texts. In *Proceedings of the 31st Annual Meeting of*

the Association for Computational Linguistics (ACL'93), Ohio State University, Columbus, Ohio, June 22-26 1993.

Evgeny Matusov, Richard Zens, and Hermann Ney. Symmetric word alignments for statistical machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, pages 219–225, University of Geneva, Switzerland, August 23-27 2004.

A. M. McEnery and M. P. Oakes. Sentence and word alignment in the crater project. In *Proceedings of the EACL/SIGDAT Workshop: From Texts to Tags: Issues on Multilingual Language Analysis*, pages 77–86, Dublin, Ireland, 1995.

I. Dan Melamed. A geometric approach to mapping bitext correspondence. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing (EMNLP'96)*, University of Pennsylvania, Philadelphia, May 17-18 1996a.

I. Dan Melamed. Automatic construction of clean broad-coverage translation lexicons. In *Proceedings of the 2nd Conference of the Association for Machine Translation in the Americas (AMTA'96)*, pages 125–134, Columbia, MD, October 2-5 1996b.

I. Dan Melamed. A portable algorithm for mapping bitext correspondence. In *Proceedings of the 35th Annual Meeting of the Association for Computational*

Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97), Madrid, Spain, July 7-12 1997a.

I. Dan Melamed. A word-to-word model of translational equivalence. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, Madrid, Spain, July 7-12 1997b.

I. Dan Melamed. Automatic discovery of non-compositional compounds in parallel data. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing (EMNLP'97)*, pages 97–108, Brown University, Providence, Rhode Island, August 1-2 1997c.

I. Dan Melamed. Manual annotation of translational equivalence: The blinker project. Technical report, Department of Computer and Information Science, University of Pennsylvania, 1998. IRCS Technical Report #98-07.

I. Dan Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, 2000.

Arul Menezes and Stephen D. Richardson. A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora. In *Proceedings of the Workshop on Data-driven Machine Translation*, pages 39–47, 2001.

Magnus Merkel and Lars Ahrenberg. Evaluating word alignment systems. Techni-

cal report, Linkoping University, Uppsala University and Goteborg University, 1999. PLUG Report.

A. Meyers, R. Yangarber, and R. Grishman. Alignment of shared forests for bilingual corpora. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pages 460–465, Copenhagen, Denmark, August 5-9 1996.

Rada Mihalcea and Ted Pedersen. An evaluation exercise for word alignment. In *Proceedings of the HLT/NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10, Sapporo, Japan, July 7-12 2003.

M. Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine*, 12:34–51, 1991.

Robert C. Moore. Improving IBM word-alignment model 1. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, pages 519–526, Barcelona, Spain, July 21-26 2004.

Robert C. Moore. A discriminative framework for bilingual word alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'05)*, Vancouver, B.C., Canada, October 6-8 2005.

Grace Ngai and Radu Florian. Transformation-based learning in the fast lane. In *Proceedings of the 2nd Meeting of the North American Chapter of the Associ-*

ation for Computational Linguistics (NAACL'01), Carnegie Mellon University, Pittsburgh, PA, June 2-7 2001.

Doug W. Oard and Bonnie J. Dorr. A survey of multilingual text retrieval. Technical report, University of Maryland, Institute for Advanced Computer Studies, April 1996. Technical Report UMIACS-TR-96-19.

Franz J. Och. An efficient method to determine bilingual word classes. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, University of Bergen, Norway, June 8-12 1999.

Franz J. Och. GIZA++: Training of statistical translation models. Technical report, RWTH Aachen, University of Technology, 2000. Available at <http://www-i6.informatik.rwth-aachen.de/~och/software/GIZA++.html>.

Franz J. Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03)*, pages 160–167, Sapporo, Japan, 2003.

Franz J. Och. Google's approach to machine translation. In *Proceedings of the 2005 NIST Machine Translation Evaluation Plan (MTEval 2005)*, 2005.

Franz J. Och and Hermann Ney. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th International Conference*

on Computational Linguistics (COLING'00), Saarbrücken, Germany, July 31-August 4 2000a.

Franz J. Och and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, pages 440–447, Hong Kong, China, October 1-8 2000b.

Franz J. Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 295–302, University of Pennsylvania, Philadelphia, PA, July 6-12 2002.

