ABSTRACT

Title of dissertation: SEARCHING TO TRANSLATE AND TRANSLATING TO SEARCH: WHEN INFORMATION RETRIEVAL MEETS MACHINE TRANSLATION

Ferhan Ture, Doctor of Philosophy, 2013

Dissertation directed by: Professor Jimmy Lin
Department of Computer Science
and College of Information Studies

With the adoption of web services in daily life, people have access to tremendous amounts of information, beyond any human's reading and comprehension capabilities. As a result, search technologies have become a fundamental tool for accessing information. Furthermore, the web contains information in multiple languages, introducing another barrier between people and information. Therefore, search technologies need to handle content written in multiple languages, which requires techniques to account for the linguistic differences. Information Retrieval (IR) is the study of search techniques, in which the task is to find material relevant to a given information need. Cross-Language Information Retrieval (CLIR) is a special case of IR when the search takes place in a multi-lingual collection.

Of course, it is not helpful to retrieve content in languages the user cannot understand. Machine Translation (MT) studies the translation of text from one language into another efficiently (within a reasonable amount of time) and effectively (fluent and retaining the original meaning), which helps people understand what is being written, regardless of the source language.

Putting these together, we observe that search and translation technologies are part of an important user application, calling for a better integration of search (IR) and translation (MT), since these two technologies need to work together to produce high-quality output.

In this dissertation, the main goal is to build better connections between IR and MT, for which we present solutions to two problems: *Searching to translate* explores approximate search techniques for extracting bilingual data from multilingual Wikipedia collections to train better translation models. *Translating to search* explores the integration of a modern statistical MT system into the cross-language search processes. In both cases, our best-performing approach yielded improvements over strong baselines for a variety of language pairs.

Finally, we propose a general architecture, in which various components of IR and MT systems can be connected together into a feedback loop, with potential improvements to both search and translation tasks. We hope that the ideas presented in this dissertation will spur more interest in the integration of search and translation technologies.

SEARCHING TO TRANSLATE AND TRANSLATING TO SEARCH: WHEN INFORMATION RETRIEVAL MEETS MACHINE TRANSLATION

by

Ferhan Ture

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
2013

Dedication

to my parents, my family, and my dearest Elif...

Acknowledgments

I am indebted to all the people who have made this dissertation possible and because of whom my graduate experience has been one that I will cherish forever. I am very lucky to have had great advisors, friends, and family, and would like to express my gratitude with a few words.

First and foremost I'd like to thank my advisor, Jimmy Lin, from whom I could not have asked more. Above all, he has always made sure that I have everything to make me comfortable throughout my Ph.D life. He has played a crucial role in preparing me as a researcher, by introducing me to interesting topics, setting a high bar for standards, and always encouraging me to keep active in the field. He has also taught me well as a software developer, by passing along his industry experiences, involving me in the development of high-quality software. Additionally, I have benefited tremendously from Jimmy as an excellent role model in academia, and he has been a great inspiration during challenging times.

I would also like to thank Doug Oard, for guiding me throughout the process with his great vision and breadth of knowledge, often helping me to see the bigger picture. I feel very special to have such a great expert of Information Retrieval in my committee.

Philip Resnik is one of the reasons I chose this program, and I have learned a lot from him both as a teacher and researcher. His enthusiasm has encouraged me to keep excited about my work and not give up easily. As a pioneer in the field of Machine Translation, Philip's vision has influenced my research in a profound way.

I would also like to thank Mihai Pop for his valuable feedback as one of my proposal committee members and also Hal Daumé, Bonnie Dorr, and Sennur Ulukus for agreeing to serve on my dissertation committee and for sparing their invaluable time reviewing the manuscript.

Without the endless support of my previous advisors in Turkey – Deniz Yuret, Kemal Oflazer, and Esra Erdem – this dissertation would have been a distant dream. I would like to thank them for believing in me during the early years of my career.

These five years wouldn't have been the same if it wasn't for my colleagues in the CLIP Lab. I would like to thank all of them for being part of this awesome research group. I have learned a lot from the weekly seminars, as well as from daily interactions within the lab. I would like to thank Hua, Ke Zhai, Hassan, Tan, Viet-An, Asad, Junhui, Yuening, Amit, and Greg, and especially Vlad, Ke Wu, Chris Dyer, Kristy, Hendra, Zhongqiang, Jags, Tamer, Mossaab, Lidan, and Nima for being such cool friends, colleagues, lab mates, and more.

I would also like to acknowledge help and support from some of the staff members, especially Joe Webster for his extraordinary efforts to make sure that computing resources were ready when needed.

My friends outside the lab deserve a special thank you. Many thanks to Tuğrul, Şimal, Ali Fuad, Bengü, Bedrettin, Enes, Yasin, Salih, Orhan, Gökçe, Edvin, Melis, David, Jose, Pilar, Aslı, Mustafa Dayıoğlu, Mustafa Hatipoğlu, Ömür and Bahadır for their sincere friendship. Hanging out with them has been the best cure to the daily stress of graduate life. I am also thankful to all of my friends in Turkey, who have provided moral support despite the physical distance.

Last, but not least, I am grateful to be part of a wonderful family. My parents took great responsibility and went through many difficulties, without any hesitation, just to make sure that I receive good education. My entire family has always stood by me and guided me through my career. Their love and support brings me peace and comfort.

My dear wife, Elif, is the one who has made this journey enjoyable. She has been a constant motivation for me, and I cannot even imagine how it would've been possible without her endless love and care.

It is impossible to remember all, and I apologize to those I've inadvertently left out. Thank you all!

Table of Contents

List of Figures

Chapter 1: **Introduction**

1.1 Motivation

With the ever-increasing adoption of web services in daily life, people have increasing access to tremendous amounts of information, ranging from product reviews on commercial websites to descriptions of concepts in online encyclopedias. However, the rate at which data is generated exceeds any human's reading and comprehension capabilities. In addition, most of the generated data might be either irrelevant, redundant, or even incomprehensible. As a result, search technologies have become a fundamental tool for accessing information, used by millions everyday. Information Retrieval (IR) is the study of search techniques, in which the task is to find material (e.g., documents) relevant to a given information need (e.g., query).

It is essential that IR approaches can handle large, noisy, and complex data collections. One challenge is to search in collections containing text in multiple languages. This is becoming a very common characteristic of web collections: according to recent statistics, the proportion of Arabic content on the web is increasing nine times faster than English, and only 27% of web users are native English speakers [56]. Therefore, search technologies need to handle content written in multiple languages, which requires some technique to account for the linguistic differences. Cross-Language Information Retrieval (CLIR) is a special case of IR when the search takes place in a multi-lingual collection.

Of course, it is not helpful to retrieve content in languages the user cannot understand. The goal of Machine Translation (MT) is to translate text from one language into another. With the ability to translate text efficiently (within a reasonable amount of time) and effectively (fluent and retaining the original meaning), this technology makes it possible to understand what is being said, regardless of the source language.

Putting all of this together, we observe that search and translation technologies are part of an important user application: A user should be able to search multi-lingual web collections, by expressing the information need in any language, and the relevant content should be presented in any desired language. This application calls for a better integration of search (IR) and translation (MT), since these two technologies need to work together to produce optimal output. Furthermore, improving solutions to either problem will bring better access to information outside the limits of a particular language and population.

Motivated by the opportunity to improve our everyday life, the goal of this dissertation is to build better connections between IR and MT. We first demonstrate that IR techniques can be used to improve MT, and then show the same applies in reverse: MT techniques can be used to improve the state of the art of IR. In order to complete the cycle, we then introduce an architecture that connects the pieces together, and provides a path to a better integration of IR and MT.

1.2 Outline

As discussed above, this dissertation explores two problems as part of the same overall architecture. The first problem focuses on "searching to translate," or how one can use search techniques to improve translation modeling. The second problem focuses on "translating to search," or how one can use the advanced translation modeling approaches to improve the search process. We finally introduce the overall architecture connecting these two problems, and describe how to potentially bring iterative improvements to the existing models in IR and MT.

In Chapter 2, we review related work in the various relevant subjects, such as pairwise similarity, parallel text extraction, query translation in CLIR, and previous attempts to integrate IR and MT. Chapters 3 and 4 describe the two problems introduced above, namely "searching to translate" and "translating to search," respectively. A more thorough description of these two chapters are given below. In Chapter 5, we propose a bootstrapping approach to connect the components of Chapters 3 and 4. Finally, conclusions and future work are presented in Chapter 6.

1.2.1 Searching to Translate

The core of a state-of-the-art MT system is the underlying statistical translation model, which contains parameters that need to be learned via machine learning (ML) algorithms. In the MT and ML literature, it has been repeatedly shown that more training data usually means better models [8], since the learning process involves more examples and can better capture patterns in the model. For MT, the training data needs to be in the form of a *parallel corpus*, or *parallel text*, which is a list of aligned sentence pairs that are mutual translations. Having humans generate this type of data is possible, but time-consuming and expensive. In some cases, we can generate a parallel corpus with little effort from text generated for other purposes. European Union proceedings, which are produced in many European languages, and Al-Jazeera news stories, which are printed in both Arabic and English, are two good examples. However, this is only possible for selected language pairs and in certain domains (typically in formal language), therefore we cannot solely rely on these data resources for building robust, general-purpose translation systems.

On the other hand, IR provides many techniques to efficiently and effectively search in large web collections. These collections potentially contain a lot of parallel text, although it has not been aligned. For example, the same news story is covered by the media in many countries, therefore it is possible that some of the sentences can be aligned throughout all of these sources. Popular books are also translated into many languages, but might not be directly usable for MT since they are not generated sentence by sentence. This type of multilingual data resources are often called *comparable corpora* as opposed to *parallel corpora*, since there is no explicit alignment at the sentence-level, but there are many text portions on the same subject.

If we could extract the parallel text within these comparable corpora, we could use it as MT training data. This is the problem explored in Chapter 3: we present an approach to search for parallel text (i.e., cross-lingual sentence pairs that are translations of each other) in a comparable corpus (i.e., two collections in different languages, which contain documents of similar content). Our two-phase approach consists of two algorithms, described below.

The first algorithm finds similar cross-lingual document pairs in a collection containing documents written in two languages. In this search task, which is called *cross-lingual pairwise similarity*, documents in one language are treated as queries, and the goal is to retrieve similar documents in the other language. For very large collections, translating all queries using an MT system is computationally infeasible within our resources.[1] Hence, our algorithm is based on word-based CLIR translation techniques for efficient translation and locality-sensitive hashing for efficient search. This is a parallelized implementation of a sliding window-based algorithm described earlier [116], which we have modified and optimized for this specific application.

The second algorithm operates on the output of the first, which is a list of similar cross-lingual document pairs, (D_e, D_f). The goal is to generate candidate sentence pairs (s_e, s_f) such that $s_e \in D_e$ and $s_f \in D_f$, and decide if each is "parallel" (i.e., mutual translation) or not. This involves a very large number of decisions, so we introduce a two-step classification approach to balance efficiency and effectiveness. Both algorithms are implemented in the MapReduce programming model for increased scalability and parallelization [33].

As a result, our two-phase approach outputs a parallel corpus, which we evaluate on the task of MT. By adding the extracted data to a baseline MT system, we show that substantial improvements can be obtained for the six language pairs we

[1]With more resources, commercial research groups can afford this type of an approach [141].

experimented with. Furthermore, this approach is essentially *for free*, as opposed to the alternative: time-consuming and expensive human labor.

1.2.2 Translating to Search

In the task of cross-language IR, we are interested in searching for information expressed in a foreign language,[2] which requires translation of the user query into the document language. It is also possible to translate the document text into the query language, but we will assume the former case for simplicity. State-of-the-art CLIR approaches perform translation by either incorporating token-to-token translations in a probabilistic structure, or by treating an MT system as a black box to translate query text. The former approach *preserves ambiguity* of the possible translations of query words, but ignores any local context (i.e., neighbor words in the query) that might be useful when making translation choices. The latter approach produces a translation that is *sensitive to query context*, thanks to MT. However, it provides only one way of translating the query, which might cause lower recall for retrieval tasks.

Despite its use as a black box for CLIR, an MT system contains much richer representations of the translation hypotheses. It is designed to search this exponential hypothesis space effectively and generate the top-scored translations very efficiently. Therefore, it is natural to exploit the search capabilities of MT systems for CLIR. Based on this motivation, in Chapter 4, we explore the problem of constructing *context-sensitive* and *ambiguity-preserving* translation probabilities from

[2]The language is foreign to the user providing the query.

the rich internal representation of a statistical MT system. We present a novel approach to incorporate various representations of the translation model within a modern statistical MT system, for the purpose of query translation in CLIR. In particular, we present a set of methods to construct a word translation probability distribution from (a) the translation grammar (from either hierarchical or flat MT approaches), and (b) the n best translations output by the MT system. Each of these approaches correspond to a compromise between the two extremes in CLIR literature: (c) context-independent word-to-word translations and (d) using MT as a black box to get one-best translation.

Furthermore, since each of these approaches has complementary strengths, we introduce a combination-of-evidence technique based on a linear interpolation between (a), (b) and (c). We also present a learning scheme to optimize these interpolation weights in a supervised manner. Experimental results on three cross-language retrieval tasks (where each task involves searching a collection in a different language) support our hypothesis that context-sensitive and ambiguity-preserving translation techniques are superior. In addition, the interpolated model yields statistically significant improvements over baselines (c), consistently for all three collections, regardless of the underlying MT model. Our conclusion is even stronger if a hierarchical MT system is used: in this case, the improvement over both (c) and (d) is statistically significant for all three collections.

1.3 Contributions

There are several contributions of this dissertation, which can be grouped by the problems each corresponds to, listed below.

Pairwise Similarity

1. We adapt Locality Sensitive Hashing (LSH) techniques to solve the cross-lingual pairwise similarity problem. We introduce a MapReduce implementation of an existing LSH-based sliding window algorithm and empirically demonstrate its linear scalability characteristics. By using LSH methods, our approach can significantly reduce the amount of work required by a naive brute-force implementation, just by trading off a small amount of effectiveness.

2. We derive an analytical model of the sliding window algorithm, by introducing a deterministic version of the randomized process. This allows us to find a theoretical estimate of effectiveness (i.e., recall) as a function of the collection characteristics. This is very helpful due to two reasons: with LSH-based approaches, a common issue is the existence of many parameters, with little guidance on how to set or tune them for a given task. On the other hand, a challenge with data-intensive approaches is the long running time of each experiment, which makes parameter tuning very time-consuming. With our analytical formulation, the user can estimate effectiveness and efficiency without running any experiments. As a result, by easily determining the effect of

each parameter, it is much easier to find a parameter set tuned for a particular application.

3. We present a detailed empirical exploration of the parameter space when solving pairwise similarity on German-English Wikipedia. We also compare these results to our analytical model, and show that our model reasonably estimates the true recall values for a range of parameter settings. We point out various tradeoff decisions within the parameter space, and provide insight about the cases in which LSH-based methods are beneficial for this problem.

Parallel Text Extraction

1. We introduce a MapReduce algorithm to effectively parallelize computational work when generating candidate sentence pairs from the output of the pairwise similarity algorithm (i.e., pairs of document identifiers that correspond to similar document pairs).

2. We introduce a novel two-step classification approach for efficiently yet effectively deciding which candidate sentence pairs should be added to the final parallel corpus.

3. We demonstrate the potential of our approach by evaluating it both intrinsically (i.e., assessing classification accuracy by comparing against ground truth) and extrinsically (i.e., using the extracted parallel text for training a MT system). Experiments indicate that improvements are possible over strong baselines for a diverse set of language pairs.

4. We explore the effect of extracted parallel text size and translation quality on the evaluation measure (i.e., BLEU score). This analysis demonstrates the trade-off between *less and higher-quality* and *more and lower-quality* data. In order to show the robustness of our approach and strengthen our conclusions, we experiment with a diverse set of six language pairs: German-English, Spanish-English, Chinese-English, Arabic-English, Czech-English, and Turkish-English.

Cross-Language Information Retrieval

1. To the best of our knowledge, our CLIR approach is the first to incorporate the internal representation of modern statistical MT systems into the query translation process. By exploiting different components of an MT system, we show that it is possible to combine the advantages of token-based and 1-best MT-based CLIR approaches.

2. We empirically demonstrate that a combination-of-evidence technique might improve CLIR performance beyond any of the individual approaches.

Searching to Translate vs. Translating to Search

1. This dissertation provides a thorough exploration of one way to connect the fields of IR and MT, by showing that search can be used to improve translation and vice versa. As described above, we present solutions to two problems, corresponding to the two sides of the integration of IR and MT. Additionally, we introduce a more general framework, providing a vision on how a better

integration of IR and MT might be used to improve the state of the art of both fields.

2. Within this general framework, we explore a bootstrapping approach to show that improvements from parallel text extraction can be bootstrapped into the CLIR model, resulting in an improved extraction approach. Based on our evaluation, which is described in Chapter 5, we see additional improvements to the BLEU score after this bootstrapping procedure.

Finally, in support of open science, all of the produced data are released openly. This includes a set of similar Wikipedia article pairs, as well as a parallel corpus for six language pairs: German-English, Spanish-English, Czech-English, Turkish-English, Chinese-English, and Arabic-English. Additionally, all of our code is freely available to the entire research community, as part of the Ivory project at University of Maryland.[3] We believe these novel resources are helpful for research in many cross-language problems, including but not restricted to MT and IR.

[3]ivory.cc

Chapter 2: **Related Work**

This chapter contains an overview of previous work related to various parts of this dissertation. Being at the center of this dissertation's focus, we will first introduce the unfamiliar reader to Machine Translation and Information Retrieval, respectively in sections 2.1 and 2.2. Following these two sections, we will summarize previous attempts to solve the problems we explore in this dissertation, mainly context-sensitive query translation in cross-language information retrieval, cross-lingual pairwise similarity, and parallel text extraction.

2.1 Machine Translation

Machine translation (MT) is the task of translating text written in one natural language (source language) into corresponding text in another language (target language). Until early 1990s, MT research focused on rule-based systems, where linguistic experts would manually create a set of rules that described how text in the source language transforms into text in the target language, by representing both structural and lexical transformations [69]. After the 1990s, with increasing availability of large datasets and computational power, MT research started to move towards statistical methods using machine learning techniques. This led to the rise

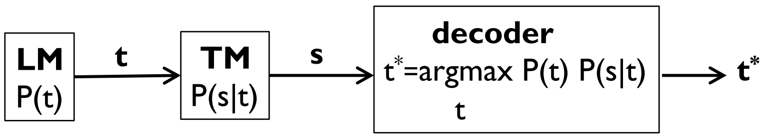


Figure 2.1: Illustration of the noisy channel process, describing how a target-language sentence t is first generated, then gets garbled into the source language. Given the output of the noisy channel process, the MT decoder tries to recover the most probable explanation: the translation that is most likely to have been distorted into the input sentence.

of statistical machine translation (SMT), in which the text to be translated is assumed to have gone through a noisy channel [127]. In this model, we assume some sentence t was generated in the target language, but then got garbled into the source language, so that the task is to recover the most probable sentence t^* that could explain the given source sentence s. The process therefore consists of two probabilistic sub-processes: (i) generation of the target-language sentence (i.e., $P(t)$), and (ii) translation from target to source language (i.e., $P(s|t)$). Figure 2.1 illustrates the noisy channel model, and the following formulates this process:

$$P(t|s) = \frac{P(s|t)P(t)}{P(s)} \tag{2.1}$$

Here, s represents a sentence in the source language, whereas t represents a sentence in the target language. Given s, we want to compare possible translation hypotheses t, thus the denominator $P(s)$ is insignificant. The remaining two processes $P(s|t)$ and $P(t)$ are modeled separately, commonly referred to as the *translation model* and *language model*, respectively. These are combined into a log-linear model to describe the likelihood of a given hypothesis t:

$$\log P(t|s) = \log[P_{\text{TM}}(s|t) \times P_{\text{LM}}(t)] \tag{2.2}$$

$$= \log P_{\text{TM}}(s|t) + \log P_{\text{LM}}(t) \tag{2.3}$$

There are interesting problems on how to train and use language models (see [64] for a review of language modeling techniques used for speech recognition, as well as other related work [14, 30, 152, 68]), but we leave aside the language modeling problem in this dissertation, as we are mainly interested in translation modeling. As a complementary resource, Lopez's survey [86] provides a detailed literature review of statistical MT approaches.

The next two sections (Sections 2.1.1–2.1.2) describe the modeling part of MT. In Section 2.1.3, the decoding problem is described. Finally, in Section 2.1.4, we present some background on MT evaluation.

2.1.1 Word Alignment

Statistical models of translation mostly descended from the IBM Models [16, 17], which encode the translation process at the word level. In these models, we assume there is an alignment that explains how words were transformed from the source language to the target language. Given that $s = s_1...s_m$ and $t = t_1...t_l$, an alignment $a : \{1, ..., m\} \rightarrow \{0, ..., l\}$ is a function determining which target word each source word is aligned to. t_0 is considered as a special NULL value, so that a value of 0 means the source word is not aligned to any target word. Under this model, the probability of s being the translation of t is a sum over all possible ways to align words in s and t:

$$P_{\text{TM}}(s|t) = \sum_{\text{alignment } a} P(s,a|t) \tag{2.4}$$

$$= \sum_{\text{alignment } a} P(a|t)P(s|t,a) \tag{2.5}$$

where $P(a|t)$ is the probability of the alignment, and $P(s,t|a)$ is the probability that s is generated, given the alignment structure. We call the resulting translation model P_{TM} to distinguish it from other distributions. Once Equation 2.5 is computed, the most likely alignment is given by the following equations:

$$P(a|s,t) = \frac{P(s,a|t)}{\sum_{\text{alignment } a'} P(s,a'|t)} \tag{2.6}$$

$$a^* = \arg \max_{\text{alignment } a} P(a|s,t) \tag{2.7}$$

IBM Model 1 describes a simple process for word alignment, by naively assuming that (i) each possible alignment is equally probable for a given target sentence, and (ii) each source word is determined only by the target word it is aligned to. Given a target sentence t, the process of word alignment takes three steps in this basic model:

1. Pick the number of source words to generate, m, with constant probability C.

2. Pick some alignment between $s = s_1...s_m$ and $t = t_1...t_l$ from a uniform distribution. Each alignment has equal probability: $\frac{1}{(l+1)^m}$.

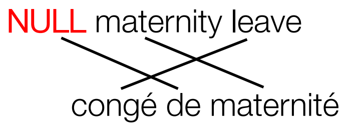


Figure 2.2: Illustration of how words are aligned under IBM Model 1.

3. Generate each source word s_j with probability $P(s_j|t_a(j))$.

As a result, under IBM Model 1, the alignment probability is the following:

$$P_{\text{Model 1}}(a|s,t) = C\frac{1}{(l+1)^m}\prod_{i=1}^{m} P(s_i|t_{a(i)}) \tag{2.8}$$

In order to illustrate the alignment process in IBM Model 1, consider the sentence pair in Figure 2.2. In this example, the variables are instantiated as follows:

$t =$ “maternal leave”

$s =$ “congé de maternité”

$l = 2$

$m = 3$

As stated above, there are $(l+1)^m$ possible alignments, which is 27 for this example, one of which is is illustrated in Figure 2.2. The probability of this alignment can be computed via Equation 2.8:

$$C\frac{1}{27}P(\text{de}|\text{NULL}) \times P(\text{maternité}|\text{maternal}) \times P(\text{congé}|\text{leave}) \tag{2.9}$$

There are other word alignment models with fewer assumptions, resulting in more complex yet accurate models. For instance, IBM Model 2 contains additional parameters for *distortion* (i.e., the position of a target word t depends on the position of the source word s aligned to t), and Model 3 also includes the notion of *fertility* (i.e., the number of source words each target word is aligned to depends on the target word) [17].

The remaining question is: How do we learn the parameters of a given alignment model (e.g., $P(s_i|t_j)$)? Parallel text is available for many language pairs, however alignments at the word level are usually not marked. If words alignments were known, we could estimate parameters of IBM Model 1 as follows:

$$P(s_i|t_j) = \frac{c(s_i, t_j)}{\sum_k c(s_k, t_j)} \tag{2.10}$$

where $c(s_i, t_j)$ is the number of times source word s_i is aligned to target word t_j within the parallel corpus.

Since word alignments are not given, they need to be treated as unobserved (or hidden) variables, requiring an unsupervised process to learn these parameters. The Expectation-Maximization algorithm [34] can be used to learn the set of parameters that maximize likelihood of the training data:

$$\begin{aligned}\theta^{(t+1)} &= \arg\max_{\theta} \log P(D; \theta^{(t)}) \qquad (2.11)\\ &= \arg\max_{\theta} \sum_{(s,t)} \log P_{\mathrm{TM}}(s|t; \theta^{(t)}) \quad \text{(independence assumption)}\\ &= \arg\max_{\theta} \sum_{(s,t)} \log \sum_{\text{alignment } a} P(a|t; \theta^{(t)}) P(s|t, a; \theta^{(t)}) \quad \text{(by Equation 2.5)}\end{aligned}$$

This is an iterative learning procedure that will update parameters at current iteration (i.e., $\theta^{(t+1)}$), using model parameters from the previous iteration (i.e., $\theta^{(t)}$). Since IBM Model 1 describes a convex learning problem, the learning procedure is guaranteed to reach the global maximum (i.e., parameters that maximize data likelihood). More complex models may run into local maxima problems due to a concave learning problem. Therefore, in practice, it is recommended to train IBM Model 1 parameters from a uniform prior distribution, and then use the resulting parameters as initial parameters for more complex models. Additionally, running the word alignment in each direction (switching the source and target languages) and then combining the two mappings has found to increase overall accuracy [106]. After doing this bidirectional alignment, we end up learning a statistical model of how text is translated, in both directions: $P_{\mathrm{TM}}(s|t)$ and $P_{\mathrm{TM}}(t|s)$.

2.1.2 Translation Model

These word alignments describe a statistical model for translation, however using them directly for MT yields poor results. It has been shown that representing translation units at the phrase level provides a much better model for translation, usually referred to as Phrase-Based Machine Translation (PBMT) [71, 107, 92].[1] In phrase-based MT models, a *phrase translation table* can be generated either directly using phrase alignments [92], or more commonly by inducing all bilingual phrase pairs that are consistent with word alignments [71, 107].

Given a sentence pair, $s = s_1...s_m, t = t_1...t_l$, and a corresponding word alignment mapping, a, a bilingual phrase pair consists of a source phrase $p_s = s_i s_{i+1}...$, and a target phrase $p_t = t_j t_{j+1}...$, and is said to be consistent with a if there are no source (target) words aligned to a target (source) word outside of the phrase boundaries (ignoring alignments with the special NULL character). For instance, from the word alignment in Figure 2.2, the following phrase pairs can be generated:

(“congé”	, “leave”)
(“congé de”	, “leave”)
(“maternité”	, “maternal”)
(“de maternité”	, “maternal”)
(“congé de maternité”	, “maternal leave”)

The parameters of this PBMT system are phrase translation probabilities, associated with each generated phrase pair. These parameters can be learned by counting each phrase pair (that is consistent with the word alignment) as an observation, and normalizing as in Equation 2.10.

[1]A phrase is simply a contiguous sequence of words, and does not have any linguistic motivation.

```
R1: [S] || [S,1] || [S,1]
R2: [S] || [X,1] || [X,1]
R3: [X] || [X,1] leave in europe || congé de [X,1] en europe
                                 || 1-0 2-3 3-4 || 1.0
R4: [X] || maternal || maternité || 0-0 || 0.69
```

Figure 2.3: A SCFG with four rules.

The major drawback of this PBMT formalism is that we cannot represent “gaps” between words in a phrase, which limits the generalizability. With this in mind, the phrase-based MT approach was extended to include *hierarchical* representations, using synchronous context-free grammars (SCFGs), instead of flat phrase tables [24, 25]. In this approach, the translation model is a SCFG, which consists of rules of the format `[X] ||` α `||` β `||` $\mathcal{A}$ `||` $P(\alpha \rightarrow \beta)$, indicating that the context free expansion $X \rightarrow \alpha$ in the source language occurs synchronously with $X \rightarrow \beta$ in the target language, with a probability of $P(\alpha \rightarrow \beta)$. Rule probabilities are computed similarly to phrase translation rules in PBMT systems, based on lexical translation probabilities learned from word alignments (i.e., parameters of IBM Model 1). In this case, we call α the Left-Hand Side (LHS) of the rule, and β the Right-Hand Side (RHS) of the rule. We use indexed nonterminals (e.g., `[X,1]`) since in principle more than one nonterminal can appear on the right side. A sequence of word position pairs $\mathcal{A}$ represents the word alignment function (i.e., which word in α is aligned to which target word in β).

Consider the four rules in the SCFG described in Figure 2.3. `S` refers to the sentence; therefore, the first two rules are special rules without any lexical items, describing that there is one sentential form, consisting of a single variable. In the third and fourth rules, we see the structure of the English phrase and how it is

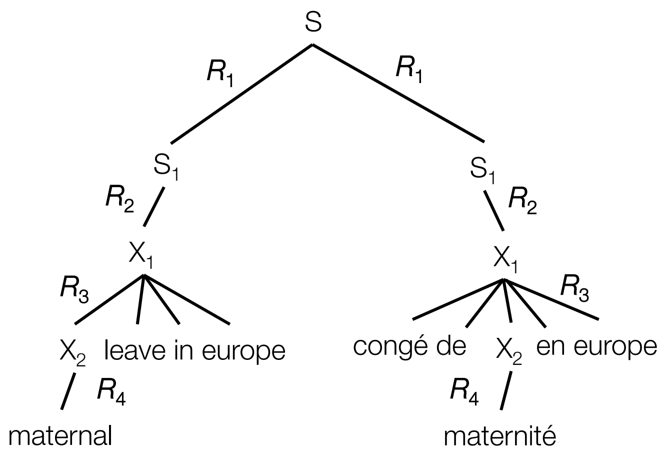


Figure 2.4: Illustration of how *maternal leave in europe* is translated with a synchronous context-free grammar.

translated into French.

Figure 2.4 shows the derivation tree that synchronously parses and translates the source text (tokenized as *maternal leave in europe*) into *congé de maternité en europe* using the above grammar rules. The left and right trees in the figure correspond to the bottom-up parse of the source text and its translation, respectively. Each line between symbols in the figure indicates a rule application from the above grammar, in which case we annotate it with the corresponding rule id.