Franz J. Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):9–51, March 2003.

Franz J. Och and Hans Weber. Improving statistical natural language translation with categories and rules. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL/COLING'98)*, pages 985–989, Montreal, Quebec, Canada, August 10-14 1998.

Franz J. Och, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *Proceedings of 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large*

Corpora (EMNLP/VLC'99), pages 20–28, University of Maryland, College Park, MD, June 21-22 1999.

Franz J. Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology and the Meeting of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL'04)*, pages 161–168, Boston, MA, May 2-7 2004.

David Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 311–318, University of Pennsylvania, Philadelphia, PA, July 6-12 2002.

B. Parmanto, P.W. Munro, and H.R. Doyle. Improving committee diagnosis with resampling techniques. In D.S. Touretzky, M.C. Mozer, and M.E. Hesselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 882–888. MIT Press, Cambridge, MA, 1996.

Ted Pedersen. A simple approach to building ensembles of naive bayesian classifiers

- for word sense disambiguation. In *Proceedings of the Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP/NAACL'00)*, pages 63–69, Seattle, Washington, April 29-May 3 2000.
- M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- Katharina Probst and Ralf Brown. Using similarity scoring to improve the bilingual dictionary for word alignment. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 409–416, University of Pennsylvania, Philadelphia, PA, July 6-12 2002.
- P. Procter. *Longman Dictionary of Contemporary English: Computer Codes for the Definition Space Other than the Subject Field Compositional Translation*. Longman Group LTD, 1983.
- J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
- Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing (EMNLP'96)*, University of Pennsylvania, Philadelphia, May 17-18 1996.
- Philip Resnik. Parallel strands: A preliminary investigation into mining the web for bilingual text. In *Proceedings of the 3rd Conference of the Association for*

Machine Translation in the Americas (AMTA '98), Langhorne, Pennsylvania, October 28-31 1998.

Philip Resnik and I. Dan Melamed. Semi-automatic acquisition of domain-specific translation lexicons. In *Proceedings of the Applied Natural Language Processing Conference (ANLP'97)*, pages 340–347, Washington DC, March 31-April 3 1997.

Philip Resnik and Noah A. Smith. The web as parallel corpus. *Computational Linguistics*, 29(3):349–380, September 2003.

Martin Riedmiller. Advanced supervised learning in multilayer perceptrons - from backpropagation to adaptive learning techniques. *Journal of Computer Standards and Interfaces*, 16, 1994.

Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN'93)*, pages 586–591, San Francisco, California, 1993.

Galina Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781, 1994.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

D. E. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel*

Distributed Processing, Vol. I Foundations, pages 318–362. MIT Press, Cambridge, MA, 1986.

Ken Samuel, Sandra Carberry, and K. Vijay-Shanker. Dialogue act tagging with transformation-based learning. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL/COLING'98)*, Montreal, Quebec, Canada, August 10-14 1998.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2): 197–227, 1990.

Michel Simard and P. Plamondon. Bilingual sentence alignment: Balancing robustness and accuracy. *Machine Translation*, 13(1), 1998.

Michel Simard, G. Foster, and P. Isabelle. Using cognates to align sentences in bilingual corpora. In *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI'92)*, pages 67–81, Montreal, Quebec, Canada, June 1992.

Frank A. Smadja, Kathleen R. McKeown, and Vasileios Hatzivassiloglou. Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, 22(1):1–38, 1996.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. A discriminative matching approach to word alignment. In *Proceedings of the Conference on Empirical Meth-*

ods in Natural Language Processing (EMNLP'05), Vancouver, B.C., Canada, October 6-8 2005.

Jörg Tiedemann. Combining clues for word alignment. In *Proceedings of the 10th Conference of the European Chapter of the ACL (EACL'03)*, pages 339–346., 2003.

Christoph Tillmann. A projection extension algorithm for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'03)*, Sapporo, Japan, July 11-12 2003.

Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga. A DP-based search using monotone alignments in statistical translation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, pages 289–296, Madrid, Spain, July 7-12 1997.

Kai Ming Ting. The characterisation of predictive accuracy and decision combination. In *Proceedings of the 13th International Conference on Machine Learning (ICML'96)*, pages 498–506, Bari, Italy, July 3-6 1996.