Notice the representational advantage of having variables in addition to phrase pairs: this approach can model translations with "gaps," allowing a better generalization. As an example, rule R_3 can be used for translating *paternal leave in Europe* as well, assuming we have an additional rule R_5 for the lexical translation:

R_5.`[X] || paternal || paternité || 0-0 || 0.72`

Rule R_4 is not applicable in this case, so R_5 replaces it, producing *congé*

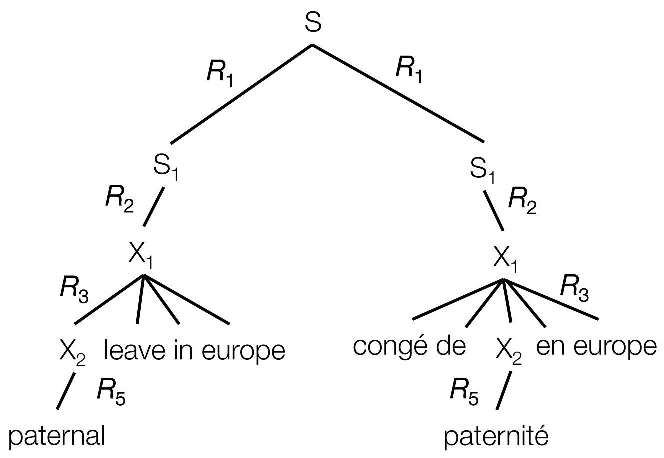


Figure 2.5: Illustration of how *paternal leave in europe* is translated with a synchronous context-free grammar.

de paternité en europe (Eng. *paternal leave in Europe*) as the translation. This derivation is illustrated in Figure 2.5.

In order to make it easier for comparison, we hereafter refer to phrase translation tables and SCFGs as "flat" and "hierarchical" translation grammars, respectively.

2.1.3 Decoding

The above derivation is only one example among many possible ways to translate the source sentence, *maternal leave in Europe.* Since there might be multiple ways to generate the same translation, we need to marginalize over the entire set of derivations to get the true probability of a translation:

$$P_{\mathrm{TM}}(s|t) = \sum_{D \in \mathcal{D}(s,t)} P_{\mathrm{TM}}(s, D|t) \tag{2.12}$$

where s and t are source and target sentences, and $\mathcal{D}(s,t)$ is the set of derivations that synchronously generate s and t.

The size of $\mathcal{D}$ grows exponentially, therefore Equation 2.12 is typically approximated by replacing the sum operator with a maximum operator (called the Viterbi approximation) [25, 71]. With the ability to score a given translation candidate, we can search through all possible translations and find the highest scoring one:

$$t^* = \arg\max_t P(t|s)$$

This process is called *decoding*, and modern MT systems provide efficient algorithms to decode a given sentence efficiently, using heuristics to prune the search space meaningfully (e.g., cdec [36]). The actual decoding speed depends on how much pruning is performed and the CPU specifications, but a handful of sentences can be decoded within a second even on a single desktop computer.

2.1.4 Evaluation

When evaluating MT output, two aspects are most important: fluency and adequacy. The former corresponds to how well-written the output is, in terms of being grammatical and having correct word choices. The latter corresponds to how well the meaning of the source text is preserved in the output. Since it is time-consuming and costly to have humans evaluate MT output, researchers have turned to automatic methods, which compare the MT output to a human reference translation. Currently, the most popular metric is BLEU, which calculates an n-

gram error rate for $n = 1, ..., k$ and produces a weighted average of these.[2] This measure is intolerant to word choice, and will assign lower scores to synonyms of reference translations. In order to alleviate this issue, using multiple references has shown to produce a more appropriate assessment. There are many points of criticism towards BLEU, however this debate is out of the scope of this dissertation. For more information on MT evaluation and alternative metrics, see Chapter 8 of Koehn's SMT book [69]).

2.2 Information Retrieval

Below is a broad definition of the academic field of Information Retrieval, by Manning, Raghavan, and Schütze [91]:

"Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)."

In our work, we assume that the *information need* is provided in the form of a query (ranging from a few words to an entire document), and the *relevant material* is a ranked list of documents written in natural language. Therefore, the task is to compute a relevance score, $Score(D|q)$, for each document D in a large collection of documents, given a query q.

[2] A typical value for k is 4 [69].

2.2.1 Vector-Space Model

A typical approach in IR is to represent each document as a vector of weighted *terms* (called *document vector*), where a term is a class representing some unit of text in the document (usually words or stems). A pre-determined list of stop words (e.g., "the," "an," "my") may be removed from the set of terms, since they have been found to create noise in the search process. In this so-called *vector space model*, the weight of each term represents its importance in the document or query.[3] The relevance of document D, relative to query q is usually computed by treating each query term independently, and aggregating the term-document weights:

$$Score(D|q) = \sum_{\text{term } q_j \text{ in } q} \omega(q_j, D) \tag{2.13}$$

The ω function is an IR weighting scheme, and can be implemented in many different ways, such as BM25 [118]. The most important two components of an IR weighting scheme are *term frequency* (tf) and *document frequency* (df). The former is the ratio of frequency of the query term in the document, and the latter is the ratio of documents in which the query term appears, within a large collection:

$$\text{tf}(q_j, D) = \frac{\text{count of } q_j \text{ in } D}{\text{number of tokens in } D} \tag{2.14}$$

$$\text{df}(q_j) = \frac{\text{number of documents } q_j \text{ appears in}}{\text{total number of documents in collection}} \tag{2.15}$$

Term frequency corresponds to relevance or "aboutness" of the term within

[3]The first known use of this model is in the SMART retrieval system [91, 119].

the document context, whereas document frequency refers to the specificity or "rareness" of the term [118]. Models with a weighting scheme based on the term and document frequency values are referred to as tf-idf models.

Additionally, it is useful to perform document length normalization in a weighting scheme. This might involve normalizing all document vectors to unit length, or learning a pivot document length and normalizing all document lengths with respect to that [128].

2.2.2 Other Models

In addition to the vector space model in Equation 2.13, alternative approaches have been successful at modeling relevance in IR. A popular approach is probabilistic IR, in which the probability of query-document relevance is modeled directly, by computing the *odds* of a given query-document pair being relevant:

$$\begin{aligned} odds(D,q) &= \log \frac{P(\text{relevant}|D,q)}{P(\text{non-relevant}|D,q)} && (2.16)\\ &= P(R=1,D,q)/P(R=0,D,q) && (2.17)\\ &= P(D|R=1,q)P(R=1,q)/P(D|R=0,q)P(R=0,q) && (2.18)\\ &= P(D|R=1,q)/P(D|R=0,q) && (2.19)\\ &\ (P(R,q) \text{ is constant for given query}) \end{aligned}$$

Assuming that words are independent and that words not in the query are equally likely to appear in relevant and non-relevant documents, we end up with the following formulation:

$$odds(D,q) = \prod_{\{\text{term } j:\ q_j \in D\}} p_j(1-u_j)/u_j(1-p_j) \tag{2.20}$$

$$p_j = P(q_j \in D | D \text{ is relevant}, q) \tag{2.21}$$

$$u_j = P(q_j \in D | D \text{ is non-relevant}, q) \tag{2.22}$$

For easier computation and interpretation, we can use the logarithm of the *odds* function, resulting in a value known as the *Retrieval Status Value* (RSV) [44]. There are various alternatives for estimating the parameters u_j and p_j. The former (probability of query term appearing in non-relevant document) corresponds to a df-like "rareness" feature, since almost all of the documents in a collection are non-relevant. The latter (i.e., probability of query term appearing in relevant document) is commonly assumed 0.5 [91], but can be updated after an initial retrieval run, by *pseudo-relevance feedback*: assuming the retrieved documents are relevant, set the estimate based on maximum likelihood.

Another successful retrieval model is based on language modeling, introduced by Ponte and Croft [113]. In this formalism, each document D is represented by a language model, which is a probability distribution of terms (i.e., sum of probabilities for all terms in D is 1). A *term* might simply refer to word or stem classes, but it is also possible to use more complex linguistic structures as terms, such as multi-word expressions, or grammar-based syntactic forms.

Given a query q, the relevance of each document is defined probabilistically (i.e., probability of document, conditioned on query), which is augmented as follows:

$$P(D \text{ is relevant}|q) = \frac{P(q|D \text{ is relevant})P(q)}{P(D \text{ is relevant})} \qquad \text{(by Bayes' Rule)} \tag{2.23}$$

$$P(D \text{ is relevant}|q) \sim \frac{P(q|D \text{ is relevant})}{P(D \text{ is relevant})} \qquad (P(q) \text{ is constant for given query}) \tag{2.24}$$

$$\sim P(q|D \text{ is relevant}) \qquad (P(D) \text{ is assumed to be uniform}) \tag{2.25}$$

$$\sim \prod_{\text{term } j} P(q_j|D \text{ is relevant}) \qquad \text{(word independence assumption)} \tag{2.26}$$

Language model parameters can be estimated by the term frequency values (Equation 2.14), but they will suffer from data sparsity: a document has a limited amount of text to appropriately estimate the underlying model of language use. This issue is addressed by using smoothing techniques, such as the Jelinek-Mercer method, which interpolates between estimates from the document and a more general collection, C [151]:

$$P(q_j|D) = \alpha \, \frac{\mathrm{tf}(q_j, D)}{\sum_w tf(w, D)} + (1 - \alpha) \; P(q_j|C) \tag{2.27}$$

Another discounting approach is Bayesian smoothing with Dirichlet priors:

$$P(q_j|D) = \frac{\mathrm{tf}(q_j, D) + \alpha\ P(q_j|C)}{\sum_w tf(w, D) + \alpha} \tag{2.28}$$

In IR approaches, it is quite common to make the independence assumption between query words, although it is obviously a very strong assumption that almost never holds. On the other hand, there have been previous approaches that model the local context into the retrieval model. Explicitly representing some of the term dependencies in a structured query form has yielded success empirically [49, 97], and there are many approaches that consider multi-word expressions/phrases [5, 155] in a rather ad-hoc way. Despite various attempts to overcome the independence assumption, the word-based vector space model described in Equation 2.13 has remained as a standard baseline approach in IR, mainly due to its simplicity, efficiency, and flexibility.

2.2.3 Retrieval

In IR, we are generally interested in the top k scoring documents of a collection of size N, given a query q. Therefore, a naive implementation of the retrieval process may use a heap data structure of size k, using the query-document score (i.e., $Score(D|q)$) as key and the document id as value. Once all N documents are processed, we are guaranteed to have the top k documents in the heap. This procedure requires $O(N\log k)$ time complexity and $O(k)$ space complexity.

However, notice that we do not need to process all documents in order to retrieve the ranked list for q. One way to optimize the process is to ignore documents that do not contain any of the query terms. In order to achieve this, an *inverted index* of the collection is built, which is a mapping from each term to its posting list: a list of pointers to the documents the term appears in. Each document in the collection is assigned a *document id*, which is a unique integer identifier of the document. The posting list of each term, say t, uses the document id as a pointer to the actual document D, along with other information, such as the term frequency (i.e., $\text{tf}(t, D)$), and sometimes the position of t within D. There are many approaches to build and maintain the inverted index, which is out of the scope of this dissertation, but a few points are worth mentioning.

For retrieval, there are two main approaches to computing the query-document scores (i.e., $Score(D, q)$): term-at-a-time and document-at-a-time [140]. The latter technique processes documents one by one, computing the query-document score before moving to the next document. By having access to an inverted index, we only need to process documents that appear in the union of the query terms' postings lists.[4] This reduces the amount of processing substantially, when compared to the $O(N\log k)$ complexity shown above.

For even better optimization, we can take advantage of the term independence assumption, and compute partial query-document scores, for each query term. For example, for the query "maternal leave," the partial score of term *maternal* is computed for each document in the index: $\omega(\text{“maternal”}, D_0)$, $\omega(\text{“maternal”}, D_1)$, ...,

[4]If we have a boolean query, it is sufficient to consider documents at the intersection for `AND`, and union for `OR`.

and so on. Each document has a score accumulator, into which these partial scores are accumulated. After all query terms are processed, each accumulator contains the corresponding query-document score. Due to the extra storage requirements of this approach, many researchers have studied methods to limit the number of accumulators based on a variety of optimization strategies [78, 111, 134].

The major advantage of term-at-a-time scoring is the ability to apply certain heuristics. Documents in a posting list are usually sorted either by their document id, or by the term frequency. The latter order allows early termination heuristics with term-at-a-time scoring: after the tf falls below a certain threshold, we can terminate the processing of the postings list. Another useful heuristic is to process terms in order of their df value, assuming that terms with lower *df* should contribute higher partial scores, and it might be possible to terminate before processing some of the high-*df* terms.

2.2.4 Evaluation

There have been many evaluation measures proposed for the retrieval task [91]. Precision is the ratio of correctness among identified relevant documents, whereas recall is the ratio of truly relevant document that were identified. Therefore, the general focus of evaluation strategies is to find a metric balancing *precision* and *recall*, where the perfect balance depends on the application: web search requires high precision, whereas paralegals expect very high recall. F-score (or F_1 score) is a popular way to combine precision and recall without any prior preference. It is the harmonic mean of these two values:

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.29}$$

Another way to think of combining precision and recall is through the so-called precision-recall curve, which plots the precision of the system as a function of recall. The retrieval system is then measured by the area under this curve. This requires an integral of the function over recall values in range [0,1], which is not feasible to compute exactly. As a solution, 11-point *interpolated average precision* approximates this value by computing precision at 11 recall points within [0,1], at equal intervals of 0.1. An alternative is to (starting from the highest ranked document) compute precision after every relevant document in the ranked list, until the list is exhausted, and divide the sum of these precision by the total number of retrieved documents. This is called the average precision (AP), and the mean of AP values across a set of queries corresponds to MAP. MAP has shown to be stable and has been widely adapted in the IR community.

Other measures may incorporate graded relevance scores, since binary values (i.e., relevant vs. non-relevant) are often incapable of capturing accurate assessments. *Discounted cumulative gain* (DCG) computes the total gain of the retrieved documents, in which the document ranked i^{th} in the ranked list (i.e., d_i) contributes a gain of $g_i = \text{relevance}(d_i)/\max\{\log(i), 1\}$. In order to allow a perfect 1.0 value, the sum of gains is normalized by dividing to the upper bound, yielding the normalized discounted cumulative gain, or $nDCG$:

$$DCG_m = \sum_i^m g_i \tag{2.30}$$

$$nDCG_m = \frac{DCG_m}{(\sum_i^m 1.0/\max\{\log(i), 1\})} \tag{2.31}$$

2.2.5 IR Across Languages

Cross-language information retrieval (CLIR) is a special case of IR, in which the query is written in a language different than the documents. This has become a popular problem due the increase of multi-lingual collections.

The mismatch between query and document languages needs to be addressed. Although there are approaches that try to find a language-independent "concept space" for CLIR, the commonly adapted, more efficient solution is to perform "translation". This is different than Machine Translation since producing translated text is not the *goal* of CLIR; it is merely used as a *tool* to match the vocabularies of queries and documents.

In the CLIR literature, there are two main approaches to address the vocabulary mismatch problem: translating the query into document language, or translating the documents into the query language. One advantage of query translation is the opportunity for interacting with the user to enhance the query translation, as well as easier integration with expansion techniques. Another benefit is the lower computational work associated with translating queries, when compared to all documents in the collection. On the other hand, from the user's point-of-view, retrieved documents need to be translated anyway, so we might as well do that in advance. A major drawback of query translation is relatively low translation quality, due to limited context in short queries (typically a few words), as opposed to well-formed long sentences that commonly comprise document text. In this dissertation, we focus on query translation due to its popularity in the CLIR literature, making it easier to compare to related work. However, all proposed methods can be adapted for document translation just as well.

Regardless of whether queries or document are being processed, there are two main ways to achieve translation in CLIR: directly applying an MT system to translate the text, or by projecting the vector representation through a term-to-term bilingual mapping. Let us illustrate the difference between the two translation approaches by an example query, $q =$"maternal leave". The MT-based CLIR approach translates the query into the target language (e.g., French), which turns out as $q' =$"congé de maternité". The resulting translated query, q', can be directly used for retrieval, treating it as the original query:

$$Score_{\text{CLIR-MT}}(D|q) \tag{2.32}$$

$$= Score(D|q') \tag{2.33}$$

$$= \omega(\text{“congé”}, D) + \omega(\text{“de”}, D) + \omega(\text{“maternité”}, D) \tag{2.34}$$

The computation of this sum follows naturally, since all documents are already in the target language (i.e., French), and therefore it is straightforward to compute the tf and df of the translated query's terms.

As an alternative approach, computing $Score(D|q)$ directly would require defining the weights of the non-translated query terms. Assuming a standard weighting scheme with tf and df components, the query-document score is computed as follows:

$$Score_{\text{CLIR-Token}}(D|q) = \sum_{q_i \in q} \omega(\text{tf}(q_i, D), \text{df}(q_i)) \tag{2.35}$$

which corresponds to the following equation for our running example:

$$\begin{aligned} &Score_{\text{CLIR-Token}}(D|\text{“maternal leave”}) = \\ &\qquad \omega(\text{tf}(\text{“maternal”}, D), \text{df}(\text{“maternal”})) + \omega(\text{tf}(\text{“leave”}, D), \text{df}(\text{“leave”})) \end{aligned} \tag{2.36}$$

In order to complete the missing parts in this equation, Darwish and Oard proposed a translation approach based on a token-to-token *bilingual mapping* [31], which we call “token-based” CLIR. This mapping consists of probability distributions for each word in the query-language vocabulary, where each distribution describes the possible translations in the document language, with associated probabilities. For example, the following is the distribution for the translation of “maternal”:

$$P_{\text{maternal}}(\text{“maternel”}|\text{“maternal”}) = 0.8 \tag{2.37}$$

$$P_{\text{maternal}}(\text{“maternelle”}|\text{“maternal”}) = 0.15 \tag{2.38}$$

$$P_{\text{maternal}}(\text{“maternité”}|\text{“maternal”}) = 0.05 \tag{2.39}$$

These values can be learned from the word alignments in a parallel corpus, as shown in Section 2.1.1, in Equation 2.10. Basically, after running the iterative learning procedure, the parameters of IBM Model 1 define the translation probabilities required for Equation 2.37.

Once the translation probabilities are known, the tf and df values can be translated as follows:

$$\mathrm{tf}(q_j, D) = \sum_{t \in D} \mathrm{tf}(t, D) P_{q_j}(t|q_j) \tag{2.40}$$

$$\mathrm{df}(q_j) = \sum_{t \in D} \mathrm{df}(t) P_{q_j}(t|q_j) \tag{2.41}$$

Following our previous example, the term frequency becomes:

$$\begin{aligned} \mathrm{tf}(\text{“maternal”}, D) = {} & \mathrm{tf}(\text{“maternel”}, D) \times 0.80 \\ & + \mathrm{tf}(\text{“maternelle”}, D) \times 0.15 \\ & + \mathrm{tf}(\text{“maternité”}, D) \times 0.05 \end{aligned} \tag{2.42}$$

and similarly for the document frequency computation.

After the tf and df of the query terms are computed, we can plug in the values into Equation 2.35 to score documents in a cross-language setting.

The token-based CLIR approach described above achieves strong empirical results [31], mostly because it *preserves the ambiguity* of language. The underlying bilingual mapping is learned from a large number of sentence pairs, capturing the diverse language use existent in the training examples. However, this also means that success is highly dependent on how similar the training data is to the given query. Furthermore, highly ambiguous words have very noisy translation choices. For example, there are many different senses of the word "leave," such as *take time off*, *forget*, and *exit*, and each of these senses corresponds to a different translation: in French, *congé*, *laisser*, and *quitter* refer to these three senses. The most appropriate translation choice depends on the sense of the word, therefore it is essential to consider which sense of the word is used in the query. For instance, knowing that the query is "maternal leave," the probability that "leave" is used in the *take time off* sense increases substantially.

By incorporating the query context into the translation process, we potentially find more appropriate translation choices. This is exactly why the MT-based approach (Equation 2.32) sometimes outperforms token-based CLIR (Equation 2.35). The former method takes context into account, implicitly through the use of MT techniques, which model the query translation without making any word independence assumptions.

Due to its critical role in the success of CLIR, this polysemy problem has received much interest from researchers: given all possible ways to translate each query term, the task is to select, rank, or assign probabilities to translation choices, conditioning on the query context (i.e., computing updated probabilities $P_{q_j}(t|q_j, q)$) for each query term q_j. We refer to this problem as *context-sensitive CLIR* and list related work below.

2.3 Context-Sensitive CLIR

In this dissertation, we introduce novel solutions for the CLIR problem, focusing on the notions of *context-sensitivity* and *preserving linguistic ambiguity.* As mentioned above, there are two main translation approaches in CLIR, which we refer to as MT-based (Equation 2.32) and token-based (Equation 2.35). These approaches have complementary strengths: MT makes good use of context, but at the cost of producing a single translation, thus discarding other possibilities that might be useful. Therefore, these approaches are context-sensitive, but not ambiguity-preserving. On the other hand, token-based approaches preserve all of the ambiguity that exists in the training examples, from which the translation probabilities are trained, yet do not model the local query context. As a result, these approaches produce ambiguity-preserving, yet context-independent translation probabilities.

While both of these approaches have been successful in cross-language retrieval tasks, there is room for improvement. As reviewed in Section 2.1, statistical MT systems consist of *context-sensitive* and *ambiguity-preserving* models of translation, as well as approximate-search algorithms to efficiently search and rank the most appropriate translations. In spite of such appealing properties, CLIR research has borrowed very little from the field of MT. Related work is listed below, followed by a discussion of what is missing, and the novelty of our work.

One of the first attempts to incorporate MT ideas into IR research dates back to Ponte and Croft's use of a language model to score query-document pairs in monolingual IR [113]. Details of this approach were presented in Section 2.2.2.

Drawing further inspiration from MT, Berger and Lafferty extended the idea, by modeling retrieval in the same generative process as the noisy channel model in MT. It is assumed that this noisy channel corrupted some document(s) into the query, and the task is to recover these document(s) [10]. Their probabilistic model contains a *translation component* that represents synonymy or other word relationships.

Applying these ideas to the cross-language case was a natural extension, by combining a language model and a translation model to perform CLIR, similarly to MT. Given a query q, and document D, Kraaij et al. [73] showed two ways to incorporate the translation model:

1. Modeling translation on the query side:

$$\forall t_i \in T: \; P(t_i|q) = \sum_{\text{query term } s_j \in S} P(t_i|s_j)P(s_j|q)$$

$$Score(D|q) = \sum_{\text{document term } t_i \in T} P(t_i|q) \log P(t_i|D)$$

2. Modeling translation on the document side:

$$\forall s_j \in S: \; P(s_j|D) = \sum_{\text{document term } t_i \in T} P(s_j|t_i)P(t_i|D)$$

$$Score(D|q) = \sum_{\text{query term } s_j \in S} P(s_j|q) \log P(s_j|D)$$

Both of these approaches exhibit substantial improvements over earlier dictionary-based baselines, reporting Mean Average Precision (MAP) scores in the range of 90% of monolingual comparison conditions [73]. Others have presented similar approaches with slightly different smoothing techniques, yielding similarly strong empirical results [148, 147].

Federico and Bertoldi presented a similar approach, in which they use Hidden Markov Models (HMMs) to represent the translation as a sequential process [41], using a bigram language model (with discounting) for transition probabilities, $P(t_{i+1}|t_i)$, in addition to the standard bilingual translation probabilities. The inclusion of transition probabilities allows more appropriate translation choices. For example, in addition to considering possible translations of *maternal* and *leave* separately, say *maternel* and *congé*, the HMM considers the probability of *congé* following *maternel* as well (i.e., $P(\textit{maternel}\ |\textit{congé}\ ))$, which is not part of the other approaches.

In addition to the CLIR models explicitly modeling the translation process, Lavrenko et al. present a relevance model, which they apply to both monolingual [77] and cross-lingual cases [76]. In the latter paper, they describe how to model the joint probability $P(t_i, q_1...q_n)$ for a given document term t_i and query $q = q_1...q_n$, and then use that to approximate relevance $P(D|q)$. Source and target language models are built from each sentence pair of a given parallel corpus, so that the joint probability can be estimated as follows:

$$P(t_i, q_1, ..., q_n) = \sum_{s',t'} P(t_i|\text{LM}(t')) \prod_{i=1}^{n} P(q_i|\text{LM}(s')) \tag{2.43}$$

The translation models used in these approaches have also been successfully used for query expansion [51]. The authors model relevance based a query expansion process, in which each query word is assumed to expand into many document word based on the query context. They provide empirical evidence that the probabilities of this approach can be learned from click logs of web search engines.

As an alternative to approaches that try to model relevance as some generative process, many researchers have instead focused on post-processing the translation space to reduce ambiguity, based on query context. For a given query $s_1 ... s_n$, we can generate many ways to translate each query term, using techniques described in Section 2.2.5. This creates a translation space (assuming s_i has k_i different translations) of size $K = \prod_{i=1}^{n} k_i$.[5] Many of these K choices for the query translation are plausible, but some are more appropriate, given the query context.

Consider our running example, *maternity leave in europe*, and assume that the terms have 3, 4, 3, and 1 possible translations, respectively. Figure 2.6 shows a graph representing the translation space, where each path corresponds to a valid translation (there are $3 \times 4 \times 3 \times 1 = 36$ for this example). Knowing the original query, a French speaker would notice that the most appropriate choice is (maternité → congé → de → europe). Furthermore, even without knowing the original query, just looking at pairwise choices is helpful. For instance, *maternel* (Eng. *maternal*) is much more similar in context to *congé* (Eng. *take time off*) than *laisser* (Eng. *forget*). Similarly, *en* and *Europe* appear much more frequently together than *dans* and *Europe*.

Based on this motivation, many researchers have tackled the problem of scoring

[5]The space becomes much larger if word order is considered.

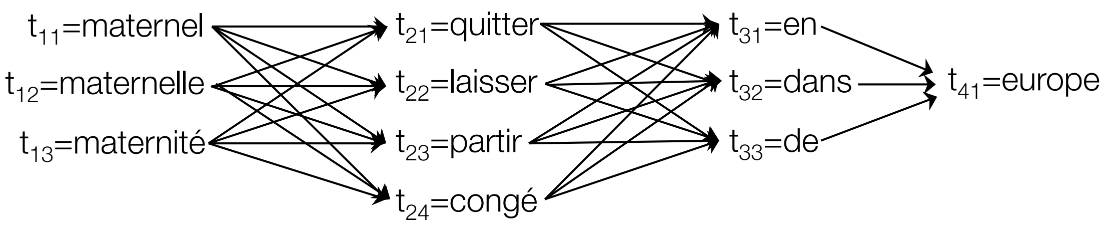


Figure 2.6: A graph representation of the translation space for query *maternal leave in europe.*

pairs of translation terms, in order to select the translation set with highest *cohesion*:

$$Cohesion(t_{1k_1} \rightarrow t_{2k_2} \rightarrow t_{3k_3} \rightarrow t_{4k_4}) = \prod_{i=1}^{4} \prod_{j=1 \wedge j \neq i}^{4} sim(t_{ik_i}, t_{jk_j}) \tag{2.44}$$

where $P(t, t')$ is the probability of t and t' appearing in the same context, which is typically estimated from corpus statistics. Notice that Equation 2.44 includes this measure between all pairs of translation terms, not just consecutive ones, as exemplified above.There is a rich literature on implementing this idea, with many proposed methods to compute the pairwise similarity: Gao et al. [50] use point-wise mutual information, whereas others propose using Dice similarity [2], and mutual information [84]. These approaches select terms greedily (i.e., pick translation of first word that maximizes cohesion, then move to second word, etc.), whereas Seo et al. [125] show further improvements when all possibilities are considered.

Expressing term dependency relations explicitly has been shown to produce good results in monolingual retrieval [49, 97], but extending their ideas to CLIR has proven not to be as straightforward as one might expect. Cao et al. [20] attempt this by modeling the translation space explicitly as a graph structure, and show that a random walk algorithm can be applied to compute probabilities of different paths. Their approach considers synonyms for query expansion as well as translations in their graph-based approach. Another recent paper by Wu and He [146] also integrates expansion and translation, by utilizing an MT-based translation model.

Another way to incorporate dependencies between query terms is to consider multi-word expressions (so-called "phrase translation," although the "phrases" are often statistical rather than linguistic phenomena) in order to limit polysemy effects [2, 5, 7, 23, 95, 155]. Despite the popularity and effectiveness of including phrase translations, they are typically based on existing phrase translation tables, which are limited to certain domains and languages. Also, due to computational difficulties, these approaches incorporate only a small subset of the possible phrase translations.

In contrast with approaches that try to translate the query into document language, or vice versa, there have also been approaches to map both components into a language-independent semantic space, which is modeled as a latent variable [83]. The major drawback of these approaches is the computationally intensive nature of the underlying techniques, such as latent semantic indexing (LSI) [46]. It is also more difficult to interpret the transformation, compared to translation-based CLIR approaches.

Related to LSI-based CLIR is the notion of bidirectional translation introduced by Wang and Oard [145]. In this approach, the goal is to match the meaning of the query and document by using translation probabilities in both directions. This is a direct extension of the token-based CLIR approach, with the modification that translation direction is not explicitly included in the model. The probability of source and target terms s_j and t_i having a matched meaning is defined by the following:

$$P(s_j \leftrightarrow t_i) = \sum_x p(x|t_i) \times p(x|s_j) \tag{2.45}$$

where x represents a meaning.

In their paper, Wang and Oard show that *synsets* (i.e., sets of synonymous terms) in WordNet can be used as a representation of meaning, and present several implementations of this model based on this idea. They report significant improvements over the strong unidirectional CLIR baseline, with MAP scores comparable to a monolingual baseline [145].

As a great summary of research trends in cross-language IR, we refer interested readers to Jian-Yun Nie's recent book [104]. In his book, Nie mentions the need for better integration of MT and IR, and specifically describes how this can be achieved. Below, we state the few previous attempts to achieve this integration, and relate them to the contributions of this dissertation.

One early CLIR system did try augmenting MT output using a bilingual dictionary [75] in order to include alternative translations. With the developments of statistical translation models, modern MT systems contain rich representations of alternative translations of the source text (which is a query in our case). Despite this rich representation, MT-based CLIR approaches typically use one-best results, due to the convenience of treating MT systems as black boxes. Magdy et al. [89]

recently showed that it is beneficial to "look inside" the MT black box, and treat the process specially for query translation. The authors reported improvements in retrieval effectiveness just by modifying the MT preprocessing steps, in accordance with the IR pipeline. More recently, Nikoulina et al. [105] built an MT model tailored to query translation by (a) tuning the MT model weights on a set of queries and reference translations, and (b) reranking the top n translations of the MT system to maximize performance on a held-out query set. While improvements were more substantial using the latter method, another interesting finding was the low correlation between translation and retrieval quality. This indicates that doing better translation as measured by MT standards may not help the retrieval process. Combining the n-best derivations is also routinely used in speech retrieval [109].

Our approach focuses on combining different sources of evidence within an MT system, by building translation models from three different intermediate representations: word alignments, translation grammar (hierarchical or flat), and n-best translation list. Each of these translation models have different characteristics in terms of *context sensitivity* and *preserving ambiguity*, therefore they have complementary strengths. We present an interpolation method to combine the advantages of all three approaches. Hence, our contributions can be considered complementary to the few attempts to integrate IR and MT recently.

2.4 Pairwise Similarity

The pairwise similarity problem involves finding all similar pairs of objects (typically, above some similarity threshold) in a large collection, which has a quadratic time complexity as a function of the collection size. There are two challenging aspects of this task: (1) measuring similarity between two objects efficiently and accurately, and (2) finding efficient and scalable approaches to search the quadratic problem space.

In general, two approaches to the problem exist: *index-based* approaches focus on building an inverted index and employing pruning methods it to achieve efficient similarity search [9, 57, 135, 143, 22], whereas *signature-based* approaches transform the data into a more compact representation for performing an approximate similarity comparison with reduced time and space complexity [40, 15, 21, 4, 26, 53, 4, 122].

Index-based solutions are useful for *exactly* finding which pairs of objects (e.g., documents) share features (e.g., terms). These approaches suffer from the fact that each comparison is computed twice (i.e., it is not straightforward to avoid considering both (x, y) and (y, x)). Another drawback is that that index creation is an extra step before the pairwise similarity computations, and most approaches assume that the index can be loaded onto memory.

Dynamic index creation partially addresses this issue [121], in which case postings are added to the inverted index as objects are processed. Bayardo et al. [9] present a series of optimizations on top of the standard index-based pairwise similarity approach, and report that their final algorithm can achieve state-of-the-art efficiency without making any approximations (as in signature-based methods).

A *signature* is a hash, or footprint, of the vector representation of an object (e.g., document). Therefore, signature-based solutions transform the problem into grouping objects based on their signatures. This allows a much faster search and lower space requirements. Even though there are theoretical guarantees on the upper bound of error due to the approximations in signature-based approaches, there is still a non-trivial amount of error in practice.

Rabin was the first to suggest using hash functions to find similar vectors, where the challenge is to limit the number of hash collisions [115]. One way to achieve this is the highly successful locality-sensitive hashing (LSH) approach, introduced by Indyk and Motwani [63]. In their paper, they show how LSH can be used to solve the nearest neighbor search problem in various domains. Broder introduced the notion of min-wise independent functions to generate signatures (called shingles in the paper), which has shown to satisfy the LSH requirements for Jaccard similarity [15]. Charikar later introduced a method based on random projections to compute signatures, and proved theoretical guarantees for the cosine similarity measure [21].

LSH was first presented as a solution to the nearest neighbor search (NNS) problem: Given a collection of objects and a query, find the object p closest to the query q (i.e., find p^* s.t. $p = \arg\min_{p'} \delta(p', q)$). The approximate version, called ϵ-NNS, finds some object that is at most ϵ farther from the query than its true nearest neighbor (i.e., find p s.t. $\delta(p, q) < (1 + \epsilon)\delta(p^*, q)$). This problem can be reduced to a more general version, called Point Location in Equal Balls (PLEB).

In PLEB, we are given a radius r and a notion of an open ball around some object p: $B(p,r) = \{p' : \delta(p,p') \leq r\}$. For some query q, the task is to decide if there is any object p such that q is within r distance of it (i.e., find p s.t. $q \in B(p,r)$). The approximate version, called ϵ-PLEB, requires the following:

- if $\exists p$ s.t. $q \in B(p,r),$ return `YES` and some object p' s.t. $q \in B(p, r+\epsilon)$
- if $\forall p : q \notin B(p, r+\epsilon),$ return `NO`
- if closest point to q is p and $r \leq \delta(p,q) \leq r+\epsilon,$ return `YES` or `NO`

Indyk and Motwani prove that LSH can be used to solve the ϵ-PLEB problem [63]. Given a collection of vectors in domain S, let $H : S \to U$ be a hash function family, where U is the target domain. H is said to be a (r, ϵ, p_1, p_2)-LSH family if the following holds: $\forall$ objects $u, v \in S,$ and any hash function $h \in H$:

- if $\delta(u,v) \leq r, P(h(u) = h(v)) \geq p_1$
- if $\delta(u,v) > r+\epsilon, P(h(u) = h(v)) \leq p_2$

where δ is a distance metric satisfying the two constraints for H. We use distance instead of similarity to stay consistent with earlier work, but notice that obtaining a corresponding definition for a similarity measure is straightforward.

In order to solve ϵ-PLEB, we can define a LSH signature as a concatenation of k hash values. For the i^{th} signature, we sample k hash functions from H, and define the signature function $g_i : S \to U^k$ s.t. $\forall p \in S, g_i(p) = (h_1(p), ..., h_k(p))$. This procedure is repeated to get l signature functions, based on which the collection is bucketed. This can be visualized as having l hash tables, where keys are computed using the corresponding signature function. Once the collection is split into these l buckets, we can devise an algorithm to answer ϵ-PLEB for a given query q, by searching all buckets corresponding to the query's l signatures, $g_1(q), ..., g_l(q)$. The algorithm stops after finding $2l$ objects within these buckets. Then, we compute distance between the query and each of the $2l$ candidate objects, and return `YES` if we find one closer than $r + \epsilon$.

Correctness of this algorithm is guaranteed if the number of collisions is less than $2l$ (i.e., $|\{p : \exists j = 1...l \text{ s.t. } (\delta(q,p) > r + \epsilon) \wedge (g_j(p) = g_j(q))\}| < 2l$), and similar objects share at least one signature (i.e., $\delta(p,q) < r \Rightarrow g_j(p) = g_j(q)$ for some $j = 1...l$). It has also been proven that these two conditions hold if k is chosen to be $log_{1/p_2} n$, and $l = n^{\frac{\log 1/p_1}{\log 1/p_2}}$, so the time complexity is $O(n^{\frac{\log 1/p_1}{\log 1/p_2}})$ [63, 21].

The drawback of this approach is that it solves ϵ-PLEB; we need to run it for a varying set of r values in order to solve the ϵ-NNS problem. Charikar introduced a different, sort-based approximate search algorithm that directly solves ϵ-NNS [21]. In this algorithm, we generate l random permutation functions, and each signature is permuted with respect to each, resulting in l lists of signatures. Each of these lists is sorted by lexicographic order of the bits, so for a given query q, we perform a binary search on each list to identify the objects with longest prefix. Once we perform $2l$ comparisons, it is guaranteed that we have hit an approximate nearest neighbor. The proof of the correctness and time complexity is very similar to above.

Ravichandran et al. [116] applied Charikar's sort-based algorithm to noun clustering, after modifying the approach to find the nearest neighbors of all objects, instead of a single query. In their approach, instead of a binary search in each sorted list, a fixed-size window slides from top to bottom, and any pair that appears in the same window is compared for similarity. The running time to find all similar pairs is $O(nBQ)$, where B is the window size and Q is the number of permutations.

Following the introduction of LSH techniques through these few papers, many researchers have applied the methods to various problems, ranging from entity resolution [61], to audio identification [55, 98], and image search [74, 65]. A variant of the pairwise similarity problem is *near-duplicate detection*, which aims to detect highly-similar pairs, sometimes without a predefined similarity threshold or even a similarity measure. Both index-based [135, 149] and signature-based [27, 90, 72, 59, 62] approaches have been proposed to address this problem, including Broder's approach that pre-dates LSH [15].

Pairwise similarity has also been extensively studied by the text processing and data mining research community, where it is commonly referred to as the *set similarity join* problem [57, 143]. A more general variant that has received much attention is computing pairwise *functional* computations between elements in a large dataset [99, 66].

There is a line of research using specialized data structures to partition the metric space, such as K-D trees and R-trees, surveyed by Gaede and Günther [47]. The major drawback of these approaches is the increased computational costs in high-dimensional spaces. More recently, Low and Zheng presented an approach based on matrix compression, which finds exact results and is scalable to high-dimensional spaces [87].

2.5 Cross-Lingual Pairwise Similarity

All of the above focus on mono-lingual or homogeneous similarity, either similarity within one language or similarity within a homogeneous collection of objects. In the cross-lingual variant of the problem, the main challenge is to handle the extra noise caused by the translation process. Methods to translate documents in CLIR are directly applicable to objects in any feature space (see Section 2.2.5 for translation approaches in CLIR).

Due to translation ambiguities, the similarity values are lower than the mono-lingual case, requiring lower thresholds to detect pairs. Our preliminary experiments suggest about 0.3 as a threshold in cross-lingual pairwise similarity, as opposed to about 0.6 for mono-lingual, and 0.9 for near-duplicate detection. As a result, it becomes much more difficult to distinguish between similar and dissimilar pairs. In order to alleviate this issue, it is essential to devise approaches where the error introduced by LSH techniques is minimal. We provide more details on this topic in Chapter 3.

Although they are not interested in the cross-lingual case, Zhai et al. [153] tackle the problem of similarity search with the same focus: finding similar pairs when the similarity threshold is low. Their approach is a combination of index-based and signature-based methods, and assumes a sparse binary feature representation of objects in the collection. Although they only show results for the monolingual case, their approach seems to be applicable to the cross-lingual pairwise similarity problem.

We are aware of some work that attempts to solve the cross-lingual version of the problem, but in a slightly different setting. Anderka et al. [3] make the conclusion that both signature-based and index-based approaches require at least a linear scan of the collection due to the low similarity thresholds in cross-lingual pairwise similarity. However, the authors do not describe an algorithm or show any experimental results. In another recent paper, Platt et al. [112] describe techniques for projecting multi-lingual documents into a single vector space: the proposed training "encourages" similar document pairs to have similar vector representations. Mehdad et al. [94] view this problem as a textual entailment, and introduce an approach to solve cross-lingual textual entailment, focusing on the translation process more than the search algorithm. Snover et al. [132] solve a variant of this problem for domain adaptation in MT, where the goal is to find text comparable to a given sentence. The authors present a probabilistic relevance model to achieve this, and use the retrieved text to adapt the language and translation models of the MT system.

2.6 Extracting Parallel Text for MT

Section 2.1 provided detailed information on the importance of parallel text for statistical MT systems. As modern translation systems are heavily driven by training on large amounts of parallel text, it has become an essential task to exploit all available resources where parallel sentences may be found. The web is a rich source of parallel text, and has therefore attracted many researchers for mining purposes [1, 13, 35, 45, 93, 101, 102, 129, 117, 100, 157].

Before any discussion on how such data can be automatically obtained, we need to clearly define different levels of "parallel text", and describe the resources that might contain it. The following is a list of five rough categories of multi-lingual text, in decreasing order of comparability.

1. *Parallel*: Standard sentence-aligned parallel text, in which each sentence pair is assumed to be mutual translations For example, European parliament proceedings (Europarl) is a widely used parallel corpus for European languages.

2. *Noisy parallel*: Multiple translations of the same document, which is less parallel than above because the translator has a lot of freedom in making slight changes to the way content is expressed. However, we would expect sentences in one document to correspond to those roughly at the same position in the other language. Book translations are common examples to this category.

3. *Fully comparable*: Documents from different languages that describe the same event or subject. In this category, a document pair might differ slightly in length due to certain details being omitted in either document. As an example, we expect to see missing sentences within paragraphs. News stories generated in multiple languages is a good example for fully comparable documents.

4. *Semi-comparable*, also named *quasi-comparable* by Fung and Cheung [45]: Documents from different languages that contain portions describing the same event or subject, although the documents might have different subjects at a higher level. In this category, certain sections/chapters in one document might

not appear at all in the other, so it is expected that they might differ greatly in terms of length. Wikipedia contains many semi-comparable document pairs, in addition to fully comparable ones.

5. *Arbitrary multilingual text*: In this category, we cannot make any assumptions about the existence of parallel text. There is no constraint to the content of either document (e.g., web).

Gale and Church's seminal paper introduced a sentence alignment program based on a simple yet effective probabilistic model of character count ratios, supported by experiments on German-English and French-English data [48]. Their approach assumes a *noisy parallel* corpus as input, in which there are pairs of documents that are translations of each other but do not have sentence alignments. Some sentences might need to be aligned to multiple sentences in the other language, but there is a strong correspondence between the two documents. This is different than a *fully comparable* corpus, in which we only assume there are cross-lingual document pairs on the same subject, which might differ to a great extent.

Recent parallel text extraction approaches usually adopt a two-step process:

1. Identify similar documents and generate candidate sentence pairs.

2. Filter candidate pairs to retain parallel sentences.

The general solution to the first step involves computing pairwise similarities across multi-lingual corpora, which corresponds to the cross-lingual pairwise similarity problem described in Section 2.5. Since this is computationally intensive, most previous studies fall back to heuristics. Munteanu and Marcu [101] use temporal information, treating all news articles published at most five days apart as similar. Resnik and Smith [117] present the first paper with a working system that mines parallel text from the web, in which they use the Altavista search engine to find the same page in different languages within a specified web domain. As a result, these two approaches are designed to operate on collections with fully comparable documents. Smith et al. [129] exploit Wikipedia's "interwiki" links, that provide a linkage between similar articles written in different languages. This collection can be better categorized as between fully and semi-comparable, since these linked articles can still differ substantially in content. Fung and Cheung's approach [45] uses the TDT3 corpus, which they label as semi-comparable due to the off-topic content among the news transcriptions.

The second step (filtering candidate sentence pairs) usually involves a classification task, in which each candidate sentence pair is classified to be "parallel" or not. Due to the associated cost of classification, previous papers have introduced various heuristics to remove many candidate pairs without running a classifier. For example, Munteanu and Marcu [101] automatically discard pairs of sentences that have a length ratio above 2 (or below 0.5), and Smith et al. [129] follow this approach.

The usefulness of the parallel text depends on the scale of the input, as much as the accuracy of the algorithms. Resnik and Smith [117] mention that large web collections are not available, so they are able to find only 8,294 candidate Arabic-English page pairs in the Internet Archive, which is further reduced after duplicate filters. Nevertheless, adding the extracted parallel text to a CLIR system brings a 12% increase in MAP. Fung and Cheung [45] focus on precision, as opposed to scale, so their evaluation considers only 2,500 sentence pairs. According to a human evaluation, 67% of these are correct, which is 24% better than the baseline approach.

Munteanu and Marcu [101] build an end-to-end pipeline for parallel text extraction, finding 7.5 million pairs of news articles, corresponding to 17 million candidate sentence pairs, from which they generate a parallel corpus of 430 thousand sentence pairs. When evaluated intrinsically, their classifier yields very high precision (>95%) but low recall (67% when trained on same domain as test instances, 45% when trained on different domain). They also run extrinsic evaluation on MT, and results show that improvements are more substantial when there is less training data to start with, therefore it is better applied to low-resource languages.

More recently, Smith et al. [129] introduced a similar approach, aligning sentences in German, Spanish, and Bulgarian Wikipedia to English Wikipedia. They only consider existing "interwiki" links across Wikipedia collections, which are missing many links and contains some erroneous ones, but it provides 0.5 million German-English article pairs to start with. As a result, their approach extracts 1.7 million German-English (de-en), 1.9 million Spanish-English (es-en), and 146 thousand Bulgarian-English (bg-en) sentence pairs. report recall at 80% and 90% precision, which ranges from 69% and 59% (respectively) for de-en, 90% and 94% for bg-en, 72% and 82% for es-en.

As a summary, previous work either (i) lack the ability to scale to very large collections [101, 117], or (ii) present approaches that assume a certain structure and format of the input (e.g., news articles [101], Wikipedia pages with metadata [129], web pages from the same domain [117], or (iii) assume that the input corpus consists of a higher level of comparability (i.e., noisy parallel or fully comparable) than we would desire [117]. In addition, the approaches that actually run on fully or semi-comparable corpora do not show that the extracted parallel text significantly improves the quality of translation for general purpose MT, although results are encouraging when applied to low-resource [101] or domain adaptation [129] settings.

As part of Chapter 3, we describe an end-to-end pipeline that partially addresses all three of these issues. Our approach is efficient and scalable, and does not assume any inherent structure of the collection. As opposed to Smith et al. [129], our approach does not rely on manually linked Wikipedia articles. Although our approach may not apply to arbitrary multilingual text, experiments show that it can successfully extract parallel text from semi-comparable document pairs.

A recent study from Google also describes a solution to this problem without the three shortcomings mentioned above [141]. In their approach, all non-English web pages and books[6] are translated into English, thus transforming the problem into identifying similar monolingual document pairs. They use a hashing-based approach to efficiently identify similar document pairs, and then filter non-parallel sentence pairs based on an word alignment score. Despite the similarities, our approach makes several additional contributions: First, our work is the first to explore the effect of extracted data size on evaluation results. Our conclusions are a bit more nuanced than simply "more data is better," since there is a trade-off between quality and quantity. Even with very small amounts of text added, we can gain

[6]They run experiments on all books digitized as part of the Google Books project.

solid improvements in the task of MT (up to 3.5 BLEU points of increase). Overall, our approach requires far less computational resources and thus is within the reach of academic research groups: we do not require running an MT system on one side of the entire collection, which is orders of magnitude slower than our CLIR-based approach with little benefit in quality (see Chapter 3 for a detailed comparison between the two approaches). Furthermore, our novel two-stage classification approach yields an increase in the overall efficiency (speed) of the system with minimal loss in effectiveness (accuracy).

Another explored idea in bilingual text extraction is *bootstrapping*. After extracting sentence pairs from the input corpus, it is possible to improve the underlying translation model, and "bootstrap" from it: re-run the extraction to find better and more sentence pairs. With more data, the translation model of the cross-lingual pairwise similarity component has a larger vocabulary, allowing it to identify similarities that were ignored before.

We note that a similar bootstrapping idea was described by Fung and Cheung [45]. However, their approach only adds named entities, so the effect is limited. Furthermore, their manual evaluation only focused on precision, so it is unclear how useful the extracted bitext is for the task of MT. Munteanu and Marcu also showed the usefulness of bootstrapping from the extracted bitext, however their focus is on low-resource languages and domain adaptation, so improvements are measured over a small initial training dataset [101]. Callison-Burch and Osborne similarly show that training data can be generated from scratch, using a co-training procedure [19]. In contrast with previous work, we evaluate our bootstrapping approach on complete end-to-end MT, using a diverse set of language pairs, including both low- and high-resource language pairs.

Wikipedia has become especially popular for mining parallel sentences because of easy accessibility, availability in many languages, and its detailed link structure. One study shows how to exploit the "interwiki" links in Wikipedia to decide if two foreign-language sentences are parallel, based on how many of the relevant articles are linked to each other [13]. Smith et al. [129] also use linked Wikipedia articles as input to their classification algorithm. Other related work includes using Wikipedia to improve CLIR [103], and finding cross-lingual article pairs without explicit translation [114]. There has also been work on fixing links in Wikipedia, since these have found to contain many errors [32].

As a summary of papers related to parallel text extraction and sentence alignment, we suggest Tiedemann's recent book [137].

2.7 Scalability / MapReduce

With the large amounts of data available today, distributing computations has become an essential task. Traditional parallel programming approaches, such as Message Passing Interface (MPI) [42], require the developer to manage the many issues that arise during concurrent runtime (e.g., deadlock, race conditions, machine failures). As a result, a significant portion of development time is spent on handling system-level details. MapReduce [33] was introduced as an alternative, easy-to-understand model for distributed algorithms. It has become very popular in both industry and research, thanks to its simplicity and accessibility (through open-source implementations, such as Hadoop, Hive, etc.).

Often we observe that information processing algorithms take a certain structure: some operation is applied to each input record (e.g., one line of a file, or web page) to generate intermediate results, which are then aggregated into a final output. Motivated by the high-level abstraction in functional programming, a MapReduce program consists of a `Map` function, describing the operation over input records, and a `Reduce` function, which describes the aggregation process. Multiple workers may run the `Map` and `Reduce` functions synchronously, each called a *mapper* and *reducer*, respectively. The responsibility of a MapReduce developer is to describe these two functions, as part of the algorithm design. In MapReduce, processing is over key-value pairs, therefore another aspect of algorithm design is to decide what key-value types will be used as input and output of mappers and reducers.

The execution of a MapReduce program is as follows: Each mapper is applied to every input key-value pair to generate an arbitrary number of intermediate key-value pairs. The reducer is then applied to all values associated with the same key, and generates output key-value pairs. Every call to the `Map` (`Reduce`) function processes a single key-value pair, but a mapper (reducer) needs to call `Map` (`Reduce`) as many times as it receives input records. This two-stage processing structure is illustrated in Figure 2.7.

Running MapReduce requires a special distributed file system [52] in order to read input files and write the output. Once the algorithm is implemented as required and input data is uploaded to the file system, the underlying MapReduce

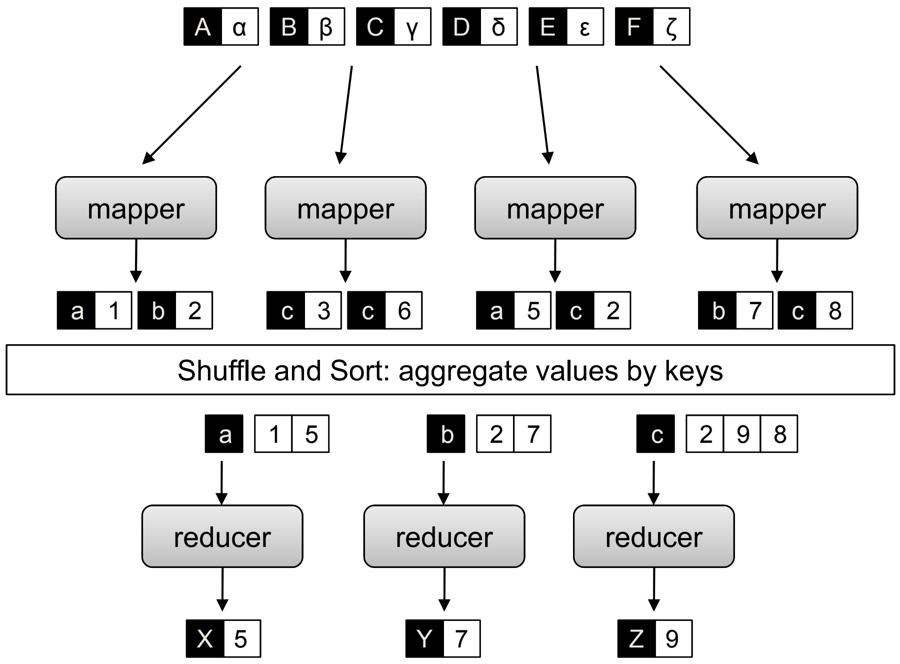


Figure 2.7: Overview of the two-stage MapReduce processing structure.

framework takes care of all execution details, such as synchronization, scheduling, and fault tolerance. One strength of MapReduce is its flexibility: the same algorithm can run on clusters ranging from few CPU cores to ones with thousands of them. Also, unlike traditional programming models like MPI, MapReduce does not have special hardware requirements. It is designed to run on many commodity computers, as opposed to few high-end servers. Another important feature of MapReduce is the focus on data locality, which aims to minimize the movement of data from one cluster node to another. Instead, the scheduling framework tries to move code to data as much as possible. As seen in Figure 2.7, MapReduce sorts records by their key values when moving data from mappers to reducers. Sorting data is the essence of many algorithms across all fields in Computer Science, therefore this external, distributed sort operation is very helpful. We refer interested readers to Lin and

Dyer's book on text processing in MapReduce for more details [81].

In this dissertation, we use Hadoop, which is an open-source implementation of the MapReduce programming model, and its accompanying open-source distributed file system, HDFS. The problem of pairwise similarity computation has been studied in MapReduce previously [80, 39, 143, 66]. However, these algorithms adopt an index-based approach, which stands in contrast to our signature-based approach. More recently, Bahmani et al. [6] introduced an LSH-based approach to pairwise similarity, in which they present an improvement to standard LSH, and report two orders of magnitude reduction in the amount of network traffic of their MapReduce implementation.

Chapter 3: **Searching to Translate: Efficiently Extracting Bilingual Data from the Web**

3.1 Overview

As described in Section 2.6, there are different levels of parallel text. A *semi-comparable corpus* is a pair of collections written in different languages, in which it is assumed that there are many *semi-comparable* cross-lingual document pairs. Documents that are semi-comparable share portions of text describing the same event or subject, although the documents might have different subjects at a higher level. Certain parts of one document might not appear at all in the other, so it is expected that they differ greatly in terms of length. On the other hand, a *parallel corpus* (also called *parallel text*, *bilingual text*, or *bitext* throughout the chapter), is a list of sentence pairs that are translations of each other. This type of corpora can be directly used for training machine translation systems, as opposed to comparable corpora.

In this chapter, we present an end-to-end pipeline, which takes a semi-comparable corpus as input and produces a parallel corpus as output. All experiments are conducted on Wikipedia: this multi-lingual encyclopedia with content in hundreds of languages is a great example of a semi-comparable corpus. The entire pipeline is implemented in Hadoop, the open-source implementation of MapReduce [33], which has recently emerged as a popular model for large-scale data processing. This allows the entire computation load to be parallelized to an arbitrary number of servers and scale up to web-scale collections.

Our processing pipeline runs in two phases: In the first phase, similar cross-lingual document pairs are found within the input comparable corpus (Section 3.2). The second phase extracts parallel sentence pairs from these cross-lingual document pairs (Section 3.3). We evaluate our approach on the task of machine translation (MT) in Section 3.4. Figure 3.1 illustrates an overview of the system components.

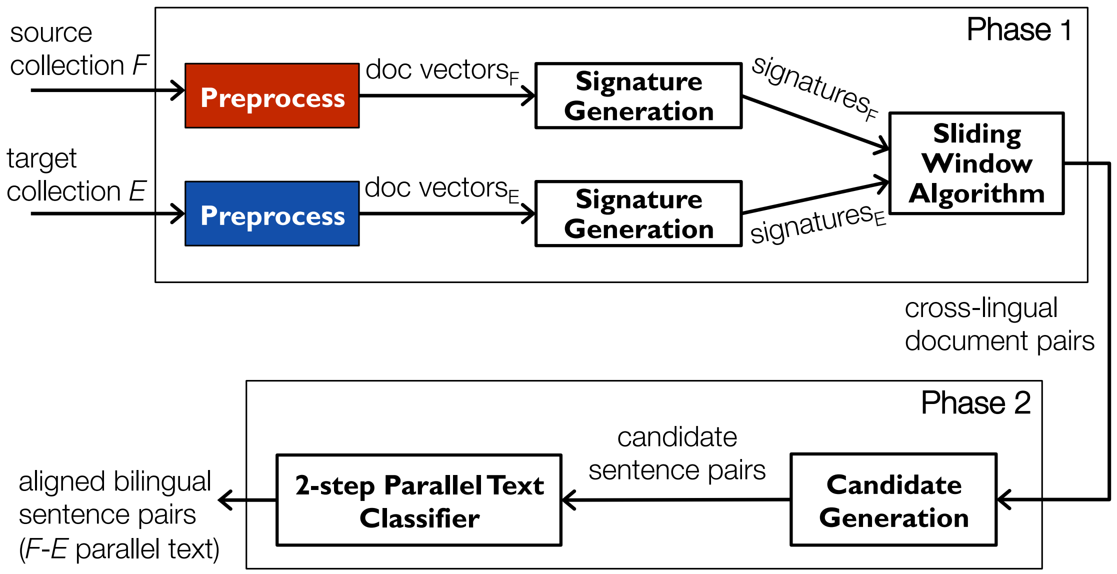


Figure 3.1: Overview of the end-to-end pipeline for parallel text extraction, starting from a comparable corpus, ending with aligned bilingual text (i.e., parallel corpus). F and E denote the two corpus languages.

3.2 Phase 1: Large-Scale Cross-Lingual Pairwise Similarity

3.2.1 Pairwise Similarity

In information retrieval (IR) applications, terms within a document are usually assumed to be independent, which allows us to represent documents by a vector-space model. In this model, a document d is represented as a vector of term weights $w_{t,d}$ (i.e., document vector), one for each term $t \in V$ that appears in d, where V is the collection vocabulary. There are, of course, many possible weighting schemes for $w_{t,d}$, from which we adopt BM25 [118], which has been successfully used in many retrieval tasks. In this scheme, each weight is a function of document frequency (df) and term frequency (tf) values, normalized by document length.

Given this representation, similarity between two document vectors u and v is computed via cosine similarity, a measure of the cosine of the angle between the vectors, which can be computed by the inner product of normalized vectors:

$$\text{Sim}(u, v) = \cos(\theta(u, v)) = u \cdot v/|u||v| \tag{3.1}$$

Note that the L_2 normalization in Equation 3.1 is on top of an L_1 normalization during the BM25 weight computation. The latter normalizes term frequency based on the document length, whereas the former directly normalizes the document vector by its magnitude. Since BM25 term weights are positive and vectors are normalized, the cosine similarity ranges from 0 (i.e., no similarity) to 1 (i.e., exact similarity).

The pairwise similarity problem is to find all similar pairs of documents (u, v) in a collection, where we define a pair to be similar if the cosine similarity is above a certain threshold (i.e., $\mathrm{Sim}(u, v) > \tau$). In *cross-lingual* pairwise similarity, the same problem needs to be solved when the vectors correspond to documents in different languages, thus introducing additional complexity.

There are two major challenges to this problem: matching vocabulary spaces, and searching for similar pairs. In the following two sections, we discuss our approach to overcome these two challenges.

3.2.2 Cross-Lingual Pairwise Similarity

Due to the difference in vocabulary spaces, document vectors in different languages are not directly comparable. As a result, we need to find a way to match the vocabulary spaces in this setting. We experimented with two different solutions to overcome the vocabulary mismatch: text translation and vector translation.

The first approach is to *translate document text* from one language into the other using an off-the-shelf machine translation system. For instance, if we have German and English documents, we could translate all German documents into English, and then perform pairwise similarity in the English vocabulary space. The advantage of this approach is that modern machine translation systems can produce high-quality results, especially for European languages. However, the disadvantage is that machine translation is typically time-consuming and resource-intensive. This approach has been implemented by Uszkoreit et al. [141].