Erik F. Tjong Kim Sang, Walter Daelemans, Herve Dejean, Rob Koeling, Yuval Krymolowski, Vasin Punyakanok, and Dan Roth. Applying system combination to base noun phrase identification. In *Proceedings of the 18th International*

Conference on Computational Linguistics (COLING'00), pages 857–863, Saarbrücken, Germany, July 31–August 4 2000.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. Extensions to HMM-based statistical word alignment models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'02)*, pages 87–94, University of Pennsylvania, Philadelphia, PA, July 6-7 2002.

Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3–4):385–404, December 1996a.

Kagan Tumer and Joydeep Ghosh. Estimating the bayes error rate through classifier combining. In *Proceedings of the 13th International Conference on Pattern Recognition (ICPR'96)*, volume 2, pages 695–699, Vienna, Austria, August 1996b.

Hans van Halteren, Jakub Zavrel, and Walter Daelemans. Improving data driven wordclass tagging by system combination. In *ACL 1998, Proceedings of the 36th Conference on Association for Computational Linguistics*, pages 491–497, 1998.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

Ashish Venugopal, Stephan Vogel, and Alex Waibel. Effective phrase translation extraction from alignment models. In *Proceedings of the 41st Annual Meeting of*

the Association for Computational Linguistics (ACL'03), pages 319–326, Sapporo, Japan, 2003.

Jean Véronis. ARCADE – evaluation of parallel text alignment systems. Technical report, Université de Provence & CNRS, France, 1998. Available at <http://www.lpl.univ-aix.fr/projects/arcade/indexen.html>.

Jean Véronis. From the rosetta stone to the information society: A survey of parallel text processing. In Jean Véronis, editor, *Parallel Text Processing: Alignment and Use of Translation Corpora*, chapter 1. Kluwer Academic Publishers, London, September 2000.

Stefan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pages 836–841, Copenhagen, Denmark, August 5-9 1996.

Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. The CMU statistical machine translation system. In *Proceedings of the 9th Machine Translation Summit of the International Association for Machine Translation (MT Summit IX)*, New Orleans, LA, September 23-27 2003.

Ye-Yi Wang. *Grammar Inference and Statistical Machine Translation*. PhD thesis, Carnegie Mellon University, 1998.

Geoffrey I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, 2000.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

Dekai Wu. Aligning a parallel english-chinese corpus statistically with lexical criteria. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94)*, pages 80–87, New Mexico State University, Las Cruces, New Mexico, June 27-30 1994.

Dekai Wu. Grammarless extraction of phrasal translation examples from parallel texts. In *Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI'95)*, volume 2, pages 354–372, Leuven, Belgium, July 5-7 1995.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997.

Dekai Wu, Grace Ngai, Marine Carpuat, Jeppe Larsen, and Yongsheng Yang. Boosting for named entity recognition. In *Proceedings of the 6th Workshop on Computational Language Learning (CoNLL'02)*, Taipei, Taiwan, August 31-September 1 2002.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational*

Linguistics and 10th Conference of the European Chapter (ACL/EACL'01),
Toulouse, France, July 9-11 2001.

Yiming Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, 1999.

David Yarowsky, Grace Ngai, and Richard Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the Human Language Technology Conference (HLT'01)*, pages 109–116, San Diego, California, March 18-21 2001.

Richard Zens, Evgeny Matusov, and Hermann Ney. Improved word alignment using a symmetric lexicon model. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, pages 36–42, University of Geneva, Switzerland, August 23-27 2004.

Hao Zhang and Daniel Gildea. Syntax-based alignment: Supervised or unsupervised? In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, University of Geneva, Switzerland, August 23-27 2004.

Hao Zhang and Daniel Gildea. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, University of Michigan, Ann Arbor, Michigan, June 25-30 2005.

Ying Zhang, Stephan Vogel, and Alex Waibel. Integrated phrase segmentation and alignment algorithm for statistical machine translation. In *Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE'03)*, Beijing, China, October 2003.

Ying Zhang, Stephan Vogel, and Alex Waibel. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, pages 2051–2054, 2004.

Bing Zhao and Stephan Vogel. Word alignment based on bilingual bracketing. In *Proceedings of the HLT/NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, Sapporo, Japan, July 7-12 2003.