The second approach is to *translate document vectors* from one language into another using cross-language information retrieval (CLIR) techniques. Adopting the approach proposed by Darwish and Oard [31], a document vector v_F in some source-language vocabulary space F can be projected into a document vector v_E in the target-language vocabulary space E by "translating" the df and tf values, which are then combined in the scoring function. In this approach, for every target-language term $e \in E$, we compute df_E and tf_E values as follows:

$$\mathrm{df}_E(e) = \sum_{f \in F} P(f|e) \times \mathrm{df}_F(f) \tag{3.2}$$

$$\mathrm{tf}_E(e,d) = \sum_{f \in F} P(f|e) \times \mathrm{tf}_F(f,d) \tag{3.3}$$

where d is some source-language document, and we assume df_F and tf_F values are represented in vector v. Since term weight is a function of df_E and tf_E values, we can compute the BM25 term weights for each $e \in E$ and construct a "translated" vector v_E. In Equations 3.2 and 3.3, the quantity $p(f|e)$ is the conditional probability of a source-language word f being the translation of a target language word e. This distribution captures lexical translation ambiguities, or the fact that words in one language can translate to multiple words in another language. The transformation in the equation can be interpreted as an averaging, weighted by conditional probability: each $e \in E$ generates f with probability $p(f|e)$. Following this probabilistic generative process, we end up with the formula above.

The word translation probabilities (i.e., $p(f|e)$) can be estimated from a parallel corpus using unsupervised word alignment techniques [144, 106], also used as the basis for statistical MT systems. This is a well-understood problem, and there exist many off-the-shelf implementations of these approaches (e.g., GIZA++). Due to incorrect word alignments and noise, words might have many translations with very low probabilities, therefore it is useful to incorporate the following filters to the probability distribution $P(f|e)$:

1. Discard low-probability translation candidate f, if $P(f|e) < L$.
2. Include translation candidates (starting from most probable candidate) until cumulative probability sum C, or the number of candidates reaches H. Discard the remaining ones, then re-normalize.

3.2.2.1 MT vs. CLIR for Cross-Lingual Pairwise Similarity

Typical instances of the monolingual pairwise similarity problem involve extracting pairs with high cosine similarities (e.g., 0.9 or higher for near-duplicate detection). For the cross-lingual case, we expect similar documents in different languages to have lower similarities, since the translation process is noisy.

In order to empirically determine appropriate similarity thresholds for the cross-lingual case, and to present an empirical comparison between the two translation approaches, we performed a preliminary analysis. We experimented with German and English portions of the Europarl corpus (version 6), which contains proceedings from the European Parliament (1996–2009). We constructed artificial "documents" by concatenating every 10 consecutive sentences into a single docu-

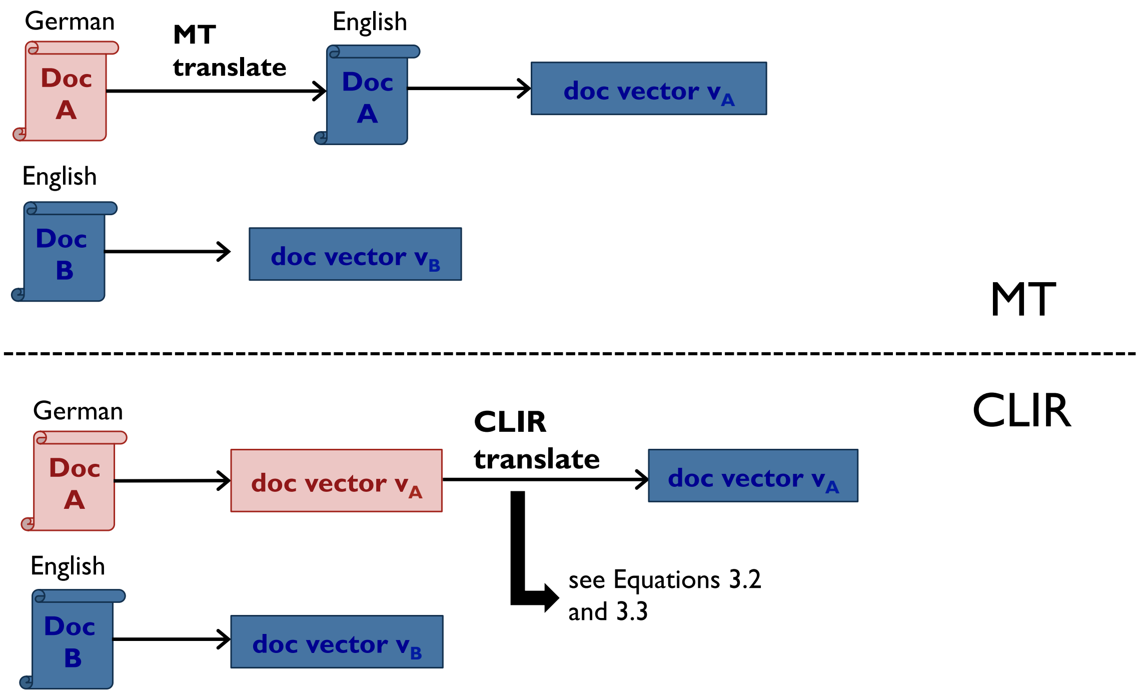


Figure 3.2: Two approaches to match vocabulary spaces in cross-lingual pairwise similarity: text translation (MT) and vector translation (CLIR).

ment. In this manner, we sampled 100 document pairs that are mutual translations of each other (and therefore semantically similar by construction). For each sample document, we created 99 non-parallel cross-lingual pairings, since each document is aligned to one document in the other language. This process provided ground truth to evaluate the effectiveness of the two translation approaches discussed above: text translation (using MT) and direct vector translation (using CLIR). In the first case, we translated each German document into English and then processed the resulting translated English document into vector form. In the second case, we first processed the German document into vector form, and then projected the vector over into English. See Figure 3.2 for an illustration of the two approaches. Finally, we computed the cosine similarities of each document pair, under both conditions.

For the MT approach, we experimented with two different systems: `cdec`, an open-source hierarchical PBMT system trained on Europarl,[1] and Google Translate, a popular commercial MT system.[2] We assess the effect of the translation method in Figure 3.3, and the effect of the underlying MT model in Figure 3.4.

In Figure 3.3, we plotted four distributions of cosine similarities, binned into intervals of 0.05. Striped columns correspond to negative (i.e., non-parallel) instances, using either CLIR translation (shown in red) or MT translation (with `cdec`, shown in blue), whereas solid columns correspond to positive (i.e., parallel) instances. We make two important observations. First, the cosine similarities are surprisingly low, especially since these are artificially created perfect document translations, and written in formal language. We expect even lower values in practice, because documents will almost always be comparable, not parallel. This observation points to a challenge in the cross-lingual pairwise similarity problem: if we want to mine comparable document pairs in different languages, we need to extract documents that have low cosine similarities. As we lower the threshold for cosine similarity, the task of distinguishing positive and negative instances becomes more difficult.

The second conclusion from Figure 3.3 is that the positive and negative instances are clearly separated with either translation approach. This means that cosine similarity is a good indicator of parallel text. Also, we observe that cosine similarity scores are higher on average for the MT approach, but vector translation seems reasonably competitive. Besides, the difference in translation quality becomes

[1]Portions used for testing were not used in training.

[2]A free API service is available at https://developers.google.com/translate

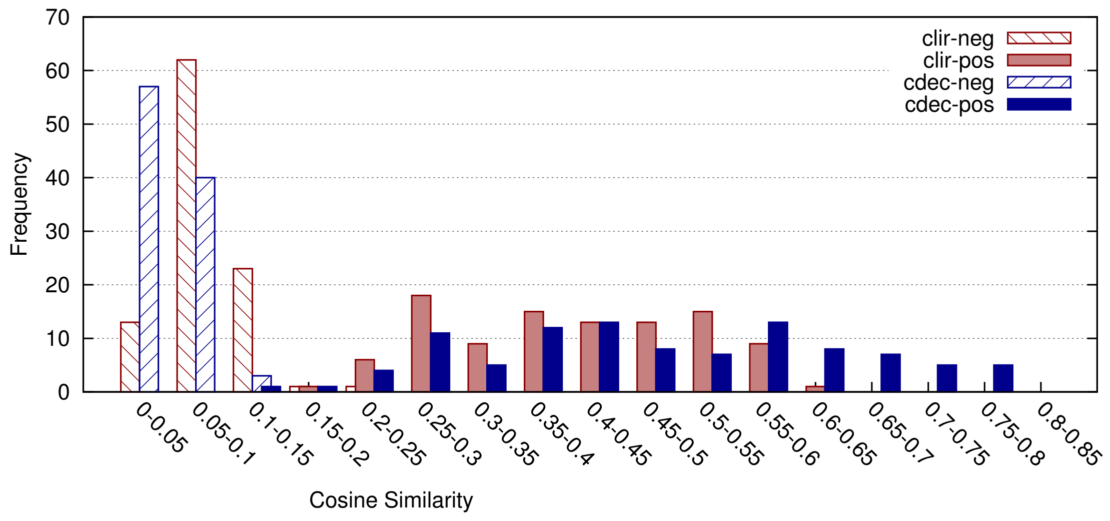


Figure 3.3: Comparing text (using the `cdec` MT system, labeled `cdec`) and vector (using CLIR, labeled `clir`) translation approaches for cross-lingual pairwise similarity, using data from German-English Europarl. This histogram illustrates that (1) similarity values for parallel documents are surprisingly low, (2) positive and negative instances are clearly separated with either translation method, and that (3) the MT approach generates slightly better results (although we argue that it is not sufficient to justify the 600 times slower running time).

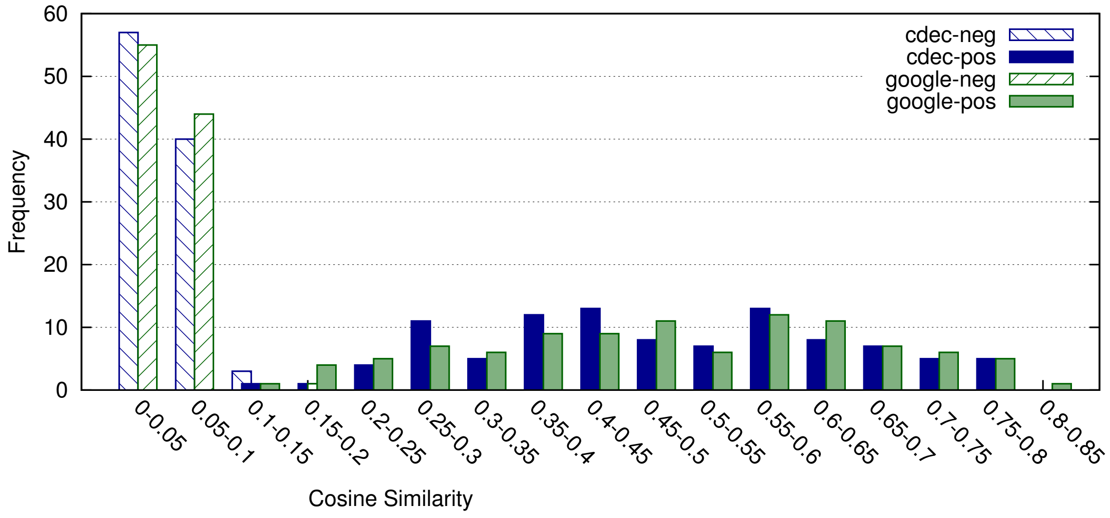


Figure 3.4: Comparing two text translation approaches (hierarchical PBMT system `cdec` and commercial MT system Google Translate) in cross-lingual pairwise similarity, using data from German-English Europarl. This histogram illustrates that there is no noticeable difference between using either MT system for this task.

even less noticeable if we consider the computational tradeoff: running an MT system on millions of documents is orders of magnitudes slower than vector translation. For example, running MT on German Wikipedia (35.4 million sentences) would require about 3147 CPU hours (roughly estimating from the per-sentence running time of `cdec`), which is much slower than the 5-hour running time with vector translation. Based on these arguments, we used the CLIR vector translation technique in our approaches.

As an additional analysis, we plotted the same four distributions, but this time comparing the two MT approaches. In this case, we observe that differences between the two cases are negligible. This might be surprising, since the Google Translate system has a much larger underlying vocabulary, as a result of having more training data. However, we do not see a difference here since we are evaluating on document pairs in the same domain (i.e., news) as the training data of the `cdec` MT system. For example, we would expect more significant differences if we were evaluating on blogs. Therefore, this result emphasizes that different MT approaches yield similar results as long as they have good vocabulary coverage.

3.2.3 Approach

Now that we have defined the problem and identified its challenges, we are ready to explain the first phase of the end-to-end bitext extraction pipeline: finding cross-lingual document pairs in a multi-lingual collection of comparable documents. Our approach is based on LSH, and consists of three steps. First, all documents are preprocessed into document vectors (Section 3.2.3.1). Second, document signatures are generated from document vectors (Section 3.2.3.2). Finally, a sliding window algorithm applied to the signatures finds all cross-lingual document pairs above a similarity threshold (Section 3.2.3.3).

3.2.3.1 Preprocessing Documents

In the preprocessing step, raw documents from both languages are parsed, tokenized (and stemmed for applicable languages), and represented as weighted document vectors. All terms that occur only once in the collection are removed from the vocabulary and each remaining term is converted into an integer for efficiency reasons. Document vectors of the source language are projected into the target language by the CLIR approach explained in Section 3.2.2. Thus, the collections in two languages are converted into a single collection of document vectors in the target language. Since we are only interested in cross-lingual pairs, we use document identifiers (docids) to determine the original language of a document vector.

3.2.3.2 Generating Document Signatures

At the core of any pairwise similarity algorithm is the similarity calculation between pairs of documents. The major drawback of cosine similarity is that the calculation requires a number of floating point operations linear to the number of terms in u and v, which is computationally expensive. Although it is possible to accurately approximate floating point values with integers using quantization methods, the similarity calculation still remains a bottleneck when dealing with many documents and large vocabularies.

Signatures are an efficient and effective way of representing large feature spaces. We explore three well-known methods to generate signatures from document vectors: random projection [21], Simhash [90], and Minhash [28]. Of course, this is only an approximation of the true cosine similarity, which is given by Equation 3.1, and we will discuss the preciseness of all three approximation methods at the end of this section.

Random projection (RP) signatures use a series of random hyperplanes as hash functions to encode document vectors as fixed-size bit vectors [21]. Let us assume a collection with vocabulary V, so that each document vector is from the domain $\mathbb{R}^{|V|}$. To obtain a signature of D bits using this approach, D randomly generated real-valued vectors of length $|V|$ are used to map each document vector u onto a RP signature $s_u \in [0,1]^D$. The i^{th} bit of s_u is determined by an inner product of u and the i^{th} random vector.

Given D random vectors $r_1, ..., r_D$, the signature s_u is computed as follows:

$$s_u[i] = h_{r_i}(u) = \begin{cases} 1 & \text{if } r_i \cdot u \geq 0 \\ 0 & \text{if } r_i \cdot u < 0 \end{cases} \tag{3.4}$$

Geometrically, the intuition behind this formula is that the hyperplane orthogonal to each random vector r_i splits the space into two: document vectors on the same side as r_i generate a positive inner product, whereas the inner product with vectors from the other side is negative.

The cosine similarity between two documents can be computed via hamming distance (i.e., number of bits that differ) between their signatures, according to the following relationship [21]:

$$\mathrm{Sim}(u,v) = \cos\left[\left(\frac{\pi}{D}\right)\mathrm{hamming}(s_u, s_v)\right] \tag{3.5}$$

Simhash signatures are essentially a "hash" of the document vector. This approach relies on a hash function that generates hash values for every term in the document vector. Given a document vector u and a hash function h that maps string terms to D-bit hash values, we generate a D-bit signature s as follows:

$$s_u[i] = \begin{cases} 1 & \text{if } \left[\sum_{\substack{\text{term } t \\ h(t)[i]=1}} u_t\right] + \left[\sum_{\substack{\text{term } t \\ h(t)[i]=0}} (-u_t)\right] > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where u_t corresponds to the weight of term t in document vector u. If the ith bit of t's hash value is 1, the term contributes its positive weight (i.e., u_t) to the ith bit of the "hash" value of the vector (i.e., $s[i]$), otherwise it contributes $-u_t$. If the sum of the term contributions to $s[i]$ is greater than 0, then $s[i]$ is set to 1, otherwise, it is set to 0. As with the RP signatures, Equation 3.5 gives the cosine similarity as a function of hamming distance between two Simhash signatures.

For this approach, signature quality depends on the quality of the hash function, which also dictates the signature length (therefore it cannot be set arbitrarily).

In this work, we use the hash function in Manku et al. [90], which was applied to detect near-duplicate web pages. There are more recent methods along similar lines that "learn" the hash of a vector, which are worth exploring in the future [154].

Minhash signature of a document vector u requires K random orderings of terms in u. Given a hash function, terms can be ordered by their hash value, or alternatively, terms can be ordered by a random permutation on the vocabulary. For each of the K orderings, the term in a document that has lowest order is picked as the "minimum hash" (minhash) of that document. The probability that two documents u and v have the same minhash term for a given ordering is $|u \cap v|/|u \cup v|$ (i.e., Jaccard similarity). This procedure is repeated for K different randomly selected orderings to reduce the risk of false positives. Consequently, the *Minhash* signature of a document vector consists of all K minhash terms. The similarity between u and v is computed by the proportion of "minimum hash" terms they share:

$$Sim(u, v) = (s_u \cap s_v)/K \tag{3.7}$$

Chum et al. [28] showed effectiveness of Minhash signatures on near-duplicate image detection.

In order to compare the precision of these three signature generation techniques, we selected a sample of 1064 document vectors from preprocessed German Wikipedia. For every pair of document vectors, we first computed the true cosine similarity (Equation 3.1), and then estimated values using each of the three signa-

Method	**Bits**	**Avg. Absolute Error**			**Time**
		> 0.0	> 0.1	> 0.2	(ms)
Minhash	64	0.061	0.168	0.215	0.28
Minhash	992	0.050	0.056	0.075	1.82
Simhash	64	0.236	0.299	0.273	0.25
RP	64	0.154	0.152	0.123	1.17
RP	100	0.124	0.122	0.101	2.03
RP	200	0.088	0.086	0.069	4.52
RP	500	0.056	0.055	0.045	10.93
RP	1000	0.040	0.039	0.033	20.91

Table 3.1: Absolute error in cosine similarity, averaged over 1064 sample signatures, is compared among three different methods and a range of sizes.

ture methods and varying signature lengths. In each case, we computed the average absolute difference between the true and estimated similarity value, shown in Table 3.1. For comparison, the last column shows the average time (in milliseconds) taken to generate one signature on a laptop with an Intel Core 2 Duo 2.26 GHz processor.

Number of bits can be set arbitrarily for the RP method, but Simhash signatures generated by the approach in Manku et al. [90] are 64 bits. On the other hand, since Minhash signatures are represented as a list of integers (corresponding to term ids), they take up 32 times (assuming 4-byte integers are used) more space than a Simhash or RP signature of the same length. For a fair comparison, we included 2-term (64 bits) and 32-term (992 bits) Minhash signatures in the evaluation.

We performed three sets of experiments: First, we computed error for all possible pairs, then filtered out pairs with similarity less than 0.1, and finally less than

0.2. Precision at very low similarities is not very important because relevant documents usually have values higher than 0.2 (based on Figure 3.3). We noticed that Minhash signatures perform worse as we filter out low-similarity pairs because it simply predicts 0.0 for many pairs. In contrast, RP signatures are more accurate when the cosine similarity is higher, an indicator of the robustness of the method.

In all cases, 64-bit RP signatures are more precise than Simhash signatures, and this becomes more apparent once the low-similarity pairs are filtered out. The only drawback of RP signatures is the amount of time necessary to generate them: an inner product is needed to determine every bit. The average time to create a 64-bit RP signature is about 5 times slower than other methods with the same number of bits and generating 1000-bit RP signatures is more than 10 times slower than 32-term (992-bit) Minhash signatures. However, the running time of the signature construction step is relatively brief compared to the actual pairwise similarity algorithm. Based on this analysis, we selected 1000-bit RP signatures for our experiments.

3.2.3.3 Sliding Window Algorithm

A naive solution to the pairwise similarity problem requires a quadratic search. In other words, the number of searched pairs is on the order of the square of the number of total objects. An approach with this scalability characteristic is not feasible in practice, as the available computational resources would not be able to handle the data increase. In fact, for web collections, linear scaling properties are essential for algorithms to be practical, therefore we turn to approximate search approaches.

The sliding window algorithm is based on LSH [15, 21, 63], which assumes that documents are represented as bit-signatures, such that two similar documents are likely to have similar signatures. The algorithm was first presented by Ravichandran et al. [116], although it is heavily based on ideas in Charikar's paper [21]. Let us first briefly describe the original algorithm before introducing our parallelized version using MapReduce.

First, the list of all document signatures in the collection is sorted according to their bits (starting from the most significant bit). In this sorted list, each signature is compared to its neighbors within a window of B signatures (i.e., sliding window). Based on the hamming distance, cosine similarity is computed for each compared pair (by Equation 3.5), and all pairs above a similarity threshold of τ are output by the algorithm. In order to increase recall, the unsorted collection is replicated and permuted with respect to a different permutation function Q times.[3] In each permuted version, the bits of each signature are permuted according to the corresponding (randomly generated) permutation function. Each permuted list is sorted before performing the sliding window comparison. A more detailed description is given in the following sections.

Based on LSH theory (see Section 2.4), higher similarity between documents leads to higher probabilities of being closer in a sorted list. Given a large enough

[3]A permutation function is basically a permuted list of the integers $[1 \ldots D]$, where D is the signature size.

Q and B, it can be shown that signatures of similar documents will be at most B positions apart in at least one table with very high probability. Since sorted order depends on the positions of bits, having multiple permutations of the same signature increases the chance of retrieving a similar pair.

We first describe a straightforward parallel implementation of the algorithm in Section 3.2.3.4. Then, we discuss several modifications to exploit the MapReduce framework and take advantage of greater parallelism in Section 3.2.3.5.

3.2.3.4 Basic MapReduce Implementation

A demonstration of the sliding window algorithm on a toy example is given in Figure 3.5, based on the basic MapReduce implementation described in this section. After describing our implementation, we will go over the example in Figure 3.5 for clarification.

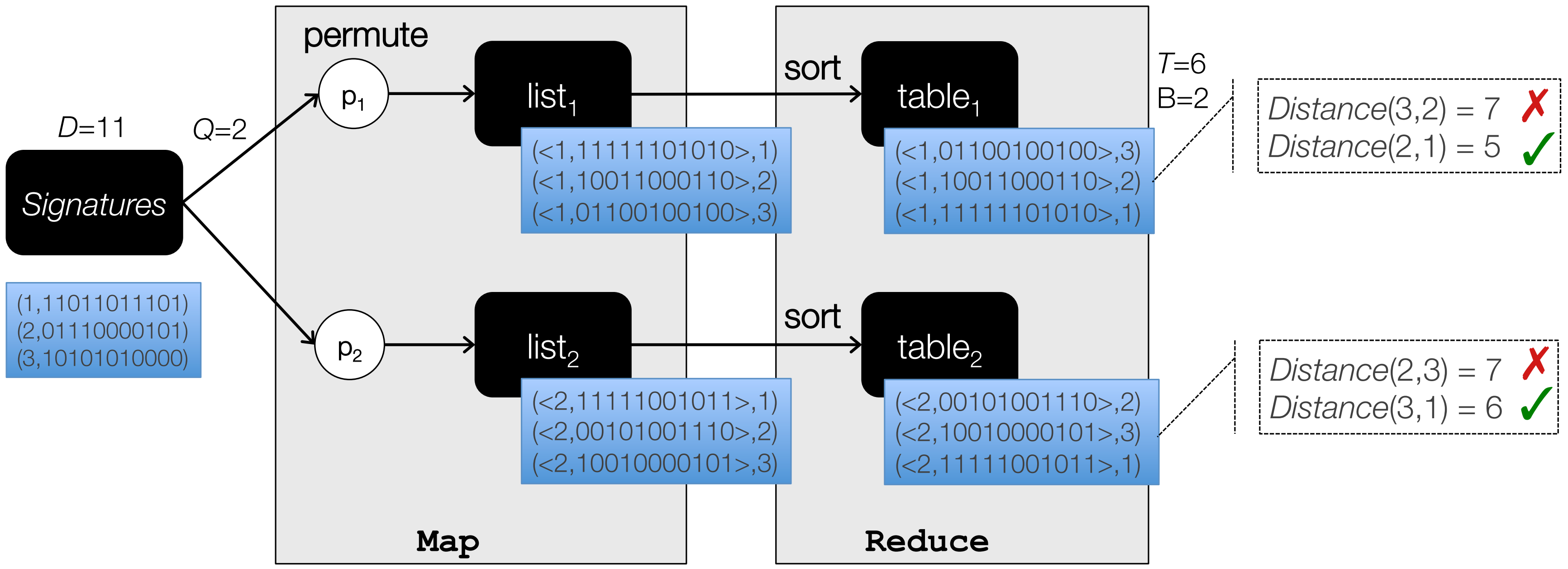


Figure 3.5: An example demonstration of the basic MapReduce implementation of the sliding window algorithm, assuming a toy collection with three signatures and parameters as follows: D=11 (i.e., number of bits), Q=2 (i.e., number of permutations), B=2 (i.e., window size), T=6 (i.e., hamming distance threshold).

In our MapReduce design, Q random permutation functions $p_1, ..., p_Q$ are created as a preprocessing step. Each document in the collection is encoded as a (docid, signature) pair, and input to the MapReduce program. Each mapper takes a docid n and signature s as input, and emits an intermediate key-value pair $(\langle i, s^i \rangle, n)$ for each permutation function $p_i, i = 1...Q$.[4]

Each key sent to a reducer is a pair of the permutation group number i and the signature s (denoted as $\langle i, s \rangle$) and the values are docids sharing that pair (i.e., docid of all documents such that the ith permutation of its signature is s).[5] The reduce step is designed so that all keys with the same permutation group number are sent to the same reducer (by appropriately partitioning the key space), and they are sorted according to the signature bits (comparison of two signatures is from most significant to least significant bit). Therefore, each of the Q reducers receives a sorted list of permuted signatures, paired with respective docids, which we call a *table*. At this point of the algorithm, we essentially have Q different versions of the original list of signatures (i.e., tables), each corresponding to a different permutation.

A queue of size B is used to implement the sliding window idea (Reduce method in Figure 3.6): in each table, we compare all signatures that are at most B positions away from each other. The reducer iterates over the list of signatures and keeps the last B in the queue (lines 13–17). At each iteration, the current signature is compared to all signatures in the queue, and docid pairs are emitted for all pairs in which the cosine similarity is above τ (lines 8–12). Since we are

[4]We denote s^i as the permutation of s with respect to the permutation function p_i, and refer to it as the ith permutation of s.

[5]Although, it is very rare that two documents have the exact same signature when the number of bits is high.

```
Mapper
1: method MAP(docid $n$, signature $s$)
2:     for all permute func $p_i \in [p_1, p_2, \ldots, p_Q]$ do
3:         EMIT($\langle i, s.$PERMUTE$(p_i)\rangle, n$)
Reducer
4: method INITIALIZE
5:     $docids \leftarrow$ new QUEUE($B$)
6:     $sigs \leftarrow$ new QUEUE($B$)
7: method REDUCE($\langle$permno $i$, signature $s\rangle$, docids $[n_1, n_2, \ldots]$)
8:     for all docid $n \in [n_1, n_2, \ldots]$ do
9:         for $j = 1$ to $sigs$.SIZE() do
10:            $distance \leftarrow s$.DISTANCE($sigs$.GET($j$))
11:            if $distance \leq T$ then
12:                EMIT($\langle docids$.GET$(j), n\rangle, distance$)
13:        $sigs$.ENQUEUE($s$)
14:        $docids$.ENQUEUE($n$)
15:        if $sigs$.SIZE() $> B$ then
16:            $sigs$.DEQUEUE()
17:            $docids$.DEQUEUE()
```

Figure 3.6: Pseudo-code of the initial version of the sliding window algorithm.

comparing signatures, we output pairs based on the hamming distance, in order to avoid computing the cosine similarity. The cosine similarity threshold τ is converted into a distance threshold, by solving Equation 3.5 for hamming distance: $T = D \times cos^{-1}(\tau)/\pi$.

In the example shown in Figure 3.5, three 11-bit signatures are input to the algorithm, and two permuted versions of each signature are generated, with respect to permutation functions p_1 and p_2. The corresponding key-value pairs are sent to the reducers. The reducer(s) receives two tables: one contains the list of three signatures permuted with p_1, in sorted order of their bits; the other table contains signatures permuted by p_2. When processing the first table in Figure 3.5, a window of size 2 allows the comparison of two pairs: signature of document 3 and 2 are compared, as well as documents 2 and 1. The first comparison yields a hamming distance of 7 bits, which is above the threshold, therefore (3,2) is not emitted. The second pair has a distance of 5, which is therefore output as similar. The second table also makes two comparisons, (2,3) and (3,1), and outputs the latter one. As a result, the algorithm makes 4 comparisons and output the only two pairs that have a hamming distance not greater than 6.

3.2.3.5 Improved MapReduce Implementation

A drawback of the basic algorithm is the inability to increase the number of reducers, because each reduce group processes a single permuted list of all the signatures in the collection. In other words, in the reduce phase, only Q reducers can operate in parallel, which may underutilize resources in very large clusters that are commonly used for MapReduce. In this case, being able to increase the number of reducers arbitrarily can help the algorithm scale better.

We achieved this property by modifying the previous algorithm in the following way. The map function is exactly the same, but the reduce step divides each table into an arbitrary number of consecutive blocks (Figure 3.7). Each of the Q reducers iterates over sorted signatures, stores them in a list (lines 9–11), and outputs the list when its size reaches a pre-specified number, M (lines 12–14). We denote each of these M-sized lists as a *chunk*, and refer to this as the *chunk generation* phase. The actual comparison and similarity computation is performed in an additional map-only MapReduce task (Figure 3.8), called the *detection* phase, where the input

```
Mapper
1: method Map(docid $n$, signature $s$)
2:     for all permute func $p_i \in [p_1, p_2, \ldots, p_Q]$ do
3:         Emit($\langle i, s.\text{Permute}(p_i)\rangle, n$)
Reducer
4: method Initialize
5:     $docids \leftarrow$ new List
6:     $sigs \leftarrow$ new List
7:     $chunk \leftarrow$ new SignatureChunk
8: method Reduce($\langle$permno $p$, sig $s\rangle$, docids $[n_1, n_2, \ldots]$)
9:     for all docno $n \in [n_1, n_2, \ldots]$ do
10:        $sigs$.Add($s$)
11:        $docids$.Add($n$)
12:        if $sigs$.Size() $= M$ then
13:            $chunk$.Set($sigs, docids$)
14:            Emit($p, chunk$)
15:            $sigs \leftarrow sigs$.SubList($M - B + 1, M$)
16:            $docids \leftarrow docids$.SubList($M - B + 1, M$)
17: method Close
18:     $chunk$.Set($sigs, docids$)
19:     Emit($p, chunk$)
```

Figure 3.7: Pseudo-code of the *chunk generation* phase of the sliding window algorithm.

```
Mapper
1: method Map(permno $p$, signatureChunk $chunk$)
2:     for $i = 1$ to ($chunk$.Size() $- B$) do      ▷ $B$ is the size of the sliding window
3:         for $j = (i+1)$ to $(i+B)$ do▷ only if $i$ and $j$ are from different languages
4:             $distance \leftarrow chunk$.Signature($i$).Distance($chunk$.Signature($j$))
5:             if $distance \leq T$ then
6:                 Emit($\langle chunk$.Docid($j$), $chunk$.Docid($i$)$\rangle$, $distance$)
```

Figure 3.8: Pseudo-code of the *detection* phase of the sliding window algorithm.

is the chunks and each mapper finds all similar pairs in one chunk. Only pairs that have a distance less than the threshold T are emitted by the algorithm. We use the docids to make sure only cross-lingual pairs are included in the output. In this phase, there can be as many mappers as there are chunks, which allows full utilization of computational resources.

Note that some comparisons will be missed at the boundary of chunks (e.g., the last element of a chunk is not compared to the first element of the next chunk). We address this issue by appending the last B signatures of the previous chunk to the beginning of the current chunk (Figure 3.7, lines 15–16). This results in redundancy in the chunks emitted to disk, because the last B signatures of a chunk are the same as the first B signatures of the next one. Thus, the total number of redundantly stored key-value pairs are (*number of chunks* x B), or $\frac{NB}{M}$, which is negligible assuming that $B \ll M$.

3.2.4 Analytical Model

In this section, we provide a theoretical analysis of our LSH algorithm, which provides a model for estimating effectiveness analytically. The starting point is Equation 3.4, which shows how random vectors are used to generate bit signatures. For any two vectors u and v, the probability that a single random projection h_r collides is given by the following:

$$\Pr[h_r(u) = h_r(v)] = 1 - \frac{\theta(u,v)}{\pi} \tag{3.8}$$

The proof can be found in [54], which we omit here, but the intuition is that the hyperplane defined by the random vector divides the collection of vectors into two disjoint sets, and the probability that any two vectors land in the same set is determined by the angle θ between them, according to the relation above.

Let $S_{\mathsf{0}} = \{u | h_r(u) = 0\}$ and $S_{\mathsf{1}} = \{u | h_r(u) = 1\}$, corresponding to the subsets of our collection of vectors that share the same hash value. Given a vector u, we can compute $h_r(u)$ and perform a linear scan in the corresponding set to find similar vectors. Given that vectors u and v have cosine similarity t, the probability that they are in the same subset is given by $1 - \cos^{-1}(t)/\pi$.

We can extend this basic approach by selecting n random vectors $\{r_1, r_2, ...r_n\}$ and construct corresponding hash functions $\{h_{r_1}(u), h_{r_2}(u), ...h_{r_n}(u)\}$. Using these hash functions we can divide up the collection into 2^n disjoint sets:

$$S_{\mathsf{000...0}} = \{u | h_{r_1}(u) = 0 \wedge h_{r_2}(u) = 0 \wedge ...h_{r_n}(u) = 0\}$$

$$S_{\mathsf{000...1}} = \{u | h_{r_1}(u) = 0 \wedge h_{r_2}(u) = 0 \wedge ...h_{r_n}(u) = 1\}$$

...

$$S_{\mathsf{111...1}} = \{u | h_{r_1}(u) = 1 \wedge h_{r_2}(u) = 1 \wedge ...h_{r_n}(u) = 1\}$$

Suppose we wish to find document vectors that are similar to u: we can apply the hash functions to determine the correct candidate set of vectors, and then perform a linear scan through all those candidates to compute the actual inner product. With n random projections, the probability that we will find similar vectors becomes $(1 - \cos^{-1}(t)/\pi)^n$. More random projections reduce the number of comparisons we must make, but at the cost of compounding the error introduced by each random projection.

To alleviate this issue, we can repeat the entire process m times using m sets of n different random projections. In this case, the probability that we identify a valid similar pair becomes the following:

$$\begin{aligned}
&\Pr[u, v \text{ in same set in at least one trial} \mid \cos(u, v) = t] \\
&= 1 - \left[1 - \left[1 - \frac{\cos^{-1}(t)}{\pi}\right]^n\right]^m \qquad (3.9)
\end{aligned}$$

The above equation quantifies the error when searching for similar vectors using LSH. The second source of error is the hamming distance computation. As shown in Section 3.2.3.2, cosine similarity can be estimated from the hamming distance with Equation 3.5.

We use this similarity estimate to efficiently decide which documents to return during the linear scan of the candidate set. For a given vector u and similarity threshold τ, we calculate the corresponding hamming distance threshold T using Equation 3.5 and return all candidates with hamming distance less than or equal to T. The precision of this decision is given by the following:

$$\begin{aligned}
&\Pr[\text{hamming}(u, v) \leq T \mid \cos(u, v) = t \wedge u, v \text{ share } n\text{-bit prefix}] \\
&= \sum_{i=0}^{T} \Pr[\text{hamming}(u, v) = i \mid \cos(u, v) = t \wedge u, v \text{ share } n\text{-bit prefix}] \\
&= \sum_{i=0}^{T} \binom{D-n}{i} \rho^i (1-\rho)^{D-n-i} \qquad (3.10)
\end{aligned}$$

where $\rho = \frac{\cos^{-1}(t)}{\pi}$ is the probability that one bit of u and v are different.

How does this relate to our sliding window algorithm? Let us first consider a variation of our algorithm. Suppose instead of applying a sliding window B over the sorted bit signatures, we performed a pairwise N^2 comparison of all bit signatures that share n prefix bits. Another way to think about this is to dynamically adjust B so that it encompasses only those signatures that share the prefix. In this case, the number of tables (permutations) in our algorithm corresponds to the number of trials m and the prefix length corresponds to the number of random projections n in the above analysis.

Putting all of this together, the expected probability of extracting a pair of vectors with similarity t (hamming distance $\leq T$) is quantified by the product of Equations 3.9 and 3.10 above:

$$\begin{aligned}&\Pr[\text{hamming}(u, v) \leq T \text{ in at least one table} \mid \cos(u, v) = t] \\ &= \left(1 - \left[1 - \left[1 - \frac{\cos^{-1}(t)}{\pi}\right]^n\right]^m\right) \sum_{i=0}^{T} \binom{D-n}{i} \rho^i (1-\rho)^{D-n-i} \qquad (3.11)\end{aligned}$$

The probability of successfully extracting a similar pair has contributions from two sources: LSH (the two documents sharing the same signature prefix) and accuracy of hamming distance.

In actuality, we scan a fixed window B, so we need to approximate the common prefix length for the purpose of this analysis. Assuming that there are C vectors in total, C/B is the number of windows we split the signatures into. As these signatures are sorted from left to right, we can approximate the prefix length to be between $\lfloor \log_2(C/B) \rfloor$ and $\lceil \log_2(C/B) \rceil$. We use both estimates to obtain a range of the estimated recall values. Although this is merely a rough model, it provides us with a basis for analytically estimating effectiveness, which we demonstrate later.

3.2.5 Experimental Evaluation

We ran our cross-lingual pairwise similarity algorithm on six language pairs, where the target language is fixed to English: German-English (de-en), Spanish-English (es-en), Chinese-English (zh-en), Arabic-English (ar-en), Czech-English (cs-en), and Turkish-English (tr-en). Since our end goal is to improve MT quality, we selected a diverse set of languages: (a) from high-resource languages for which we expect better performance in MT evaluation tasks (e.g., German, Spanish, Arabic) to under-explored languages with limited resources (e.g., Czech, Turkish), and (b) from European languages closely related to English (e.g., German and Spanish), to languages that are linguistically challenging and may require special processing (e.g., segmentation in Chinese and Arabic, morphological analysis in Turkish).

In this section, an intrinsic evaluation of the cross-lingual pairwise similarity algorithm is presented using English and German Wikipedia only, selected because they are the largest Wikipedia collections available and because significant amounts of parallel corpora exist for the language pair. Experimental results for all six language pairs are presented in Section 3.3, as part of the end-to-end extrinsic MT evaluation.

We used the German Wikipedia dump from 1/31/2011, which contains 2.41

million articles totaling 8.5 GB. We used the English Wikipedia dump from 1/15/2011, which contains 10.86 million articles totaling 30.6 GB. For both collections we discarded redirect pages and stub articles, after which we were left with 1.68 million German and 3.57 million English articles.

Our system is implemented on top of Ivory, an open-source Hadoop toolkit for web-scale information retrieval [82]. German articles were projected into English document vectors, using Equation 3.2, with filtering parameters L (probability lower bound), C (cumulative probability threshold), and H (number of candidates) fixed to 0.05, 0.95, and 15. For translation probabilities $P(f|e)$ and $P(e|f)$, we trained word alignments using GIZA++ [106] on the Europarl German-English corpus, containing 2.08 million sentence pairs from European parliament speeches. For tokenization, we used the Java-based OpenNLP toolkit.[6] After projection, document vectors containing fewer than five terms were discarded, resulting in 1.47 million German articles and 3.44 million English articles. In the next step, RP signatures were generated from document vectors. Finally, we ran the improved sliding window algorithm (Section 3.2.3.5) to extract all document pairs with cosine similarity above a specified threshold. We only output cross-lingual pairs (i.e., one document is from German Wikipedia and the other is from English Wikipedia).

All experiments described in this section were conducted on a Hadoop cluster (running Cloudera's distribution) with 16 nodes, each having 2 quad-core 2.2 GHz Intel Nehalem Processors, 24 GB RAM, and three 2 TB drives.

[6]http://opennlp.sourceforge.net

3.2.5.1 Effectiveness vs. Efficiency

The parameters in our algorithm are D (length in bits of each signature), Q (number of tables), M (chunk size), B (the window size), and T (hamming distance threshold). We fixed D to 1000 based on the discussion in Section 3.2.3.2, and fixed M to 0.49 million so that each table is split into 10 chunks. For the rest of the parameters, we explored a number of choices and observed the changes in efficiency and effectiveness. Note that we need to run the preprocessing step and signature generation step only once for these experiments (this takes 1.67 hours). The *chunk generation* phase of the sliding window algorithm must be executed once for every value of Q (this takes 0.28, 0.53, and 0.75 hours, respectively, for $Q = 100, 200, 300$). The *detection* phase must be run for every different set of the remaining parameters, for which running times on the entire German-English Wikipedia collection are plotted in Figure 3.9 (varying window sizes for each value of Q). The regression lines clearly show that the algorithm scales linearly as Q and B increase ($R^2 > 0.999$ in all three cases). Note that the hamming distance threshold T does not affect the running time, but only determines which pairs are extracted, so it is not included in this figure.

To assess effectiveness, we selected a sample of 1064 German Wikipedia articles. For every article, we calculated its true cosine similarity with all other documents in English Wikipedia. All pairs above threshold $\tau = 0.3$ are treated as the ground truth.[7] This corresponds to a hamming distance threshold of $T = 400$ for

[7]Threshold was selected based on the similarity distribution in Figure 3.3.

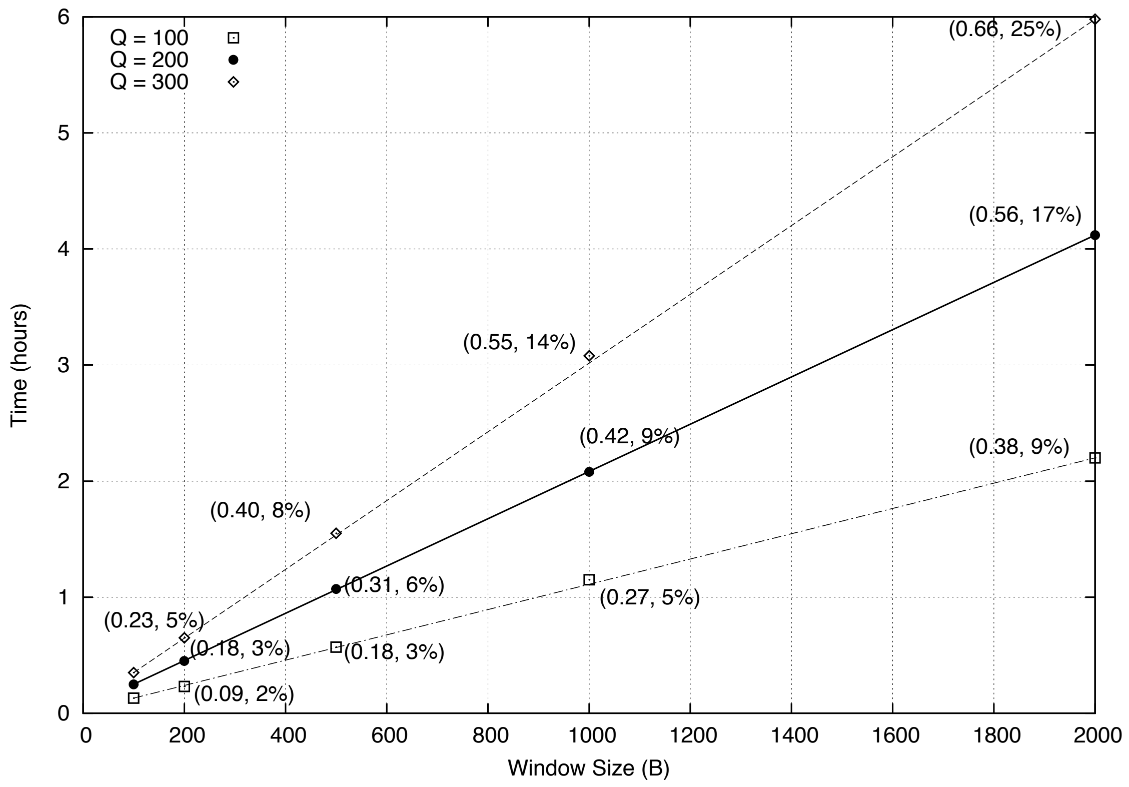


Figure 3.9: Running times of the detection phase of the sliding window LSH algorithm for $\tau = 0.3$, using 1000-bit signatures. Data points are annotated with recall and relative cost.

1000-bit signatures, since $\cos(400\pi/1000) \sim 0.3$. Note that a hamming distance of 500 corresponds to no similarity at all, so we are looking for documents that may be very dissimilar. From this, we are able to measure the quality of the pairs extracted by our algorithm. We argue that recall is the most important metric for evaluation (i.e., the fraction of pairs in the ground truth set that are actually extracted), since the task is recall-oriented by definition. Precision can be increased with post-processing: we can always filter the extracted pairs with more sophisticated techniques and discard results that fall below the similarity threshold. The time necessary for this is negligible compared to the time for extracting the pairs to begin with. For instance, with parameters $Q = 300, B = 2000$, the number of extracted pairs is 64 million, 0.0013% of all possible cross-lingual pairs in the collection. This gives an idea of how relatively less work remains for a possible post-processing step after running our the algorithm. Moreover, from an application point of view, it may not even be necessary to filter, since our τ threshold of 0.3 is somewhat arbitrary, and pairs with lower similarity scores may nevertheless be useful (see Figure 3.3).

Data points in Figure 3.9 are annotated with recall and a measure we call *relative cost*, defined as follows: for each condition, we can analytically compute the total number of similarity comparisons (i.e., in terms of hamming distance) necessary over the entire run. The cost of the chunk generation process is also taken into account in these calculations (by translating that time into an equivalent number of comparisons). We can then express this as a percentage of the total number of comparisons that a *brute-force approach* would require, which is the product of the size of the two collections: with 3.44 million English articles and 1.47 million German articles, this equals 5.05 trillion comparisons. A relative cost lower than 100% means we are "saving work" by not considering pairs unlikely to be similar, which is the point of using LSH. Each pair in the graph displays a point in the tradeoff space: the level of recall and efficiency that can be expected for a particular parameter setting.

To better understand these results, it is necessary to quantify the upper bound on effectiveness. Note that our algorithm has two sources of error: those introduced by the bit signatures (cf. Table 3.1) and those introduced by the sliding window algorithm (i.e., failure to consider the pair as a candidate). The former error is due to representation, whereas the latter is due to search. In order to isolate the search error, we can compute an effectiveness upper bound by repeating the procedure that generated the ground truth (i.e., brute-force approach), but using signatures instead of document vectors. For every sample document's signature, we computed the hamming distance with all other signatures in English Wikipedia and extracted all pairs that have a distance less than $T = 400$. We then compared these pairs to the ground truth set (i.e., brute-force on document vectors) and obtained 0.76 recall.[8] This means that the 0.04 absolute error in cosine similarity (see Table 3.1) introduced by 1000-bit random projections, which is negligible for other tasks that adopt high similarity thresholds (e.g., [116]), has a large impact for our task. Another way to understand this is that absolute error has an increasing impact as the similarity threshold is lowered (i.e., 0.04 absolute error is equal to 13% relative error when we are dealing with cosine similarity values of 0.3). From this analysis, it is important to understand that the upper bound on recall for our sliding window algorithm is not a perfect 1.0, but a more modest 0.76.

The expression "no free lunch" best characterizes our conclusion from the experimental results. There does not appear to be a single optimal solution to the cross-lingual pairwise similarity problem using LSH. Gains in efficiency inevitably come at a cost in effectiveness, and the extent to which it is worthwhile to trade off one for the other depends on application- and resource-specific constraints (i.e., how much recall is necessary, how much computational resources are available, etc.).

[8]Based on micro-averaging, where the denominator is the sum of ground truth pairs across *all* sample articles.

Representation	**Time** (ns)	**Precision**	**Recall**
1000-bit	28	0.59	0.76
2000-bit	52	0.74	0.81
3000-bit	84	0.86	0.78
Doc Vector	450	-	-

Table 3.2: Comparing different representations: average time for a similarity comparison is shown with respect to the effectiveness achieved on Wikipedia.

We provide a guide that helps the application developer locate a desirable operating point in the solution space.

We additionally explored the impact on upper bound effectiveness of different representations. Increasing the number of random projections produces longer and therefore more precise signatures, at the cost of linearly increasing running times. This is shown in Table 3.2, where we repeated the same set of experiments with 2000- and 3000-bit signatures and showed the average time (in nanoseconds) to process a single pair. Increasing signature length, however, does not appear to have much of an impact on recall. As a reference, we also show results with the original document vector. Even with 3000 bits, hamming distance calculations are still more than 5 times faster than computing similarity directly on document vectors (which require floating point operations). Once again, the motto is "there is no free lunch": representations that support faster similarity comparisons sacrifice precision.

3.2.5.2 Parameter Exploration

Since there are many tradeoffs within the parameter space of this algorithm, it would be desirable to explore this space more exhaustively, beyond the experiments in Figure 3.9. However, the long running times of large-scale experiments is a major obstacle. Nevertheless, it is possible to separately evaluate effectiveness and efficiency in a meaningful way. To compute efficiency, we quantify the amount of work required by our sliding window algorithm by the number of comparisons. This measure is convenient since it can be computed analytically without actually running the experiments, and arguably even better than (wall clock) running time since it abstracts over hardware differences and other natural variations. Since the results in Figure 3.9 confirm that our algorithm scales linearly (with respect to the parameters we tested), we can derive a straightforward mapping from number of comparisons to actual running time. To compute effectiveness, we can greatly speed up the experiments by considering only documents in the test set.

The results of separately evaluating effectiveness and efficiency are shown in Figure 3.10, where we vary the number of tables Q for $B = 200, 500, 1000$. Once again, this characterizes the tradeoff space by showing recall and relative cost (labeled on the points) that is realized by a particular parameter setting.

Figure 3.10 also shows recall estimates based on our analytical model from Section 3.2.4. As previously discussed, we bound n between $\lfloor \log_2(N/B) \rfloor$ and $\lceil \log_2(N/B) \rceil$, where N is the total number of document vectors in the collection: these are shown in the figure as dotted lines (but note overlaps). Despite the simplifications made in the analytical model, we see that the estimates are fairly good. These rough estimates allow the application developer to tackle arbitrary collections, and, *without performing any actual experiments*, estimate the effectiveness that can be achieved with different parameter settings. Combined with the ability to compute efficiency analytically, as we have shown above, the developer can characterize the tradeoff space with minimal effort. Such an analysis addresses a general weakness of LSH approaches: they require setting a large number of seemingly arbitrary parameters, without much guidance for a developer approaching new problems. Therefore, the ability to quantify efficiency and analytically estimate effectiveness provides a powerful tool for developing scalable and distributed approaches to computationally intensive problems.

To illustrate the importance of such analyses, we ran more experiments to analyze the tradeoff space of our cross-lingual pairwise similarity solution in Figure 3.11. By separating effectiveness and efficiency measures as above, we extend our analysis even further with a wider setting of parameters. In this figure, the thick horizontal line indicates the number of comparisons required for the brute-force approach, and each data point is annotated with actual recall. We observe that the same recall level can be achieved with different settings, presenting a possible tradeoff between the two main parameters of our algorithm, number of tables and window size. This particular analysis is dependent on the type of underlying hardware that can be used to run our algorithm. Since the number of tables (Q) influences the degree of parallelism, this implies that larger clusters with more processors may benefit from larger values of Q to increase cluster utilization. On the other hand, a smaller

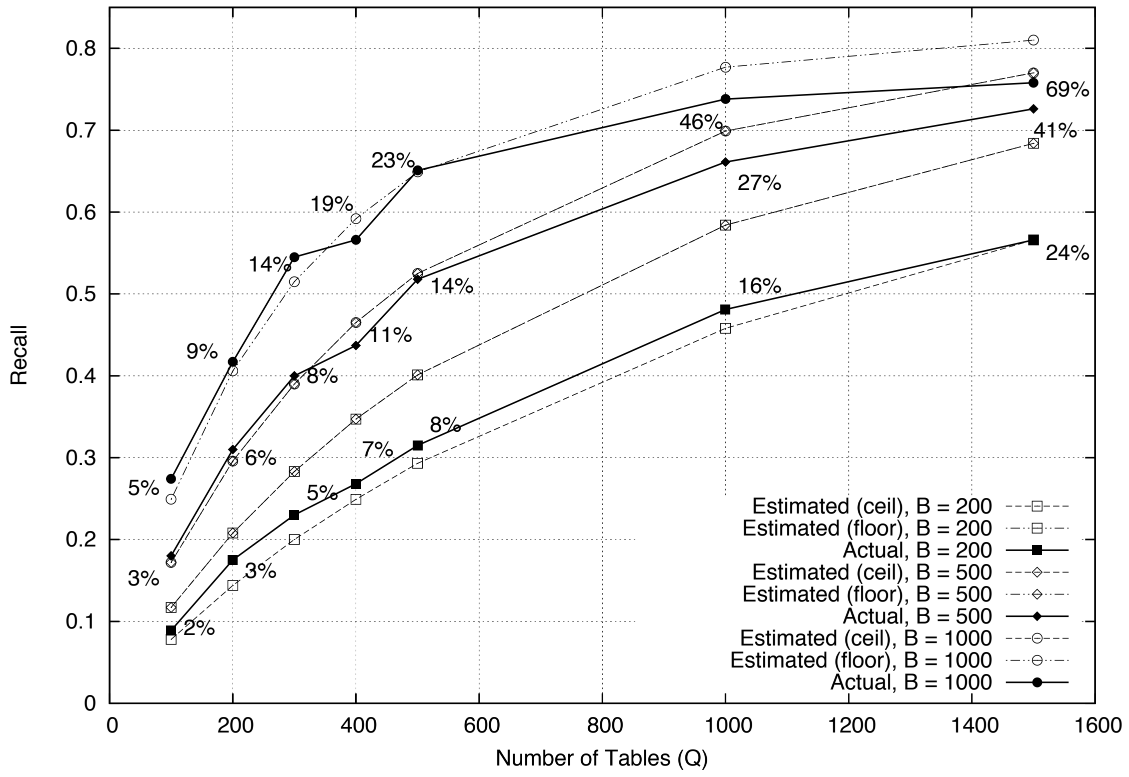


Figure 3.10: Effectiveness-efficiency tradeoffs of the sliding window LSH algorithm for $\tau = 0.3$, using 1000-bit signatures.

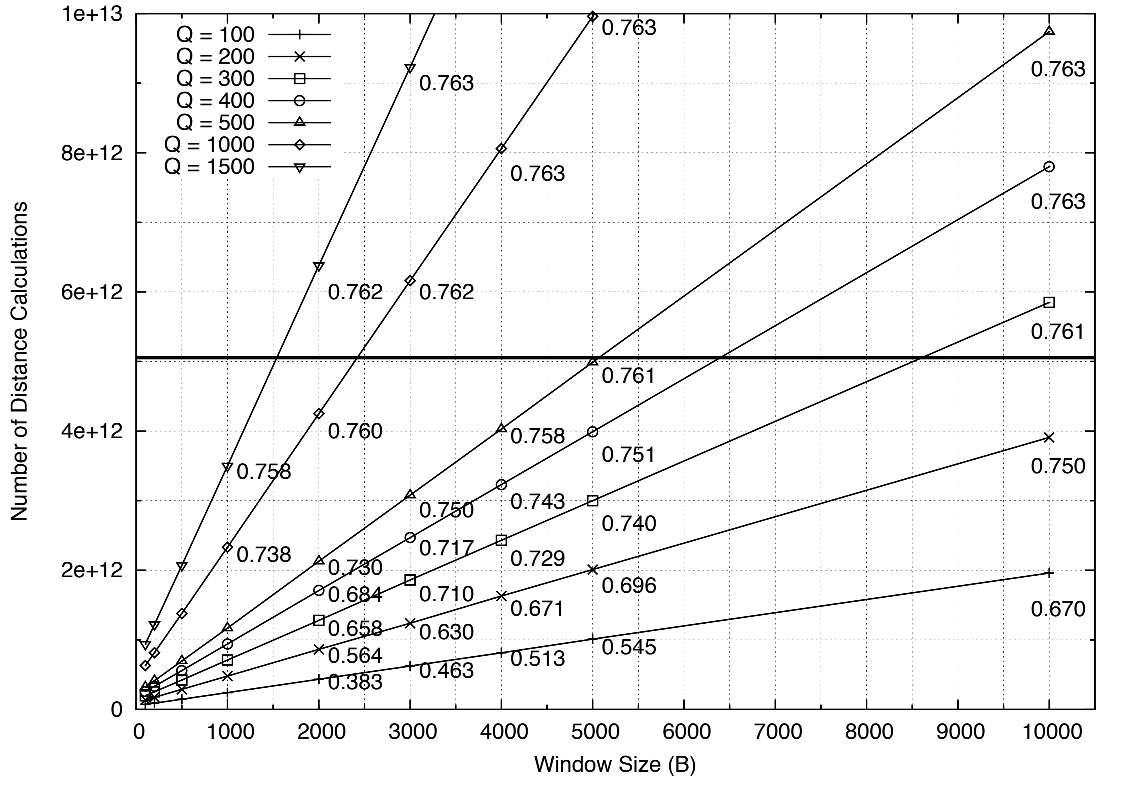


Figure 3.11: Number of distance calculations and corresponding recall values.

cluster with faster processors might benefit from larger values of B, since processor speed determines how fast individual signature chunks can be processed. Of course, we should emphasize once again that there is no "one size fits all" solution, since the most optimal operating setting in the tradeoff space depends on the characteristics of the application and the amount of computational resources at hand.

Another implication of Figure 3.11 is that the brute-force approach cannot be dismissed very easily. In fact, we see that certain levels of recall (beyond 0.761) can only be achieved by doing more work than the brute-force algorithm. This phenomenon is better viewed in Figure 3.12, in which we plot relative effectiveness as a function of relative cost. Each line in the figure corresponds to a different setting of Q, and each point corresponds to a single run of the algorithm. Reaching 100% on the y-axis means that the effectiveness has reached the upper bound, whereas reaching 100% on the x-axis means that the cost has reached the brute-force approach (i.e., we are comparing each German article to every English article). Therefore, moving beyond the thick vertical line in terms of relative cost is not meaningful: it is more efficient to simply use the brute-force algorithm. From the figure, we observe that a maximum of 99.5% relative effectiveness can be meaningfully achieved with our algorithm — for an effectiveness above that, we are better off with a straightforward brute-force approach.

On the other hand, the sliding window algorithm is designed to save work, at the cost of slightly lower recall. For instance, looking at the starting point of the $Q = 300$ line in Figure 3.12, we see that only 35% of the cost is required to achieve close to 93% relative effectiveness. Similarly, the algorithm obtains 95%

and 98% relative effectiveness with a cost about 60% and 50% less than the brute force approach, respectively. From this limited number of experiments, we might conclude that $Q = 1000$ is a good setting in most cases. However, this analysis emphasizes once again that there is no free lunch, and efficiency can only be gained at the cost of effectiveness.

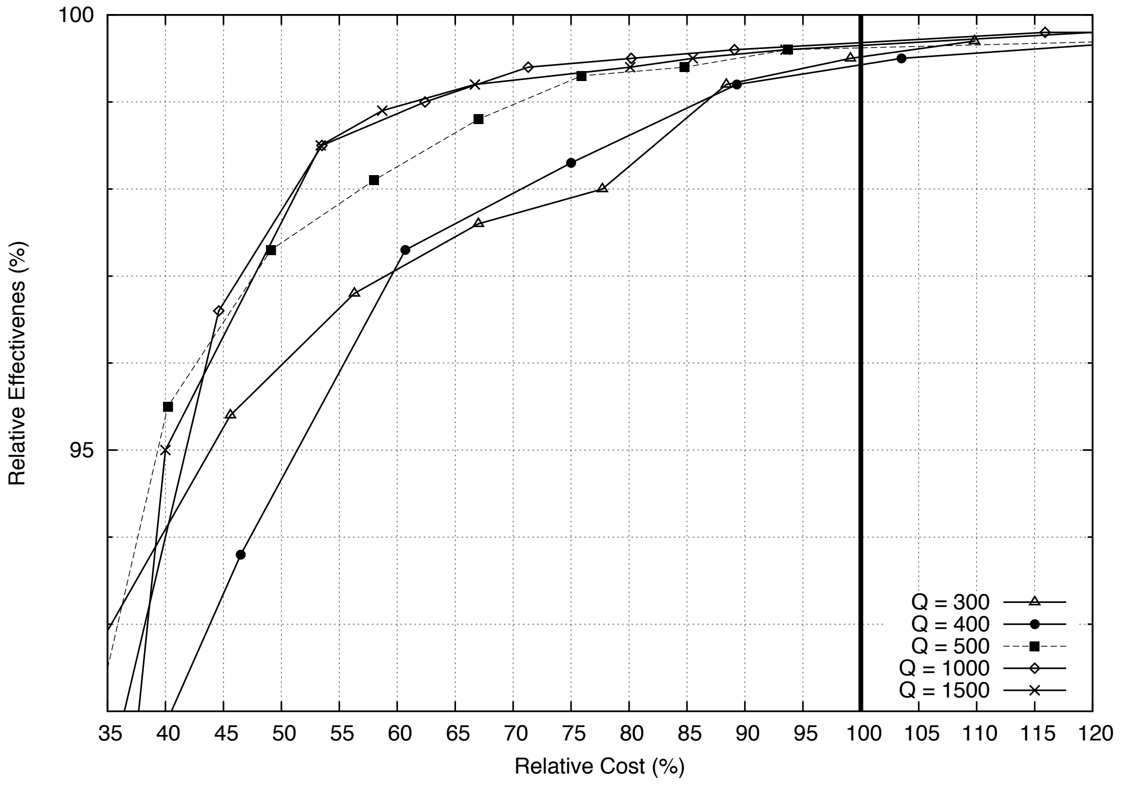


Figure 3.12: Relative effectiveness vs. relative cost of the sliding window algorithm

3.2.5.3 Error Analysis

Finally, we analyze the benefits and drawbacks of our end-to-end system, on the task of matching relevant cross-lingual documents in Wikipedia. This can be done quantitatively by comparing the algorithm output to "interwiki" language links. These links are created manually by users and editors, and are supposed to connect pages about the same entity across languages. However, there has been some work showing that these links are inaccurate [32]. Therefore, we find these links insufficient for use as ground truth, and present the following results with this strong caveat.

To establish a reference on output quality, for each sample German article, we extracted the English article with the highest cosine similarity (using document vectors). The actual "interwiki" links are available from the source of each Wikipedia page, but only 401 of the sample articles had one listed in our downloaded version of Wikipedia. Among those links, our algorithm identified 130 of them (33%) as the most similar article. On the other hand, there were no interwiki links for 61% of the article pairs found by our algorithm. Many of these may be relevant links that were missed by Wikipedia users and editors.

A deeper, qualitative analysis shows that our proposed links are generally helpful, and in many cases they are more comprehensive than the existing Wikipedia links. Table 3.3 shows a few examples of article pairs that were assigned a cosine score above 0.3. For the German article titled "Metadaten," the algorithm correctly puts "Metadata" at the top of the list and includes two other related articles: "Semantic Web" and "File format". Although "Pierre Curie," "Marie Curie" and "Hélène Langevin-Joliot" are all physicists from the same family, and therefore similar, the algorithm places Marie above Pierre in terms of similarity for the German version. This may be due to the richer content of the article about Marie Curie. For the German article "Kirgisistan," 10 articles about recent events in Kyrgyzstan, its leaders and neighbors are in the output (we show the top five matching articles in

Table 3.3). In short, linking cross-lingual articles via similarity has the potential to discover related articles, not just articles on the exact same entity. From the content of these document pairs, we can categorize them as semi-comparable pairs, based on our definition in Section 2.6.

It is interesting that our system was able to return a similar article for only 493 of the 1064 sample German articles. For the remainder, it may simply be the case that a similar article does not exist. This is true especially for articles about specific locations in Germany (e.g., "Dortmund-Aplerbeck"). After analyzing some of these cases by hand, we discovered that there are two common situations where the algorithm struggles. The first is when there is a large gap between the document lengths and contents of the articles in different languages. For instance, the German version of the article about the music show on German TV channel ZDF ("ZDF-Hitparade") is far more comprehensive than the English version. Since cosine similarity measures how similar documents are, it falls below 0.3 in cases like this. Secondly, our system has difficulty matching highly technical articles (e.g., "Farbeindringprüfung," English "Dye penetrant testing"), due to out-of-domain vocabulary: the vector projection has difficulty matching technical terms across languages. This could be solved with more data or special processing of named entities.

3.3 Phase 2: Extracting Bilingual Sentence Pairs

Once similar document pairs are identified within a comparable corpus, the next step is to process text at the sentence level and extract sentence pairs that are

German article	**Top similar articles in English**
Metadaten	1. Metadata
	2. Semantic Web
	3. File format
Pierre Curie	1. Marie Curie
	2. Pierre Curie
	3. Hélène Langevin-Joliot
Kirgisistan	1. Kyrgyzstan
	2. Tulip Revolution
	3. 2010 Kyrgyzstani uprising
	4. 2010 South Kyrgyzstan riots
	5. Uzbekistan

Table 3.3: Examples of relevant German-English article pairs found by our algorithm.

possible translations of each other. The purpose of the first phase is to efficiently reduce the search space as much as possible, while keeping a high level of recall, which is achieved by the sliding window algorithm described earlier. At the sentence level, there is less context and more noise, therefore we need to extract relevant pairs more carefully. Since the output will directly be used for training a statistical MT system, it is essential to minimize the number of non-parallel sentences for an improvement in performance.

In this section, we first describe our classification approach (Section 3.3.1), then provide a detailed description of our algorithm that extracts a parallel corpus from the cross-lingual document pairs (Section 3.3.2), followed by an intrinsic evaluation of our classification approach (Section 3.3.3). Finally, details of the evaluation of our end-to-end pipeline on machine translation can be found in Section 3.4.

3.3.1 Classifier Model

Classification of bilingual sentence pairs is typically performed by a supervised approach, where a classifier is created from a small number of sentence pairs, each labeled as "parallel" or "not parallel" [101, 129, 141]. Previous work has shown that various features help in this classification problem. We have experimented with the following features to determine if a bilingual sentence pair is parallel or not:

- **Cosine similarity** of the two sentences. Each sentence is treated as a document, and is passed through the IR pipeline for preprocessing and tf-idf computation. After translating the tf and idf values of the source-language sentence into the target language (Equations 3.2–3.3), cosine similarity is computed by an inner product between BM25-weighted vectors (Equation 3.1).

- **Sentence length ratio**: Ratio of lengths of the two sentences, where sentence length is defined as the number of tokens without any preprocessing (i.e., number of space-delimited character sequences). Sentence length has shown to be a useful indicator of parallel sentences from the very beginning of relevant research literature: Gale and Church's alignment algorithm is based on sentence length ratios [48].

- **Word translation ratio**: Ratio of words in source (target) sentence that have translations in the target (source) sentence. We use the bilingual translation probabilities learned from the initial parallel corpus in order to determine the translations of each source/target word. When computing the feature value, a source word s is considered the translation of the target word t if $P(s|t) > p$, for some threshold p. Based on an empirical analysis, we fixed $p = 0.1$. To complement the cosine similarity feature, this ratio indicates content match in two directions, and treating these two signals separately might be useful in certain cases.

- **Word alignment ratio**: Ratio of words in source (target) sentence that are aligned to some word in the target (source) sentence. As an improvement to word translation ratio, finding alignments reduce noise by only considering aligned pairs as translations. We run IBM Model 1 to word-align sentence pairs and compute the desired ratio. Higher IBM models also allow other related features based on fertility and distortion that have found to be useful in previous papers. Given trained Model 1 parameters, we can use Equation 2.6 to find the most probable alignment for a sentence pair.

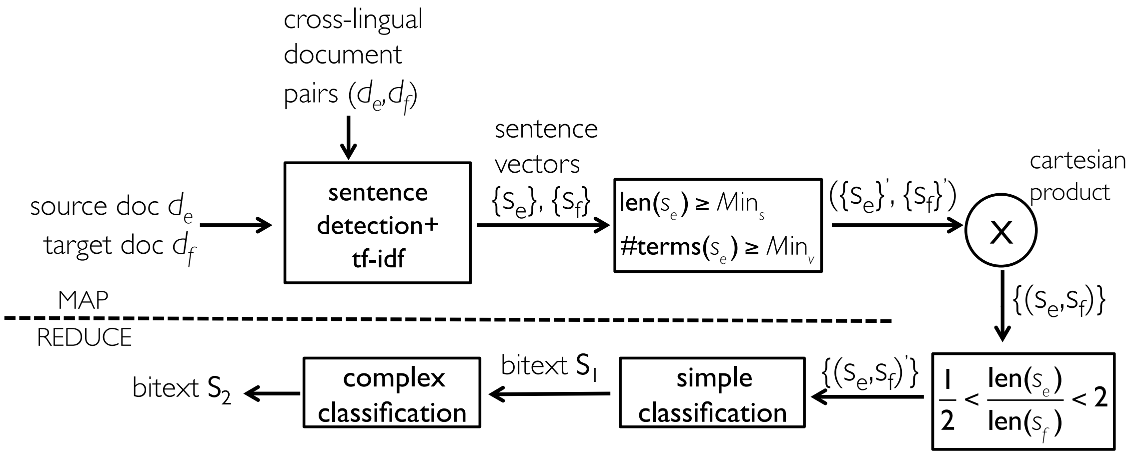
- **Upper-cased entity ratio**: Ratio of upper-cased entities in source (target) sentence that also appear in the target (source) sentence. We call a consecutive sequence of tokens $t_i, ..., t_j$ an *upper-cased entity* if all tokens start with an upper-cased letter, and the tokens to the left and right (i.e., t_{i-1} and t_{j+1}) start with a lower-cased letter. This feature has the potential to increase recall by identifying incorrectly missed pairs (i.e., false negative cases) when there is an upper-cased entity match. There are two difficulties associated with this feature. Firstly, since different languages might have names written slightly differently, we should not require an exact match, and it is challenging to find a good way to represent partial matched entities. The second issue is that this feature does not directly apply to languages that do not use the Latin alphabet (e.g., Arabic, Chinese). Some form of transliteration is needed in these cases.

- **Number ratio**: Ratio of multiple-digit numbers in source (target) sentence that also appear in the target (source) sentence. It is possible to limit the feature to only 4-digit numbers, if the focus is on years. This feature has the potential to prevent some false positive cases by signaling a mismatch between years. This is especially useful in Wikipedia text, where the exact same sentence form is used to describe events that occur annually, such as the NBA (National Basketball Association) finals. Sentence pairs with matching years should be preferred over other ones.

The bottleneck of the bitext extraction algorithm is the time required to classify a given candidate pair of sentences, because that is the core of the computational work. Since adding more features increases computational complexity of the classifier, we propose a novel two-step classification scheme to increase overall efficiency. In this approach, we use two classifiers: a *simple* and fast classifier that uses fewer features, and a *complex* and more effective classifier, which uses more features. In the first step, the simple classifier is applied to all candidate sentence pairs, which efficiently removes many of the low-scored pairs, filtering out a larger portion of the candidate set. In the second step, the complex classifier is applied only to the remaining sentence pairs, and provides a more accurate classification of parallel text.

More details on our actual implementation are given in Section 3.3.3.

3.3.2 Bitext Extraction Algorithm



cross-lingual document pairs (d_e,d_f)
source doc d_e
target doc d_f
sentence detection+ tf-idf
sentence vectors $\{s_e\}, \{s_f\}$
$\text{len}(s_e) \geq Min_s$
$\#\text{terms}(s_e) \geq Min_v$
$(\{s_e\}', \{s_f\}')$
cartesian product
X
$\{(s_e,s_f)\}$
MAP
REDUCE
$\frac{1}{2} < \frac{\text{len}(s_e)}{\text{len}(s_f)} < 2$
$\{(s_e,s_f)'\}$
simple classification
bitext S_1
complex classification
bitext S_2

Figure 3.13: An illustration of the second phase of the bitext extraction pipeline (Chapter 3). This phase takes the entire collection as input, loads the cross-lingual pairs from Phase 1 (Section 3.2), and outputs a parallel corpus.

In Section 3.2, we described phase 1 of our end-to-end pipeline (see Figure 3.1), which produces a number of docid pairs, corresponding to cross-language document pairs found to be similar. For each pair of docids output by the first phase, the next step involves generating the Cartesian product of sentences in both documents as *candidate sentence pairs.* There are many computational challenges of processing so many candidate pairs; besides, even retrieving the document text corresponding to a given docid pair itself is a non-trivial problem, since data resides on a distributed file system. Although it may be possible to load both document collections in memory for smaller collections, we envision scaling up to collections in the future for which this is not possible. As an attempt to address these challenges, we devised a scalable, distributed, out-of-memory solution using MapReduce.

```
Mapper
1: method Initialize
2:     $pwsimMap \leftarrow$ Load(phase1.out)
3:     $sentdetector \leftarrow$ Load(sentence.model)
4: method Map(docid $n$, document $d$)
5:     $similardocids \leftarrow pwsimMap$.Get($n$)
6:     if $similardocids$.IsEmpty() then return
7:     $sentences \leftarrow sentdetector$.DetectSentences($d, Min_s$)
8:     $vectors \leftarrow sentences$.ComputeBM25($Min_v$)
9:     if $vectors$.IsEmpty() then return
10:    for all docid $n' \in similardocids$ do
11:        if $d$.IsEnglish() then    ▷ a language id exists in our document format
12:            Emit($\langle n', n\rangle, \langle E, sentences, vectors\rangle$)
13:        else
14:            Emit($\langle n, n'\rangle, \langle F, sentences, vectors\rangle$)
Reducer
15: method Initialize
16:     $classifier \leftarrow$ Load(maxent.model)
17: method Reduce($\langle n_F, n_E\rangle, [\langle F, sentences_F, vectors_F\rangle, \langle E, sentences_E, vectors_E\rangle]$)
18:     for $i = 1$ to $sentences_F$.Size() do
19:         $sentence_F = sentences_F$.Get($i$)
20:         for $j = 1$ to $sentences_E$.Size() do
21:             $sentence_E = sentences_E$.Get($i$)
22:             $sentlenratio = sentence_E$.Length()/$sentence_F$.Length()
23:             if $sentlenratio > 2 \;||\; sentlenratio < 1/2$ then return
24:             $instance \leftarrow$ ComputeFeatures($v_E, v_F, sentlenratio$)
25:             $label \leftarrow classifier_{\text{simple}}$.Classify($instance$)
26:             if $label$ = 'parallel' then
27:                 Emit($sentence_F, sentence_E$)
```

Figure 3.14: Pseudo-code for the first stage of the bitext extraction algorithm: mappers implement the *candidate generation* step and reducers implement the *simple classification* step.

The algorithm consists of two MapReduce programs, corresponding to the classification scheme we described earlier. The first stage is implemented as a single map-and-reduce sequence, where the mappers generate candidate sentence pairs that may contain partial translations of each other (*candidate generation* step), and the reducers classify the most promising pairs based on the simple classifier's confidence level (*simple classification* step). The second stage simply applies the complex classifier to the output pairs of the first stage (*complex classification* step). An illustration of the two-stage bitext extraction pipeline is presented in Figure 3.13. The pseudocode for the first and second stages of the bitext extraction algorithm are shown in Figures 3.14 and 3.15.

Input to the mappers consists of (docid n, document d) pairs from both collections. In each mapper, all docid pairs $\langle n_F, n_E \rangle$ (corresponding to similar cross-lingual document pair $\langle d_F, d_E \rangle$) are loaded into a hash map (line 2). If the input docid n is not found in any of these pairs, no work is performed (lines 4–6). Otherwise, we extract all sentences of d and retain only those that have at least Min_s terms (line 7). Sentences are converted into BM25-weighted vectors in the English vocabulary space; for source-language sentences, translation into English is accomplished exactly as in phase 1, using the CLIR technique proposed by Darwish and Oard [31]. Any sentence with a weighted vector that contains fewer than Min_v terms is discarded at this point (line 8). For every $\langle n_F, n_E \rangle$ pair that the input docid n is found in, the mapper emits the list of weighted sentence vectors and original sentences, with $\langle n_F, n_E \rangle$ as the key. Since the input contains documents from both languages, we also emit a language identifier (lines 8–14).

As all intermediate key-value pairs in MapReduce are grouped by their keys for reduce-side processing, the reducer receives the key (n_F, n_E), associated with two tuples, corresponding to the sentences and BM25-weighted sentence vectors of d_F and d_E. From there, we generate the Cartesian product of sentences in both languages. As an initial filtering step, we discard all pairs where the ratio of sentence lengths is more than two, a heuristic proposed in [101]. Each of the remaining candidate sentences are then processed by the simple and fast classifier.

This algorithm is a variant of what is commonly known as a reduce-side join in MapReduce, where (n_F, n_E) serves as the join key. Note that in this algorithm, the same sentences and vectors are emitted multiple times, one for each document pair (i.e., (n_F, n_E)) that they appear in: this results in increased network traffic during the sort-and-shuffle phase. We experimented with an alternative algorithm that processes all source documents similar to the same target document together, e.g., processing $(n_F, [n_{E_1}, n_{E_2}, \ldots])$ together. In this case, all sentences in d_e are passed through the network only once, therefore reducing the overall network traffic. However, counter-intuitively, this turned out to be much slower. The explanation is skew in the distribution of similar document pairs. In our experiments, half of the source collection was not linked to any target document, whereas 4% had more than 100 links each. This results in reduce-side load imbalance, and while most of the reducers finish quickly, a few reducers end up performing substantially more computation, and these "stragglers" increase end-to-end running time.

```
Mapper
1: method Map($sentence_F, sentence_E$)
2:     $v_F \leftarrow sentence_F$.ComputeBM25($Min_v$)
3:     $v_E \leftarrow sentence_E$.ComputeBM25($Min_v$)
4:     $sentlenratio = sentence_E$.Length()/$sentence_F$.Length()
5:     $instance \leftarrow$ ComputeFeatures($v_E, v_F, sentlenratio$)
6:     $label \leftarrow classifier_{\text{complex}}$.Classify($instance$)
7:     if $label$ = 'parallel' then
8:         Emit($sentence_F, sentence_E$)
```

Figure 3.15: Pseudo-code for the second stage of the bitext extraction algorithm, in which the complex classifier is applied to all sentence pairs output by the first stage.

The output of the described MapReduce program is piped into the second stage, where the complex and slower, yet more accurate classifier is applied (Figure 3.15). Out of the hundreds of millions of sentence pairs labeled as ‘parallel’ by the simpler classifier, only millions are selected to be fine-tuned in a second round of classification. By applying the slower classifier to a smaller number of instances, we are able to save a significant amount of time. This filtering stage trivially parallelizes the work among mappers and only emits sentence pairs that score above the confidence threshold.

3.3.3 Experimental Evaluation

In order to illustrate the robustness of our approach, we explored parallel text classification for six language pairs: German-English, Spanish-English, Chinese-English, Arabic-English, Czech-English, and Turkish-English. Each language requires separate preprocessing, for which we have adapted off-the-shelf preprocessing and tokenization toolkits.[9] English and German text was tokenized using OpenNLP toolkit,[10] whereas Turkish, Czech, Spanish, and Arabic were preprocessed using Lucene’s tokenization modules.[11] Chinese was segmented using the Stanford segmenter [138]. For English, German, Spanish, and Turkish, we also lowercased text, stemmed using the Snowball stemmer, and removed stop-words. We lowercased text and performed light stemming using code from University of Neuchatel[12] for Czech and applied Lucene’s light stemmer to Arabic text.

[9]All code was implemented as part of the Ivory project: ivory.cc
[10]http://opennlp.sourceforge.net
[11]http://lucene.apache.org
[12]http://members.unine.ch/jacques.savoy/clef/CzechStemmerLight.txt

Language Pair	**Data source**	**Size** (sentence pairs)
de-en	WMT-12	2,079,049
es-en	WMT-12	2,123,036
zh-en	BOLT	4,962,955
ar-en	GALE	3,368,632
cs-en	WMT-12	782,756
tr-en	Yeniterzi and Oflazer [150]	53,113

Table 3.4: Details of the baseline parallel corpus available for each language pair included in our experimental evaluation.

Table 3.4 summarizes the initial parallel corpus available for each language pair. For European languages (German-English, Spanish-English, and Czech-English), we used the training resources provided as part of the 2012 NAACL Workshop on Statistical Machine Translation (WMT-12), consisting of the Europarl corpus (version 7) and News Commentary corpus.[13] For Chinese-English, we used parallel text gathered for the DARPA BOLT evaluation. For Arabic-English, we used the dataset from the DARPA GALE evaluation [108], which consists of NIST and LDC releases. Finally, we obtained the Turkish-English bitext from Yeniterzi and Oflazer [150], which is the best MT corpus we are aware of for this language pair.

These corpora were used for training word translation probabilities and constructing vocabularies, which are required for translating document vectors (Equations 3.2 and 3.3) in phase 1, and for computing classifier features in phase 2 of our pipeline. In this section, we first demonstrate the accuracy of our bitext classification approach, followed by an extrinsic evaluation, in which we compare different

[13]`http://www.statmt.org/wmt12/translation-task.html`

classification approaches on the task of MT.

Intrinsic Classifier Evaluation For each language pair, we trained two classifiers based on the *maximum entropy principle* (using the OpenNLP MaxEnt package[14]), which has become a popular approach for natural language processing applications due to its empirical success, and its ability to represent class predictions probabilistically [11]. We used the baseline parallel corpus of each language pair (Table 3.4) as training data: K parallel (i.e., aligned) sentences were sampled from the parallel corpus as positive training instances. For negative examples, any non-aligned sentence pair from the bitext can be considered. However, we did not use all of the $K(K-1)$ negative instances from the sample, in order to prevent the classifier being affected by the imbalance between positive and negative instances. Following Munteanu and Marcu's suggestion [101], we randomly sampled $5K$ non-parallel sentence pairs from the training data, resulting a fixed 1-to-5 positive-to-negative (i.e., parallel-to-non-parallel) instance ratio.

For feature selection, we performed a preliminary analysis on a portion of the German-English Europarl corpus. This analysis revealed that about 40% of a sample of 1000 parallel sentence pairs have a cosine similarity of 0.3 or more, whereas this value is below 0.2 for almost all of the 1 million non-parallel sentence pairs. This indicates that cosine similarity is a good discriminator of positive and negative instances in the training data. Based on this reasoning, we decided to use cosine similarity as the single feature for the simple classifier. We do not pre-

[14]http://maxent.sourceforge.net/about.html

determine a cosine similarity threshold; instead, we try different thresholds on the test set to determine a sweet spot between precision and recall.

Although cosine similarity can discriminate between many positive and negative instances, being able to correctly identify the borderline cases is essential to high effectiveness. Therefore, our complex classifier contains additional features listed above, namely sentence length ratio and word translation ratio in both directions. We manually selected this feature set based on accuracy: including word alignment ratio, upper-cased entity ratio, and number ratio did not show any improvements on a held-out set of instances.

For the test set, we sampled a set of 1000 parallel pairs from the same initial parallel corpus, for each language pair.[15] We generated all possible 999,000 non-parallel pairs by a Cartesian product from these samples. The unbalanced nature of the test set provides a better estimate of the task we are interested in, since most of the candidate sentence pairs will be non-parallel in a comparable corpus.

Experimental results for all six language pairs are shown in Table 3.5. We report precision, recall, and the F_1 (or F-score) under all test conditions, using different classifier confidence scores as the decision threshold. Recall is shown at two precision levels: high precision (i.e., 95%) and relatively lower precision (i.e., 80%). For the size parameter K, we tried different values empirically, and concluded that increasing beyond $K = 1000$ does not improve performance on a held-out development set. Therefore, all results were obtained using training data that consists of 1000 parallel sentence pairs and 5000 randomly sampled non-parallel sentence pairs.

[15]The set of sentence pairs used for training and testing are disjoint.

In Table 3.5, we observe that language pairs that perform better in the classification task (German-English, Spanish-English, Arabic-English, and Czech-English) have more training resources, with the only exception being Czech-English. We explain better performance on Czech-English by linguistic similarity: more similar languages should require fewer examples to learn how to accurately classify unseen instances. The two language pairs with lower accuracy are Turkish-English and Chinese-English, with F-scores in low 80s, as opposed to high 80s and low 90s. Recall is especially low at very high precision (R@P95) for these two cases, which is expected due to the low amount of seed training data and major linguistic differences. The difference between simple and complex classification is also most apparent at high precision, which is consistent with the motivation that the complex classifier is for high-precision decisions, as opposed to the recall-oriented simple classifier. Our German-English and Spanish-English precision-recall values are similar to the results of Smith et al.'s CRF approach [129], but we are not aware of any work with a comparable evaluation for the other language pairs.

Due to the large amounts of data involved in our experiments, we were interested in speed-accuracy tradeoffs between the two classifiers. Micro-benchmarks were performed on a commodity laptop running Mac OS X on a 2.26GHz Intel Core Duo CPU, measuring per-instance classification time, which we define as the time excluding preprocessing of sentences into BM25-weighted vectors, and including feature computation and classification. Based on our experiments on 1 million instances, the complex classifier took 105 μs per instance (96 μs for feature computation, 9 μs for classification), about 4 times slower than the simple one, which

Language	**Simple**			**Complex**		
Pair	R@P95	R@P80	F_1	R@P95	R@P80	F_1
de-en	59%	95%	88%	77%	97%	91%
es-en	80%	93%	89%	91%	97%	93%
zh-en	37%	82%	81%	40%	87%	84%
ar-en	73%	94%	88%	87%	97%	92%
cs-en	87%	93%	91%	94%	87%	92%
tr-en	45%	80%	81%	58%	79%	81%
Speed (in μs, per instance)	27			96		

Table 3.5: Comparison of the two bitext classifiers, tested on sentence pairs held out from the training data. R@P95 and R@P80 refer to recall percentage values, reported at 95% and 80% precision, respectively.

took 27 μs (18 μs for feature computation, 9 μs for classification).

In order to assess the influence of genre/domain on accuracy, we performed additional experiments for German-English only. For this evaluation, we used the set of labeled German-English Wikipedia sentence pairs created by Smith et al. [129]. This is especially useful since we are also interested in extracting bilingual text from Wikipedia. In the test set, human annotators identified 312 parallel sentence pairs in 20 Wikipedia article pairs. Similarly, negative instances were artificially generated by considering all non-aligned sentence pairs in the same document, resulting in a test set of 97,032 sentence pairs. The authors also published a set of 225 sentence pairs for training, collected from Wikipedia titles and Wiktionary[16] entries.

We tested all four possible conditions: training the classifier using in-domain and out-of-domain data, and testing it on in-domain and out-of-domain for each

[16]Wiktionary is a dictionary tool similar to Wikipedia: `www.wiktionary.org`

Training	**Testing**	**Simple**			**Complex**		
domain	**domain**	R@P95	R@P80	F_1	R@P95	R@P80	F_1
Europarl	Europarl	59%	95%	88%	77%	97%	91%
Wikipedia	Europarl	54%	97%	90%	59%	95%	89%
Europarl	Wikipedia	7%	74%	79%	12%	81%	81%
Wikipedia	Wikipedia	2%	57%	75%	3%	61%	75%

Table 3.6: Comparison of the German-English bitext classifiers trained and tested on in-domain (Europarl) or out-of-domain (Wikipedia) sentence pairs. R@P95 and R@P80 refer to recall percentage values, reported at 95% and 80% precision, respectively.

case. All results are reported in Table 3.6, from which we make several observations. First of all, results are relatively lower for the Wikipedia test set (10-15% absolute difference in F-score values and very low recall values at 95% precision), which can be attributed to more informal language and noisy alignments. Another factor is out-of-vocabulary cases, which is more substantial for the Wikipedia test set because the bilingual translation probabilities are learned from Europarl. Another interesting observation is that training the complex classifier on Wikipedia data does not bring as much improvements (over the simple one) as it does when trained on Europarl. One explanation for this behavior might be over-fitting: since the Wikipedia training set ($K = 225$) is much smaller than Europarl ($K = 1000$), it is more vulnerable to over-fitting, causing lower test set accuracy. Finally, we see that training on Europarl yields better performance (especially at high precision) on both test cases, despite the domain difference. This points to a quality difference between Europarl and Wikipedia training examples.

Extrinsic Classifier Evaluation In Sections 3.2 and 3.3, we described the two phases of our end-to-end parallel text extraction pipeline, as illustrated in Figure 3.1. Before describing our extrinsic evaluation, here is a summary of how we ran the end-to-end MT pipeline, referring back to the sections in which each part was presented. For each of the six language pairs, we started from the two raw Wikipedia collections.[17] Wikipedia articles were first preprocessed into document vectors, using Okapi BM25 term weights (Section 3.2.3.1). Source-language[18] Wikipedia articles were then projected into the English vocabulary space using CLIR techniques (Section 3.2.2). All vectors were converted into LSH signatures using the *random projections* method (Section 3.2.3.2). The sliding window algorithm accepts signatures of both collections as input, and returns pairs of document ids: each corresponds to a pair of Wikipedia articles found to be similar, according to the cosine similarity estimate of their signatures (Section 3.2.3.3).

The parallel extraction algorithm is run on the raw Wikipedia articles, but needs to load the output of phase 1 into memory before processing any article. As shown in Figure 3.13, each input article is first split into sentences, achieved by applying a trained sentence detector model,[19] using the OpenNLP toolkit.[20] Any sentence containing less than five tokens or three unique terms is discarded at this point (i.e., $Min_s = 5$, $Min_v = 3$ in Figure 3.14). For each similar article pair, every possible pair of the remaining sentences are considered for classification, except for pairs in which the sentence length ratio is above 2 or below 0.5. Finally, the

[17]Wikipedia articles in XML format are freely available from `http://wikipedia.c3sl.ufpr.br`
[18]The target language is always English in our experiments.
[19]The baseline parallel corpus was used to train the sentence detector models.
[20]http://sourceforge.net/apps/mediawiki/opennlp/index.php?title=Sentence_Detector

remaining sentence pairs are processed through the two-step classification to obtain a parallel corpus.

As the first step of our classification approach, the simple classifier is applied to the candidate pairs, each time with a different confidence threshold. We adjusted the threshold to obtain different amounts of bilingual text, and observed its effect on translation quality (this experimental condition is called S_1 hereafter). Optionally, the complex classifier is then applied to the output of the simple classifier for additional filtering (this two-step approach is called S_2 hereafter).

We compared the two experimental conditions (one-step classification, S_1, and two-step classification, S_2), by evaluating the output on German-English MT. For this evaluation only, we used data from a slightly older WMT-10 task,[21] which uses a previous version of the Europarl corpus (version 5, with 3.1 million sentence pairs) for training, a set of 2,525 sentences for tuning and 2,489 sentences for testing.[22] In all of our MT experiments, we used the state-of-the-art hierarchical phrase-based translation system (Hiero) as the translation model, which is based on a synchronous context-free grammar (SCFG) [24]. GIZA++ [106] was used to learn word alignments, and a 5-gram English language model was built using SRILM [133]. We used the C-based decoder `cdec` [36] for decoding, and the system parameters were tuned using MIRA [38]. All systems were evaluated by BLEU [110].

Here is an example run: as a result of applying the simple classifier with a threshold of 0.98,[23] 13.8 million German-English sentence pairs were extracted from

[21]`http://www.statmt.org/wmt10/translation-task.html`

[22]The remaining MT experiments in Section 3.4 use the newer WMT-12 corpus, as listed in Table 3.4.

[23]There is no meaningful interpretation for the actual value of the confidence threshold.

phase 1 output. By running the second classification step, using a threshold of 0.60 for the complex classifier, the size of this bitext was reduced to 5.8 million. This corresponds to condition S_2. Alternatively, we could raise the initial threshold to 0.988 and not apply the second classifier, which would result in a final output of 6.1 million sentence pairs. This corresponds to condition S_1. For reference, Figure 3.13 illustrates the pipeline for running phase 2 experiments.

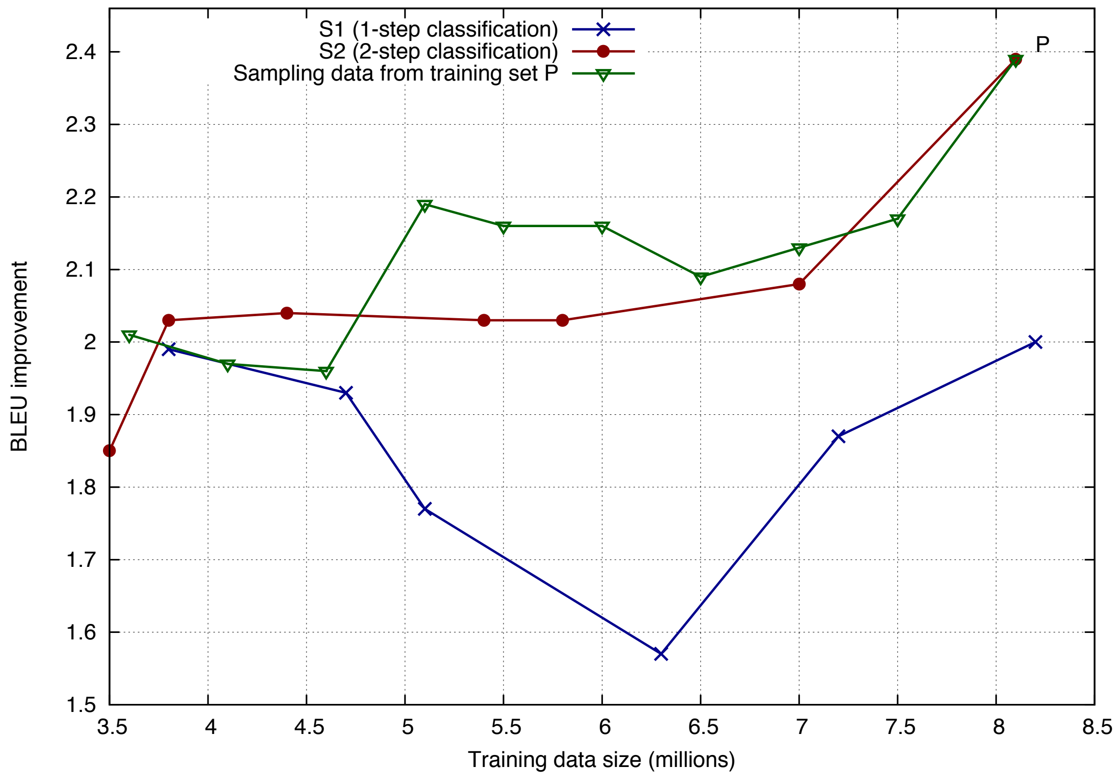


Figure 3.16: A comparison of bitext classification approaches on WMT-10 German-English test set.

Under each condition, we varied the confidence thresholds of both classifiers to generate new bitext collections of varying sizes. In each case, the final sentence pairs were added to the baseline training data, to train an entirely new translation model.[24] Therefore, each data point shown in Figure 3.16 represents the evaluation result of a particular choice of classifier thresholds, where the x-value indicates the total training data size.

Our conclusion is that the S_2 condition increases translation quality by reducing noise: for each data size that we experimented with, the BLEU score exceeded that of S_1. We also observe that BLEU increases with more data, but most of the improvements can be attributed to the small set of initially added sentence pairs. The best performing data (which we denote as P) only brings an additional 1 BLEU improvement to the initially added 500 thousand sentence pairs.

Another interesting observation is the oscillating trend of BLEU improvements under the S_1 condition. The 2-point BLEU improvement from the first 3 million sentence pairs decreases about a half point after the second batch of three million, and then returns back to a full 2-point improvement after adding the final 2 million sentence pairs. While the less stable results might be attributed to a less accurate classification approach (1-step vs. 2-step), this might also be an artifact of noisy parameter optimization. Clark et al. [29] have shown that repeating the parameter tuning procedure few times might allow higher confidence when comparing MT systems.

In order to better examine the effect of data size alone, we created partial datasets from P by randomly sampling an increasing proportion of the entire set, and repeating experiments for each of these samples. Results are also shown in Figure 3.16. As expected, we see a generally increasing trend of BLEU scores with

[24]GIZA++ additionally filters out some of the pairs based on length.

respect to data size. By comparing the three plots, we see that S_2 or random sampling from P work better than S_1. Also, random sampling is not always worse than S_2, since some pairs that receive low classifier confidence turn out to be helpful in terms of increasing BLEU score (e.g., by fixing out-of-vocabulary issues).

3.4 Comprehensive MT Experiments

Intrinsic and extrinsic evaluations of each phase were described in subsections 3.2.5 and 3.3.3. In this section, we present an extensive evaluation of the end-to-end pipeline, assessing the quality of the output (i.e., a parallel corpus) on the task of MT, for a diverse set of language pairs. We first present details on the baseline MT system, and then discuss how we included the parallel text for training an improved MT model.

A baseline MT system was trained for each language pair, on the corresponding parallel corpus in Table 3.4. For tuning parameters and testing the final output, we used the newswire datasets provided for WMT-12 for language pairs German-English, Spanish-English, and Czech-English. Chinese-English and Arabic-English systems were tuned and tested using NIST MT datasets, which contain four references for each sentence. Finally, for the Turkish-English system, we sampled about 1000 sentences from the original parallel corpus as tune and test sets, and left the rest for training purposes. Table 3.7 lists the data used for tuning MT parameters and testing our approach, as well as the BLEU score our baseline MT approach received.

Language	**Tune set**		**Test set**		**BLEU**
Pair	Source	Size	Source	Size	(test)
de-en	WMT (newstest-11)	3003	WMT (newstest-12)	3003	24.50
es-en	WMT (newstest-11)	3003	WMT (newstest-12)	3003	33.44
zh-en	NIST (MT-06)	1664	NIST (MT-08)	1357	27.52
ar-en	NIST (MT-06)	1797	NIST (MT-08)	813	63.15
cs-en	WMT (newstest-11)	3003	WMT (newstest-12)	3003	23.11
tr-en	held out from [150]	1071	held out from [150]	1095	27.22

Table 3.7: Datasets used for tuning parameters and evaluating translation quality, listed separately for each language pair. Last column shows the BLEU score achieved by the baseline MT system on the testing dataset.

For all runs, we included all standard MT features, including phrase and lexical translation probabilities in both directions, word and arity penalties, and language model scores. We compared the BLEU score of our baseline systems to comparable results reported in the literature.

The Chinese-English baseline MT system achieved 27.52 BLEU on the NIST MT-08 test set. Based on results reported directly by NIST,[25] this score would have ranked 6th out of the 20 participants in the official evaluation. For Arabic-English, the baseline BLEU score of 63.15 is much higher than reported results.[26] We are aware of only one paper that uses the same Turkish-English training data, and the authors report BLEU values between 17.08 and 21.96. These numbers are not directly comparable to our 27.22 baseline BLEU since we randomly sampled 1095 sentence pairs for testing [150]. For German-English, Spanish-English, and Czech-

[25]`http://www.itl.nist.gov/iad/mig//tests/mt/2008/doc/mt08_official_results_v0.html`

[26]`http://www.itl.nist.gov/iad/mig//tests/mt/2008/doc/mt08_official_results_v0.html`

English, our baseline results are comparable to reported results from the WMT-12 evaluation. For example, the baseline BLEU score of 24.55 for German-English would have shared first place with two of the sixteen participants, with a range of scores between 11 and 24 [18]. Our Spanish-English baseline system achieved 33.39 BLEU, which would place it second among the twelve participants of WMT-12, where reported scores range from 22 to 38 [18]. Finally, the Czech-English baseline MT system would have ranked first among the six WMT-12 participants, based on its BLEU score of 23.11 [18].

Size (in millions)	**Language** (Wikipedia XML dump date)[a]						
	English (12/1)	German (12/15)	Spanish (11/30)	Chinese (12/10)	Arabic (12/18)	Czech (12/15)	Turkish (12/17)
Pages	10.16	3.00	2.61	2.07	0.53	0.50	0.59
Articles[b]	4.04	1.57	1.19	0.60	0.32	0.30	0.31
Signatures[c]	4.01	1.42	0.99	0.59	0.25	0.26	0.23
Similar pairs	-	35.9	51.5	14.8	5.4	9.1	17.1

[a] all dump dates are from year 2012.
[b] after discarding stub, disambiguation and redirect pages.
[c] after discarding short articles.

Table 3.8: Phase 1 statistics for all Wikipedia collections (in millions). Raw Wikipedia XML dumps were downloaded from `http://dumps.wikimedia.org`.

3.4.1 Results

Tables 3.8 and 3.9 display experimental results from the first and second phases of our approach, for all six language pairs that we evaluated. In these experiments, we set the simple classifier threshold based on its accuracy on held-out examples

Size	**Source Language (Wikipedia)**					
	German	Spanish	Chinese	Arabic	Czech	Turkish
Sentences	42.3m	19.9m	5.5m	2.6m	5.1m	3.5m

Pairs	de-en	es-en	zh-en	ar-en	cs-en	tr-en
Candidate	530b	356b	62b	48b	101b	142b
Processed[a]	322b	213b	36b	23b	62b	85b
Parallel (S_1)	292m	178m	63m	7m	203m	69m
Parallel (S_2)	0.2-3.3m	0.96-3.3m	0.05-0.29m	130-320k	0.5-1.63m	8-250k

[a] after applying filtering heuristics

Table 3.9: Phase 2 statistics for all language pairs, using respective Wikipedia collections specified in Table 3.8. Number of sentences in the target language (English Wikipedia) is omitted in the table, which is slightly different for each language pair (due to vocabulary differences from the baseline training corpus), ranging from 75 to 91 million.

(as presented in Table 3.5): for each language pair, the threshold was set to the value in which 90% recall was achieved (since the goal of simple classification is high recall). For example, for Arabic-English, the classifier achieves 90% recall and 86% precision when the classifier threshold is set to 0.84. The row in Table 3.9 labeled "Parallel (S_1)" indicates the number of sentence pairs classified as "parallel" when the threshold was decided as above.

For condition S_2, the complex classifier was applied to the S_1 output with a range of threshold values, in order to see the relationship between data size and BLEU score. The last row (labeled "Parallel (S_2)") shows the range of sentence pairs extracted with this thresholding strategy. For each threshold setting, we add the parallel text output by S_2 to the baseline corpus (see Table 3.4), and run it through the MT pipeline, to get an updated translation model. The parameters of this translation model are tuned and tested on corresponding datasets in Table 3.7.

For each such MT experiment, the BLEU score (with respect to human translation references) is computed and compared to the baseline BLEU scores in Table 3.7. Figures 3.17 and 3.18 illustrate the improvement in BLEU score when the varying amounts of extracted parallel text are added. For clarity, we included the three language pairs with most amount of resources (i.e., German-English, Spanish-English, Chinese-English, and Arabic-English) in the former figure, and the other languages in the latter one (i.e., Czech-English, Turkish-English).

Several conclusions can be made from these results. First, we notice that not all extracted data bring BLEU improvements: there are data points that show a lower BLEU than the baseline for Czech-English and Turkish-English, although we see improvements in at least one experiment for both language pairs. These two language pairs have the least amount of parallel text available in the baseline corpus (See Table 3.4), indicating a correlation between amount of initially available resources and the improvements gained from bitext extraction. This is expected since the core of the parallel text extraction algorithms is the translation methods, which are highly dependent on the quality of the underlying vocabulary and word translation probabilities. With less data to generate these resources, our approach is less robust to the noise in Wikipedia, resulting in lower quality output. In Chapter 5, we present ideas to overcome this weakness, especially for low-resource language pairs.

On the other hand, we see largest improvements in Arabic-English, suggesting that a combination of high-resource and linguistically different language pairs are a good candidate for improvements from parallel text extraction approaches.

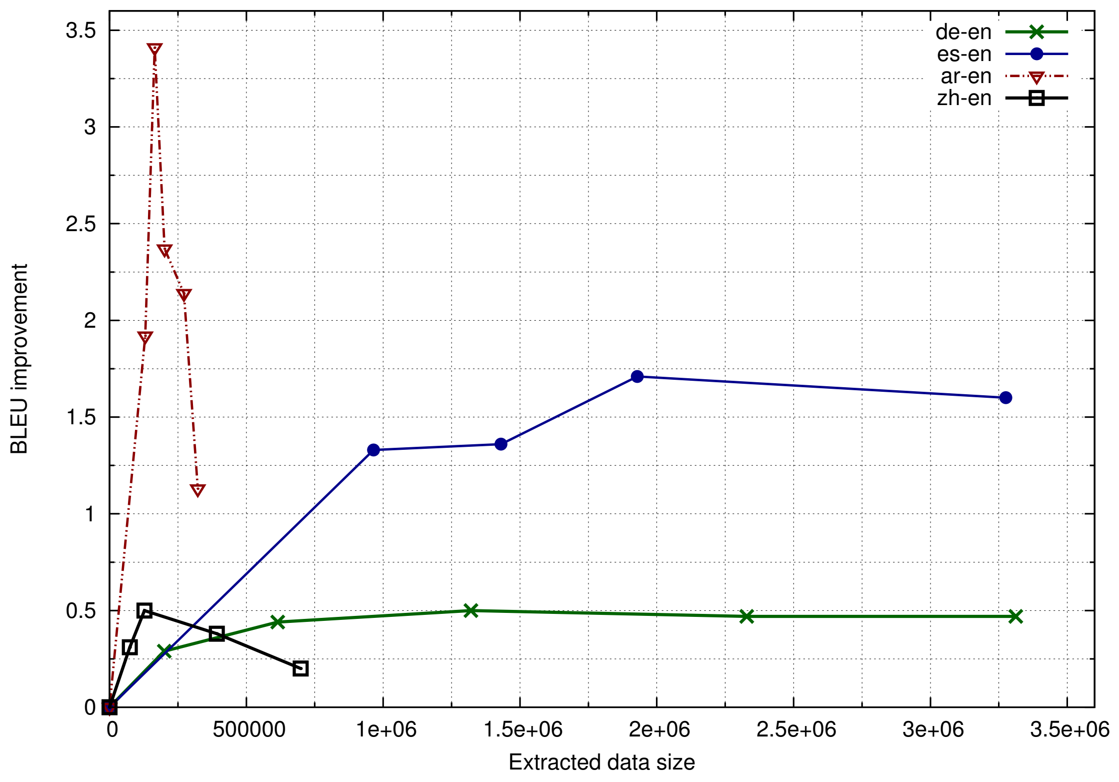


Figure 3.17: Evaluation results on translating high-resource language pairs into English: BLEU scores are reported relative to the baseline MT scores.

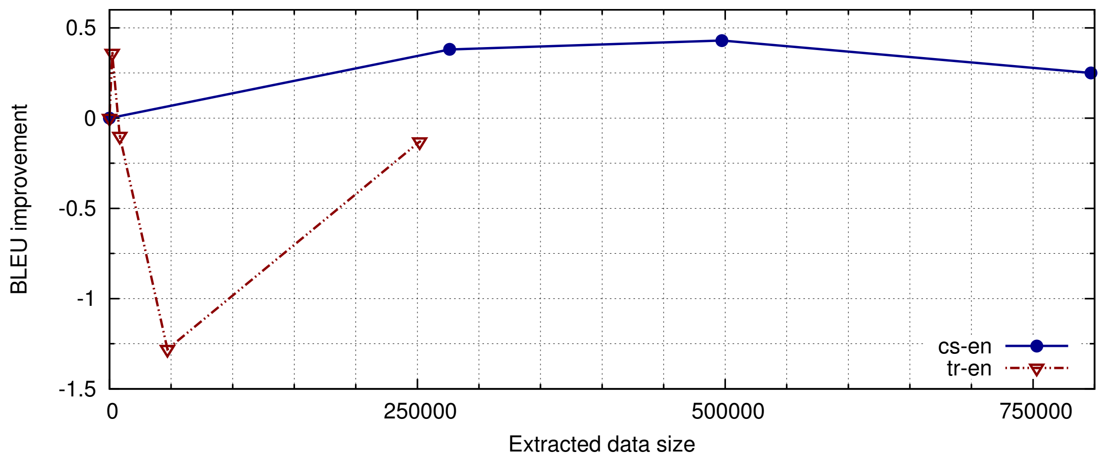


Figure 3.18: Evaluation results on translating low-resource language pairs into English: BLEU scores are reported relative to the baseline MT scores.

For Chinese-English, improvements are more modest, possibly due to the challenges associated with segmentation. Languages such as German and Spanish have a relatively higher baseline performance, making it more difficult to improve by adding more training data.

Another interesting observation is that there is a limit on the number of useful sentence pairs we can extract from Wikipedia. It is not possible to simply conclude that "more data is better" from any of our experimental results, since we see a decrease in BLEU score beyond a certain amount of data (of course, the amount varies heavily across different languages). We can generalize this observation to any comparable corpus: the parallel text our approach can extract is limited to the number of truly parallel sentence pairs within the candidate set. As a result, we need to be careful in applying these algorithms to other collections. A similar analysis should be useful to determine the upper bound of any given comparable corpus.

3.4.2 Efficiency

We show the efficiency of our approach by reporting running times for German-English experiments, which were the two largest Wikipedia collections at the time of this evaluation. Experimental runs were repeated three times to compute a 95% confidence interval for the running times of the various components. Average candidate generation time was 2.4 hours (± 0.42 hours). These candidates went through the MapReduce shuffle-and-sort process in 1.25 hours (± 0.4 hours), which were then classified in another 4.13 hours (± 0.17 hours). Processing time by the more complex classifier in S_2 depends on the threshold of S_1, because it determines the number of instances to be classified in S_2. Even when using the threshold that produced the largest bitext in German-English experiments, complex classification took only 0.52 hours. When we compare these numbers to the other collections, we notice that running time is linear in the number of sentence pairs within the similar document pairs. Since the English collection is the same in all of our experiments, running time depends on the number of sentences in the non-English collection. As an example, Chinese Wikipedia has about an eighth of the number of sentences in German Wikipedia, and the entire bitext extraction pipeline took about 1.54 hours for Chinese-English.

3.4.3 Error Analysis

For an error analysis, we manually selected some of the parallel sentence pairs output by the pipeline. In Table 3.3, we show seven sentence pairs as a representative sample of some of the strengths and weaknesses of our approach.

In the first two examples, the sentence pairs can be rated as excellent translations. These examples come from the French-English and Turkish-English corpus, indicating that such perfect pairs can be extracted for both high-resource and low-resource language pairs. A related observation that applies to the output in general is that long sentences extracted by our approach are much more likely to be good choices.

In items 3–7 of Table 3.3, we see examples of certain errors that exist in other parts of the extracted bitext as well. One such error is to match sentence pairs about different events that are recurring, such as sport events like the Super Bowl (e.g., Example 3 in the table). Since certain parts of the text match perfectly (e.g., team names, name of the tournament), it is not trivial to detect these false positive cases. We have tried several ideas to fix this, including *number ratio* features that punish instances in which a date in one sentence does not appear in the other. Another common issue is with handling names: in Example 4, two different American actresses that share the same first name are matched by our algorithm. This happens due to high similarity in the rest of the text, and can be mitigated by performing named entity recognition, as a measure to make sure named entities get translated. We implemented *upper-cased entity ratio* features in order to address this issue, but did not see improvements in classification accuracy over a held-out development set.

There are also challenges specific to the Wikipedia collections we are using, namely (a) XML parsing errors and (b) discrepancy in the content of Wikipedia articles across languages. Example 5 is a perfect example of the former issue, in which the text is parsed from a table, but we see that the word in the first table column (i.e., Bronze) is not properly parsed in the English article. On the Czech side, we see that "21px" is incorrectly parsed into the text, adding noise to the task.

Sentence pairs 6 and 7 exemplify the discrepancy issue in Wikipedia: in example 6, the time of the tornado is entered as 1912 in the Spanish article and 1913 in the English one. Without further context, one might think that there is a 1-year difference between the two reports; however, these numbers refer to the time (19:12 vs 19:13) in which the tornado happened. This 1-minute time difference is possibly due to citing from different sources. This is yet another piece of evidence that *language is ambiguous and hard*. As a result of this sentence pair, noise is introduced to the MT training process. Another example of discrepancy is in the last example, where the German sentence lists the location as "Novi Sad, Serbia and Montenegro" whereas the city is written as part of "Federal Republic of Yugoslavia" in the English version. Both are true, since those are two valid ways to refer to the country. In this case, the example might actually be useful for MT, by learning a new mapping ("Serbia and Montenegro"$\leftrightarrow$"Federal Republic of Yugoslavia").

Of course, there are also many cases in which the discrepancy is due to human error, since Wikipedia is a crowd-sourced encyclopedia. Also, differences in both content (i.e., one sentence has additional content) and structure (i.e., the same content is split into few sentences in one language) create difficulties for our current approach. Some of these weaknesses are caused by focusing only at the sentence-level.

As we see in Table 3.10 and the rest of the extracted data, many of the extracted pairs are not exact translations, as one would expect from a manually generated corpus. However, MT approaches are usually robust to such cases, where a portion of each sentence is a valid mutual translation. In addition to type I errors (i.e., false positives) discussed above, our approach also exhibits type II errors (i.e., missed pairs), and this is mostly due to vocabulary or translation coverage. If a sentence contains words that are not in our vocabulary, or words that we do not know the proper translation of, it is likely that our classifier will not label it as "parallel".

In an attempt to analyze where the bitext extraction approach is helping MT the most, our insight is that vocabulary expansion and the reduction of out-of-vocabulary (OOV) cases constitute the largest share. In most of the experiments, the reduction in OOV words was over 50% for German-English, which is very substantial. These improvements could have been even more impressive if we had evaluated on test sentences from other domains, such as blogs or chat. Even though this dissertation does not focus on MT experiments in domains other than news, this would be a very promising direction to take in the future.

1	**fr**	“Afrique Centrale” désigne l’espace géographique couvrant lensemble des 11 États membres du Comité consultatif permanent des Nations Unies chargé des questions de sécurité en Afrique centrale la République dAngola, la République du Burundi, la République du Cameroun, la République gabonaise, la République de Guinée équatoriale, la République centrafricaine, la République démocratique du Congo, la République du Congo, la République du Rwanda, la République de Sao Tomé-et-Principe et la République du Tchad
	en	“Central Africa” refers to the geographical area covering the 11 States that are members of the United Nations Standing Advisory Committee on Security Questions in Central Africa, namely, the Republic of Angola, the Republic of Burundi, the Republic of Cameroon, the Central African Republic, the Republic of Chad, the Republic of the Congo, the Democratic Republic of the Congo, the Republic of Equatorial Guinea, the Gabonese Republic, the Republic of Rwanda and the Democratic Republic of Sao Tomé and Principe.
2	**tr**	10 Mart 2010 tarihli Avrupa Parlamentosu kararı, ”sızdırılan belgelere göre,, FMH uygulanması (ilgili AB mevzuatı bekleyen diğer şeylerin yanı sıra ACTA müzakerelerin dokunma,, COD/2005 / 0127 - Ceza önlemleri uygulama (fikri mülkiyet haklarının güvence altına amalayan -II IPRED)) ve sözde ” Telekom Paketi ”ve” e-ticaret ve veri koruma ile ilgili mevcut AB mevzuatı.
	en	The European Parliament resolution of 10 March 2010 stated that ”according to documents leaked, the ACTA negotiations touch on, among other things, pending EU legislation regarding the enforcement of IPRs (COD/2005/0127 Criminal measures aimed at assuring the enforcement of intellectual property rights (IPRED-II)) and the so-called ”Telecoms Package” and on existing EU legislation regarding e-commerce and data protection.”
3	**de**	Am 5. Februar 2012 konnten die New York Giants unter der Fhrung von Coughlin die New England Patriots im Super Bowl XLVI.
	en	The Helmet Catch (February 3, 2008, New York Giants vs. New England Patriots, Super Bowl XLII)
4	**de**	Patricia Marand (1934–2008), US-amerikanische Schauspielerin
	en	Patricia Clarkson, American actress
5	**cs**	Bronz Karol von Rommel Jzef Trenkwalda Micha? Antoniewicz 21px jezdectv Jezdeck v?estrannost - dru?stva
	en	Jzef Trenkwald, Micha? Antoniewicz and Karol Rmmel Equestrian, Team eventing
6	**es**	EF1 E de Stringer Smith 1912 Tres casas tuvieron daos menores y varias casas de pollo fueron fuertemente daadas.
	en	EF1 E of Stringer Smith 1913 Three houses sustained minor damage and several chicken houses were heavily damaged.
7	**de**	Gruppe B: 28. Dezember 2002 bis 3. Januar 2003 in Novi Sad, Serbien und Montenegro
	en	Group B played in Novi Sad, Federal Republic of Yugoslavia between December 28, 2002 and January 3, 2003.

Table 3.10: Example sentence pairs extracted from Wikipedia.

3.5 Conclusions and Future Work

In this chapter, we introduced an end-to-end pipeline that generates a parallel corpus, given a corpus with semi-comparable or fully comparable document pairs and a seed parallel text to train translation models. We came to the conclusion that our approach can be successful at this task even for semi-comparable corpora due to several properties: (1) The approach benefits from the theoretical guarantees of LSH techniques, (2) the implementation allows flexibility in adjusting parameters based on collection characteristics, and (3) the evaluation provides evidence that the algorithm can identify parallel portions within semi-comparable text (e.g., an article about "Kyrgyzstan" is linked to an article about "Tulip Revolution").

Our implementation runs in two phases, corresponding to the problems of cross-lingual pairwise similarity and bitext classification. The first phase requires finding cross-lingual document pairs that have cosine similarity values above some pre-defined threshold. We adapted an LSH-based approach to the problem, as a parallelized MapReduce algorithm. We showed that 1000-bit LSH signatures are not sufficient to achieve perfect recall but this is the cost of a representation that is significantly faster to process. We experimentally and analytically quantified the effectiveness-efficiency tradeoff of the sliding window approach with respect to a multitude of parameters. This characterization provides a guide to the application developer in selecting the best operating point for a specific situation. A somewhat surprising finding is that a brute-force approach should not be readily dismissed as a viable solution, especially when high recall is desired. Our evaluation shows that we can find many useful links between Wikipedia articles in German and English.

The second phase requires generating a very large amount of candidate sentence pairs, followed by a computationally intensive classification task. We introduced a MapReduce algorithm to perform the necessary computations in a scalable and efficient manner, and demonstrated its running time on the two largest Wikipedia collections: German and English. We showed for multiple language pairs, that an impoverished, data-driven approach is potentially more effective than task-specific engineering. With the distributed bitext mining machinery described in this paper, improvements come basically "for free" (the only cost is a modest amount of cluster resources). Given the availability of data and computing power, there is simply no reason why MT researchers should not do the same and enjoy the benefits.

Part of our future work is to improve the representation and translation of text by including named entity recognition and using broader and larger vocabularies, as well as techniques from natural language processing. For example, German and Turkish word translations may be very noisy due to compounding and other morphological phenomena, so we plan on experimenting with more fine-tuned preprocessing. Using learning methods to find better random projections has been shown to work well [112] and we are interested in adapting that approach to our system, as well as explore other extensions to the LSH-based techniques to compute signatures. In terms of evaluation, it would be very interesting to use crowd-sourcing methods (e.g., Mechanical Turk) to provide a more realistic assessment of the quality of both document pairs (i.e., output of the first phase) and the parallel text (i.e., output of the second phase). Based on this evaluation, we can explore if there is a correlation between high-quality translations in the output and improvements to BLEU score on a given test set, which might provide insights on how to improve parallel text extraction approaches in general. Finally, we hope to show the scalability of our algorithm on even more language pairs and even larger datasets.

Chapter 4: Translating to Search: Context-Sensitive Query Translation for Cross-Language Information Retrieval

4.1 Overview

In this chapter, we describe a novel approach to cross-language information retrieval (CLIR) using components of a modern statistical machine translation (MT) system. In CLIR, the query and documents are presented in different languages (referred to as source and target languages, respectively), therefore either the query needs to be translated into the document language, or vice versa.[1] Our approach implements query translation, although it could be adapted to translate documents instead.

In order to perform this translation, it is common to make an independence assumption between the tokens of the query, so that each can be translated independently. However, if the entire query is considered together, it is possible to produce more appropriate translations, resulting in improved CLIR effectiveness. Based on this motivation, we introduce a framework to learn term translation probabilities

[1] It is also possible to perform a combination of both translation directions [145].

that are sensitive to the query context, where term is defined as a class of tokens, following the definition of Manning et al. [91]. We achieve this by taking advantage of the internal representation of a statistical MT system, exploiting the benefits of both translation and language models in various ways. This approach is referred to as *context-sensitive* query translation.

This chapter is organized as follows: First, a strong baseline approach for query translation in CLIR is described in Section 4.2. We then introduce our proposed methods in Section 4.3. Finally, an evaluation of our approach on three different cross-language retrieval tasks is presented in Section 4.4.

4.2 Context-Independent Query Translation

As a baseline, we consider the technique presented by Darwish and Oard, which is a state-of-the-art method for translating a vector representation from one language space into another [31]. We represent a given source-language query $s = s_1, s_2, ...$ in the target language (i.e., the document language) as a Probabilistic Structured Query (PSQ), where each token s_j is represented by its translations in the target language, weighted by the bilingual translation probability. These token-to-token translation probabilities are learned independently from a separate parallel bilingual text using automatic word alignment techniques, as described in Section 2.1.1.

In order to build a term translation probability distribution suitable for CLIR, we perform some cleaning on the probabilities output by the word aligner. For each source-language term s_j, we sort its possible translations by decreasing probability into a list $[t_{i_1}, t_{i_2}, \ldots]$. In sorted order, these translation terms are included into a new probability distribution, called Pr_{token}, until (1) the probability falls below a threshold L, or (2) the cumulative sum of probabilities reaches C, or (3) the number of translations in the distribution exceeds H. Finally, in order to generate a proper probability distribution, we normalize probabilities.[2] Below is the mathematical formulation of the term translation probability distribution:

$$Pr_{\text{token}}(t_{i_k}|s_j) = \begin{cases} 0 \text{ if } (k > H) \vee \big(p(t_{i_k}|s_j) \leq L\big) \vee \big(\sum_{l=1}^{k-1} p(t_{i_l}|s_j) > C\big) \\ \frac{1}{\xi_j} p(t_{i_k}|s_j) \text{ otherwise} \end{cases} \tag{4.1}$$

where ξ_j is the normalization factor, given by the sum of all probabilities added to the distribution from target-language vocabulary V_t:

$$\xi_j = \sum_{x \in V_t} Pr_{\text{token}}(x|s_j) \tag{4.2}$$

In IR, we collect statistics of terms within documents, such as term frequency (tf) and document frequency (df), and use those to score query-document relevance. In CLIR, these statistics are available for target-language terms only, since all documents are written in the target language. Therefore, Darwish and Oard proposed a mechanism to translate these values into the source-language vocabulary space, using term translation probabilities (e.g. Pr_{token}) [31].

In this approach, the score of document d, given source-language query s, is

[2]In a proper probability distribution, all probabilities are positive, and the sum is 1.

computed by the following equations:

$$\text{Score}(d|s) = \sum_{j=1}^{\#\text{ terms}} \text{BM25}(\text{tf}(s_j, d), \text{df}(s_j)) \tag{4.3}$$

$$\text{tf}(s_j, d) = \sum_{t_i} \text{tf}(t_i, d) Pr_{\text{token}}(t_i|s_j) \tag{4.4}$$

$$\text{df}(s_j) = \sum_{t_i} \text{df}(t_i) Pr_{\text{token}}(t_i|s_j) \tag{4.5}$$

As shown above, we use the Okapi BM25 term weighting function, due to its superior effectiveness in empirical evaluations. Although we have decided to use Okapi BM25 in our approach, any other weighting function can be substituted into Equation 4.3 in principle. For reference, we provide the formula for the BM25 function, given some term w, and a document d from a collection of documents $\mathcal{C}$:

$$\text{BM25}(\text{tf}(w, d), \text{df}(w)) =$$
$$\frac{(k_1 + 1)\text{tf}(w, d)}{k_1\big((1-b) + b\frac{|d|}{\text{avg}_{d' \in \mathcal{C}}|d'|}\big) + \text{tf}(w, d)} \log \frac{N - \text{df}(w) + 0.5}{\text{df}(w) + 0.5} \tag{4.6}$$

where parameters k_1 and b define how much we penalize longer documents. In our work, we fixed parameters to $k_1 = 1.2$, $b = 0.75$, as this combination has been previously shown to work well.

Example Let us demonstrate the underlying representation of this query translation model by an example. Following an Indri-like [96] notation, Figure 4.1 shows how the translation of English query *Maternal leave in Europe* is represented as a

PSQ under this model, for target language French.[3]

```
#comb(#weight(0.74 matern, 0.26 maternel)
      #weight(0.49 laiss, 0.17 quitt, 0.09 cong,
              0.08 part, 0.04 abandon, 0.04 voyag, ...)
      #weight(0.91 europ, 0.09 européen))
```

Figure 4.1: A PSQ, representing the translation of *Maternal leave in Europe* using Pr_{token}.

The `#comb` operator corresponds to the sum operation in Equation 4.3, whereas the `#weight` operator represents the weighted sum in Equations 4.4 and 4.5. Each of the three `#weight` structures represents the translation of one non-stop word query token: *maternal*, *leave*, and *Europe*. Within each `#weight` structure, terms follow their probabilities, which correspond to the Pr_{token} values in these equations. Since the translation distribution for the source term *leave* is unaware of the context *maternity leave*, candidates that occur most frequently in general text, such as *laisser* (Eng. let go, allow) and *quitter* (Eng. quit), have higher probabilities than more appropriate candidates, such as *cong*é (Eng. vacation, day off). Also, even though we do not present all of them above, there are 10 candidates for the translation of *leave*, due to the many senses associated with this word.

Discussion A drawback of this baseline query translation approach (hereafter referred to as "token-based") is its *context-independent* translation model, due to its assumption that the translation of a token is independent of the rest of the query. With less contextual information, this model generates many translation candidates,

[3]English and French tokens are stemmed in preprocessing.

causing an increased amount of ambiguity in the representation (e.g., Figure 4.1). The strength of context-sensitive CLIR approaches is to cleverly down-weight or discard translation candidates that are not appropriate within the given context.

At the same time, it is not possible to entirely disambiguate language, even with context-sensitive models, since there will be more than one correct way to translate many words. Hence, the ability to represent all plausible translations in a probabilistic manner is actually helpful in retrieval, since it provides guidance based on statistics obtained from training data. We call such models *ambiguity-preserving*, since the probabilistic representation *preserves the ambiguities* associated with translating the query. The strength of probabilistic token-based CLIR approaches is to preserve ambiguities within the final representation. It has been shown that such ambiguity-preserving representations are beneficial for translation [37].

In this chapter, based on the intuition that it is essential to (i) be sensitive to query context, and (ii) preserve useful linguistic ambiguities probabilistically, we describe methods to construct translation probabilities conditioned on the query context (i.e., $Pr(t_i|s_j, s)$ where s represents query context). We exploit existing statistical MT techniques to achieve this, and incorporate them into the existing token-based PSQ representation illustrated above. As a result, we mix the context-sensitive translation choices of MT with ambiguity-preserving PSQ representations of traditional CLIR, combining the strengths of both fields for this task.

4.3 Context-Sensitive Query Translation

In this section, we explore several ways to improve the token-based query translation model discussed above, by exploiting the internal representations of an MT system. We view the MT pipeline as three subsequent processes, as shown in Figure 4.2: word alignment, translation modeling, and decoding. Let us first briefly describe these three components, and discuss how they relate to the task of CLIR.

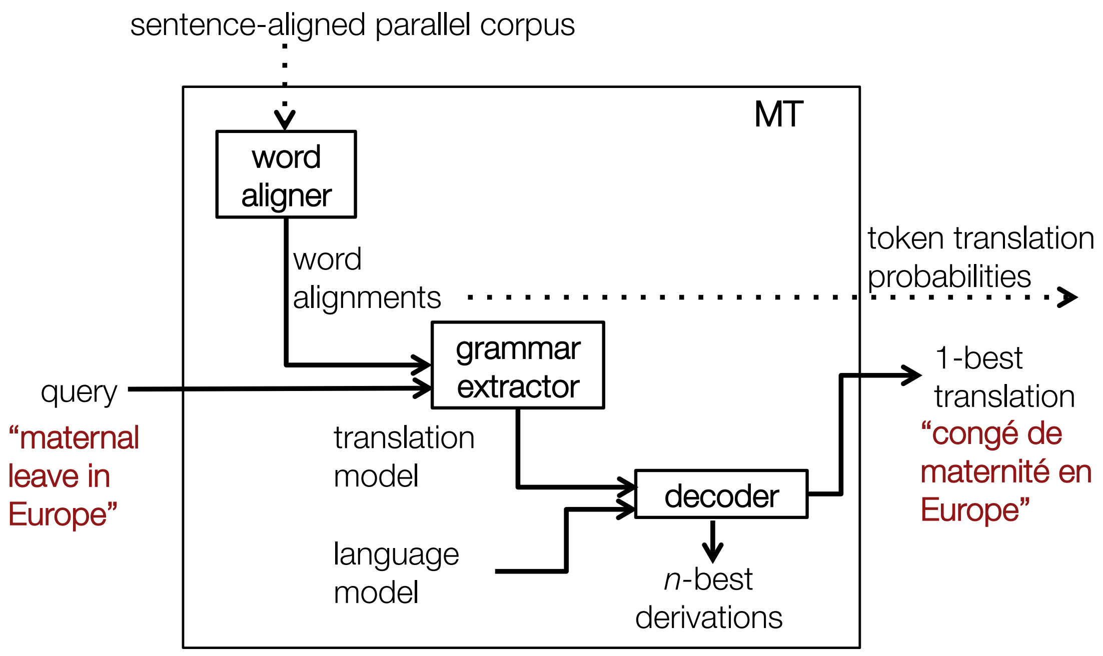


Figure 4.2: An overview of a modern statistical MT pipeline.

As discussed in Section 2.1.1, the word alignment process takes a parallel corpus, and learns word alignments that maximize data likelihood. As a by-product of the learning process, token translation probabilities are generated (i.e., parameters of the IBM models). These probabilities have been successfully applied to CLIR, as we described in the baseline approach (Section 4.2).

However, MT systems use the word alignments to build more sophisticated translation models, such as a phrase translation table or Synchronous Context-Free Grammar (SCFG). These rich representations have not been explored by CLIR researchers for query or document translation. In Section 4.3.1, we describe how one can directly use these representations to perform query translation.

The third major component of the MT pipeline is the decoder, which performs a search through the hypothesis space, to find top-scoring translations efficiently. By combining the language and translation models, the decoder can take both adequacy and fluency into account when scoring hypotheses, allowing better assessment of quality. In Section 4.3.2, we explain how to use the n top hypotheses (output by the decoding process) to perform query translation.

Finally, since each of these approaches has complementary advantages, we present an interpolation model in Section 4.3.3.

4.3.1 Learning Probabilities from Translation Model

Motivation Although the word alignment and decoding components are mostly similar across different MT approaches and implementations, the representation of the translation model varies substantially. In this section, we will focus on the the translation models of two state-of-the-art MT models, a *flat* phrase-based MT (PBMT) system and a *hierarchical* PBMT system, in the context of query translation. For a detailed review of statistical MT approaches, see Lopez's survey [86].

Regardless of whether an MT system is flat or hierarchical, the translation model consists of a set of translation rules in the following format:

rule r: α `||` β `||` $\mathcal{A}$ `||` $\ell(r)$

stating that source-language text α can be translated into target-language text β, with an associated likelihood value $\ell(r)$,[4] an unnormalized estimate of the event probability. We call α the Left-Hand Side (LHS) of the rule, and β the Right-Hand Side (RHS) of the rule. $\mathcal{A}$ represents the word alignments, which is a many-to-many alignment mapping between tokens on the LHS and RHS of the rule.

In flat MT systems, the LHS and RHS of a rule only contains text, one or more tokens on each side. These multi-token expressions are usually called *phrases*, although they are not linguistically motivated entities (selected based entirely on statistical importance). As a result, rules are typically referred to as *phrase pairs* under these models, and the set of rules in the translation model form the *phrase translation table*. In order to avoid confusion and easily contrast with hierarchical MT, we will refer to the set of rules within a flat MT system as a *flat grammar*.

In hierarchical MT systems, rules take a slightly different form:

hierarchical rule r: `[X]` `||` α `||` β `||` $\mathcal{A}$ `||` $\ell(r)$

which indicates that the context free expansion $X \rightarrow \alpha$ in the source language occurs synchronously with $X \rightarrow \beta$ in the target language. These rules form a formal model of translation, called a SCFG or *hierarchical grammar*, which differs from a flat grammar in terms of rule expressivity: the LHS and RHS are allowed to contain

[4]In practice, there are usually additional features that represent various aspects of the rule.

one or more nonterminals, each acting as a variable that can be expanded into other expressions using the SCFG. In other words, each rule describes a context-free expansion on both source and target sides, carried out in a recursive manner to generate translation hypotheses.

Consider the following two rules in order to illustrate the differences between flat and hierarchical translation models:[5]

R_1.`[S] || [X] leav in europ || cong de [X] en europ || 1-0 2-3 3-4 || 1`

R_2.`[X] || matern || matern || 0-0 || 0.69`

Applying these two rules consecutively (i.e., by expanding non-terminal variable `[X]` in R_1 by the rule R_2), we can translate *matern leav in europ* into *cong de matern en europ*. More generally, `[X]` in R_1 allows an arbitrarily long part of the sentence to be moved from the left of the sentence in English to the middle of the sentence in French,[6] even though it generates a single token (i.e., *matern*) using R_2 in this particular example. Using such rules, a hierarchical grammar (i.e., SCFG) can capture some distant dependencies in a sentence that may not be realized in flat grammars.

Given some input text, a suffix array can be used to efficiently extract all applicable rules from a SCFG [85]. Similarly, a binarized representation of the

[5]Tokens in the example have been stemmed as part of the preprocessing.

[6]This may not be in true in practice, where it is common to put length restrictions on phrases. Besides, moving around very long phrases might not be desirable when translating between most language pairs. Still, we are pointing out that such transformations *can* be represented in the SCFG formalism.

flat grammar is used to filter rules with respect to a provided input text [70]. By only leaving the small portion of the grammar that applies to the input text, this procedure significantly reduces the memory footprint in the decoding phase. We can exploit this feature of modern MT systems to learn term translation probabilities.

Method We propose the following method to construct a probability distribution from a set of rules, either from a flat or hierarchical grammar: For each rule, we use the word alignments to determine which source token translates to which target token(s). Iterating over all grammar rules that apply to a given query, we construct a probability distribution for each token that appears on the LHS of any rule.

More specifically, given a grammar $\mathcal{G}$ and query s, we first obtain the subset of rules $\mathcal{G}(s)$ for which the source side pattern matches s (by either using suffix array extraction or filtering techniques described above). Once $\mathcal{G}(s)$ is obtained, the process is identical when using flat or hierarchical MT systems: For each rule r in $\mathcal{G}(s)$, we identify each source token s_j on the LHS of r, ignoring any non-terminal symbols. From the word alignment information included in the rule structure, we can find all target tokens that s_j is aligned to.

Multiple Alignment Heuristics When s_j is aligned to multiple target tokens in a rule, it is not obvious how to distribute the probability mass. One approach is to treat each alignment as an independent event with the same probability (equal to the likelihood of rule r). We call this the *one-to-one* heuristic, and introduce two alternatives due to the following drawback: The target tokens aligned to s are not usually independent. For example, the token *brand* may be aligned to three tokens *marque*, *de*, *fabrique* (En. *brand, of, factory*), which is an appropriate translation when put together. Even if *de* is discarded as a stop word, the one-to-one heuristic will learn the token pair (*brand*, *fabrique*) incorrectly. An alternative heuristic is to ignore these cases altogether, assuming that good translation pairs will appear in other rules, so that discarding these cases would not cause any harm: we call this the *one-to-none* technique. A third approach is to combine the target tokens into a multi-token expression. So, in the above example, we would learn the translation of *brand* as *marque de fabrique*, which is a useful mapping that we might not learn otherwise. We combine the target tokens if they are either consecutive, or they are separated by one or more stop words (e.g., *de* in French). The latter case applies in the above example: *marque* and *fabrique* are not consecutive, but the only token in between them is a stop word. We call the third technique *one-to-many*, and compare these three heuristics (i.e., one-to-none, one-to-one, one-to-many) in our evaluation.

After processing all rules in a similar fashion (using either multiple alignment heuristic), we have accumulated likelihood values for all token pairs we have observed. From these values, we can gather a list of possible translations (and associated likelihood values) for each source term that has appeared in any of the rules. We can then convert each such list into a probability distribution by normalizing the likelihood scores. We call this distribution Pr_{SCFG} if using a hierarchical MT system, or Pr_{PBMT} if the underlying MT model is flat.

Below is the formulation of the construction process described above. The subscript SCFG/PBMT is used in expressions that apply to both underlying MT models.

$$Pr_{\text{SCFG/PBMT}}(t_i|s_j) = \frac{1}{\psi} \sum_{\substack{r \in \mathcal{G}(s) \\ s_j \leftrightarrow t_i \text{ in } r}} \ell(r) \tag{4.7}$$

$$\text{tf}(s_j, d) = \sum_{\{t_i | s_j \leftrightarrow t_i \in G(s)\}} \text{tf}(t_i, D) Pr_{\text{SCFG/PBMT}}(t_i|s_j) \tag{4.8}$$

$$\text{df}(s_j) = \sum_{\{t_i | s_j \leftrightarrow t_i \in G(s)\}} \text{df}(t_i) Pr_{\text{SCFG/PBMT}}(t_i|s_j) \tag{4.9}$$

where ψ is the normalization factor and $s_j \leftrightarrow t_i$ represents an alignment between tokens s_j and t_i. Mapping tf and df statistics from source to target vocabulary is achieved by replacing Pr_{token} with $Pr_{\text{SCFG/PBMT}}$ in Equations 4.4 and 4.5.

Example As an example, let us compute Pr_{SCFG} for the second token in our running example query, *Maternal leave in Europe* (which is preprocessed into *matern leav europ*). Assume that Figure 4.3 is the set of SCFG rules that contain the token *leav*, extracted from the translation model for this specific query.[7]

```
[X] || leav || cong || 0-0 || 0.38
[X] || leav || quitt || 0-0 || 0.08
[X] || leav || laiss || 0-0 || 0.27
[X] || [X] leav || [X] laiss || 1-1 || 0.22
[X] || [X] leav || cong [X] || 1-0 || 0.07
[X] || [X] leav || [X] en laiss || 1-2 || 0.02
[X] || [X] leav || [X] elle quitt || 1-2 || 0.01
[X] || [X] leav || [X] en train de quitt || 1-4 || 0.01
[X] || leav || quitt cet assembl || 0-0 0-2 || 0.01
```

Figure 4.3: A subset of the SCFG as a toy example.

[7] In reality, this is only a small portion of the rules, but we omit the rest for demonstration purposes.

In order to construct a translation distribution for term *leav*, we iterate over rules and accumulate values for each translation candidate. The candidate in the first rule is *cong* (Eng. vacation, day off), thus we accumulate a value of 0.38 for the distribution $Pr_{\text{SCFG}}(cong|leav)$. Similarly, we process the remaining rules and add values for two other translation candidates: *laiss* (Eng. let go, allow) and *quitt* (Eng. quit). In the final rule, the approach depends on the heuristic choice: If we apply the one-to-one strategy, we add 0.01 for each candidate, *quitt* and *assembl*.[8] In this case, the final distribution is computed as follows (see Equation 4.7):

$$\psi = (0.38 + 0.07) + (0.08 + 0.01 + 0.01 + 0.01) + (0.27 + 0.02) + 0.01 = 0.86$$

$$Pr_{\text{SCFG}}(\texttt{cong}|\texttt{leav}) = (0.38 + 0.07)/0.86 \sim 0.52$$

$$Pr_{\text{SCFG}}(\texttt{quitt}|\texttt{leav}) = (0.08 + 0.01 + 0.01 + 0.01)/0.86 \sim 0.13$$

$$Pr_{\text{SCFG}}(\texttt{laiss}|\texttt{leav}) = (0.27 + 0.02)/0.86 \sim 0.34$$

$$Pr_{\text{SCFG}}(\texttt{assembl}|\texttt{leav}) = 0.01/0.86 \sim 0.01$$

If one-to-many is being applied, we add 0.01 to a single multi-token term, *quitt*

[8] *cet* is a stop word in French, therefore it is ignored.

cet assembl, yielding a different computation:

$$\psi = (0.38 + 0.07) + (0.08 + 0.01 + 0.01) + (0.27 + 0.02) + 0.01 = 0.85$$

$$Pr_{\text{SCFG}}(\texttt{cong}|\texttt{leav}) = (0.38 + 0.07)/0.85 \sim 0.53$$

$$Pr_{\text{SCFG}}(\texttt{quitt}|\texttt{leav}) = (0.08 + 0.01 + 0.01)/0.85 \sim 0.12$$

$$Pr_{\text{SCFG}}(\texttt{laiss}|\texttt{leav}) = (0.27 + 0.02)/0.85 \sim 0.34$$

$$Pr_{\text{SCFG}}(\texttt{quitt cet assembl}|\texttt{leav}) = 0.01/0.85 \sim 0.01$$

Finally, we do not make any accumulation for the last rule when heuristic one-to-none is applied, producing this distribution:

$$\psi = (0.38 + 0.07) + (0.08 + 0.01 + 0.01) + (0.27 + 0.02) = 0.84$$

$$Pr_{\text{SCFG}}(\texttt{cong}|\texttt{leav}) = (0.38 + 0.07)/0.84 \sim 0.54$$

$$Pr_{\text{SCFG}}(\texttt{quitt}|\texttt{leav}) = (0.08 + 0.01 + 0.01)/0.84 \sim 0.12$$

$$Pr_{\text{SCFG}}(\texttt{laiss}|\texttt{leav}) = (0.27 + 0.02)/0.84 \sim 0.35$$

In addition to these toy examples, Figures 4.4 and 4.5 show the translation probabilities as learned from the entire set of rules, using hierarchical and flat MT systems, respectively. In both cases, we used the one-to-one alignment heuristic.

Discussion Pr_{token} and $Pr_{\text{SCFG/PBMT}}$ both describe the probability of a target-language token given a source-language token, but differ by how the probability values are computed. For both approaches, we start from a large, sentence-aligned

```
#comb(#weight(0.68 matern, 0.06 maternel, ... )
      #weight(0.36 cong, 0.25 laiss, 0.11 quitt, ... )
      #weight(0.90 europ, 0.07 européen, ... ))
```

Figure 4.4: A PSQ, representing the translation of query "Maternal leave in Europe" using Pr_{SCFG}.

```
#comb(#weight(0.68 matern, 0.06 maternel, ... )
      #weight(0.33 cong, 0.22 laiss, 0.11 quitt,
              0.04 part, 0.02 abandon, 0.02 voyag, ... )
      #weight(0.90 europ, 0.05 européen, ... ))
```

Figure 4.5: A PSQ, representing the translation of query "Maternal leave in Europe" using Pr_{PBMT}.

bilingual corpus, which is first word-aligned. From these word alignments, one can directly generate token translation probabilities, which correspond to Pr_{token}. In order to create $Pr_{\text{SCFG/PBMT}}$, we take advantage of the MT system's ability to induce a more sophisticated translation model (i.e., flat or hierarchical translation grammar), and filter the space with respect to given input text (i.e., query).

As a result, $Pr_{\text{SCFG/PBMT}}$ is different than Pr_{token} because it takes query context into account. Basically, we only look at the part of the grammar that applies to the source query text, and therefore create a bias in the probability distribution based on this context. This results in context-sensitive translations, and it also reduces the ambiguity associated with how to translate each token. For example, we might learn 10 different ways of translating *leave* from the parallel corpus, but only few might be left after filtering the choices that do not apply to the query context.

The context-aware nature of this so-called "grammar-based" approach is why the translation distribution for the example query is different than Pr_{token}. For example, the distribution of *leave* (Figure 4.4) shifts toward the more appropriate translation *congé* as a result of this approach, as opposed to the distribution in Figure 4.1. Moreover, although not shown in the figure, the number of translation candidates decreases from 10 to 5 with the hierarchical grammar-based approach.

There is also a tradeoff between using either of the two MT models for CLIR. While hierarchical models allow more flexibility in representing linguistic phenomena, this usually makes the decoding slower in practice [86].[9] On the other hand, simpler flat PBMT models do not possess the expressiveness of hierarchical models. Also, due to the lack of variables in the rule representation, the translation model contains a larger number of rules, resulting in a verbose representation. Figures 4.4 and 4.5 clearly illustrate this effect, resulting in extra noise in the translation of *leave* for the latter case.

4.3.2 Learning Probabilities from n-best Derivations

Motivation In MT, decoding is the process of searching and ranking translations. In statistical MT systems, each sequence of rules that covers the entire input is called a *derivation*, D, and produces a translation candidate, t, which is scored by a log-linear combination of features. One can add many features to score a given candidate, but two features are essential: the translation model score ($\text{TM}(t, D|s)$) is the product of rule likelihood values and indicates how well the candidate preserves

[9]We did not experience a significant speed difference in our experiments. However, a fair comparison is not possible when speed strongly depends on various implementation details and parameters. We are not aware of a direct comparison between the two approaches in the MT literature.

the original meaning, whereas the language model score LM(t) indicates how fluent the translation is. To control computational complexity, most decoders search for the most probable derivation (as opposed to the most probable string):

$$\begin{aligned}
t^{(1)} &= \arg\max_{t} \big[\max_{D \in \mathcal{D}(s,t)} \ell(t, D|s) \big] && (4.10) \\
&= \arg\max_{t} \big[\max_{D \in \mathcal{D}(s,t)} \log \ell(t, D|s) \big] && (4.11) \\
&= \arg\max_{t} \big[\max_{D \in \mathcal{D}(s,t)} (\log \mathrm{TM}(t, D|s) + \log \mathrm{LM}(t)) \big] && (4.12) \\
&= \arg\max_{t} \big[\log \mathrm{LM}(t) + \max_{D \in \mathcal{D}(s,t)} \sum_{r \in D} \log \ell(r) \big] && (4.13)
\end{aligned}$$

where $\mathcal{D}(s,t)$ is the set of possible derivations that generate the pair of sentences (s,t) (e.g., the sequence of four rules that translate the example query in Section 2.1.2 forms one such derivation).

One way to use the decoder for CLIR is to replace the source query with its most probable translation. In this "one-best" query translation approach, Equations 4.3–4.5 simplify to:

$$\mathrm{Score}(d|s) = \sum_{i=1}^{m} \mathrm{BM25}(\mathrm{tf}(t_i^{(1)}, d), \mathrm{df}(t_i^{(1)})) \qquad (4.14)$$

Although this one-best strategy has been shown to work well in many cases, it discards potentially useful information generated by the decoder. Decoders produce a set of candidate sentence translations in the process of computing Equation 4.10, so it is possible to generalize our translation model to consider the n most probable translation candidates, instead of the single best one.

Method In order to learn token translation probabilities from the n-best translations, we start by preprocessing the source query s and each candidate translation $t^{(k)}, k = 1 \dots n$.[10] For each source token s_j, we use the derivation information to determine which grammar rules were used to produce $t^{(k)}$, and the word alignments within these rules to determine which target tokens are associated with s_j in the derivation. By doing this for each translation candidate $t^{(k)}$, we construct a probability distribution of possible translations of s_j based on the n query translations. Specifically, if source token s_j is aligned to (i.e., translated as) t_i in the k^{th} best translation, the value $\ell(t^{(k)}|s)$ is added to its probability mass. Similar to $Pr_{\text{SCFG/PBMT}}$, we apply one of the three possible heuristics to follow when a source token is applied to multiple target tokens in a rule. The following formulates how this new probability distribution (called Pr_{nbest}) is constructed:

$$Pr_{\text{nbest}}(t_i|s_j) = \frac{1}{\varphi} \sum_{\substack{k=1 \\ s_j \leftrightarrow t_i \text{ in } t^{(k)}}}^{n} \ell(t^{(k)}|s) \tag{4.15}$$

$$\text{tf}(s_j, d) = \sum_{t_i} \text{tf}(t_i, d) Pr_{\text{nbest}}(t_i|s_j) \tag{4.16}$$

$$\text{df}(s_j) = \sum_{t_i} \text{df}(t_i) Pr_{\text{nbest}}(t_i|s_j) \tag{4.17}$$

where φ is the normalization factor. We should emphasize that Pr_{nbest} is a well-defined probability distribution for each s_j, so if a source token is translated consistently into the same target token in all n translations, then it will have a single translation with a probability of 1.0.

[10]Here, $t^{(k)}$ denotes the k^{th} most likely translation of s, according to the log-linear MT model.

The process to construct Pr_{nbest} is almost identical to $Pr_{\text{SCFG/PBMT}}$, with the only difference being that we iterate over rules used in the derivations of the top n translations. For comparison, Figure 4.6 shows translation probabilities for the same example query, using Pr_{nbest}.

```
#comb(#weight(0.91 matern, 0.09 maternel, ... )
      #weight(1.0 cong)
      #weight(1.0 europ))
```

Figure 4.6: A PSQ, representing the translation of query "Maternal leave in Europe" using Pr_{nbest}.

Discussion Since entire query translations are used as context in Pr_{nbest}, we refer to this CLIR approach as "translation-based". In terms of the MT pipeline (Figure 4.2), probabilities in Pr_{nbest} are learned after the final processing stage, which is decoding. As a result, in contrast with $Pr_{\text{SCFG/PBMT}}$, the language model is incorporated into the query translation process, potentially guiding it into more fluent output.

Of course, language models are trained from well-formed text, typically from formal sources (e.g., news). Therefore, these models are not tuned towards queries, which consist of a few keywords that are not guaranteed to form a fluent sentence or phrase. There are large language models that have been successful in a wide range of applications, yet they increase the memory footprint substantially. Recently, there has been some success on reducing the memory requirements of language models, although this is not a focus of this dissertation [58].

Another implication of learning probabilities after decoding is the need for tuning: this is a non-trivial procedure with a running time typically much longer than decoding itself. Also, there are very few parallel corpora specifically for query text, and tuning MT systems specifically for CLIR has received almost no attention by researchers, except for a recent paper by Nikoulina et al. [105].

As discussed before, the advantage of the token-based approach is the ability to model all of the translational varieties (or ambiguities) existent in the bilingual corpus, although these may be too noisy to properly represent the query translation space. We showed that Pr_{SCFG} reduces some of this ambiguity by incorporating the query context. For Pr_{nbest}, we would expect even further reductions, since we are now only considering the top-scoring n derivations (as opposed to all applicable grammar rules). This is easily observed from the example in Figure 4.6, in which the term *leav* has only one translation, as opposed to 10 and 5 in Pr_{token} and Pr_{SCFG}, respectively. Due to this phenomenon, using Pr_{nbest} might perform worse due to overfitting to the query context (e.g., potentially good translations might be discarded due to low scores by the LM), or better due to irrelevant translation choices being removed.

The overfitting issue is partially mitigated by using the n-best translation derivations, as opposed to the 1-best translation, which treats the MT system as a black box. However, the lack of textual variety in the n most probable derivations is a known issue, caused by the fact that statistical MT systems identify the most probable derivations (not the most probable strings), many of which can correspond to the same surface form. This phenomenon is called "spurious ambiguity" in the MT literature, and it occurs in both flat [71] and hierarchical phrase-based MT systems [25]. For instance, according to Li et al. [79], a string has an average of 115 distinct derivations in Chiang's Hiero system. Researchers have proposed several ways to cope with this situation, and integrating some of these ideas into our CLIR approach might be worthwhile to explore in the future.

4.3.3 Combining Sources of Evidence

All three approaches for query translation (i.e., token-based, grammar-based, and translation-based) have complementary strengths, providing a tradeoff between context-sensitive and ambiguity-preserving translation approaches. Figure 4.7 illustrates this tradeoff between the different CLIR models introduced in this section. On one side of the spectrum, we have the token-based CLIR model, which nicely represents the varieties of language through a probabilistic representation, but assumes independence between query terms. This results in a ambiguity-preserving, yet context-independent approach. On the other end, we have the translation-based models (1-best and n-best), which rely heavily on query context during translation, and this may result in better or worse performance, based on how well that query is handled by the MT models. In the middle, grammar-based CLIR provides a compromise between the two extremes, preserving some of the ambiguity represented in the translation model but also conditioning the translation process on query context.

In order to combine the strengths of each model, we introduce a unified CLIR

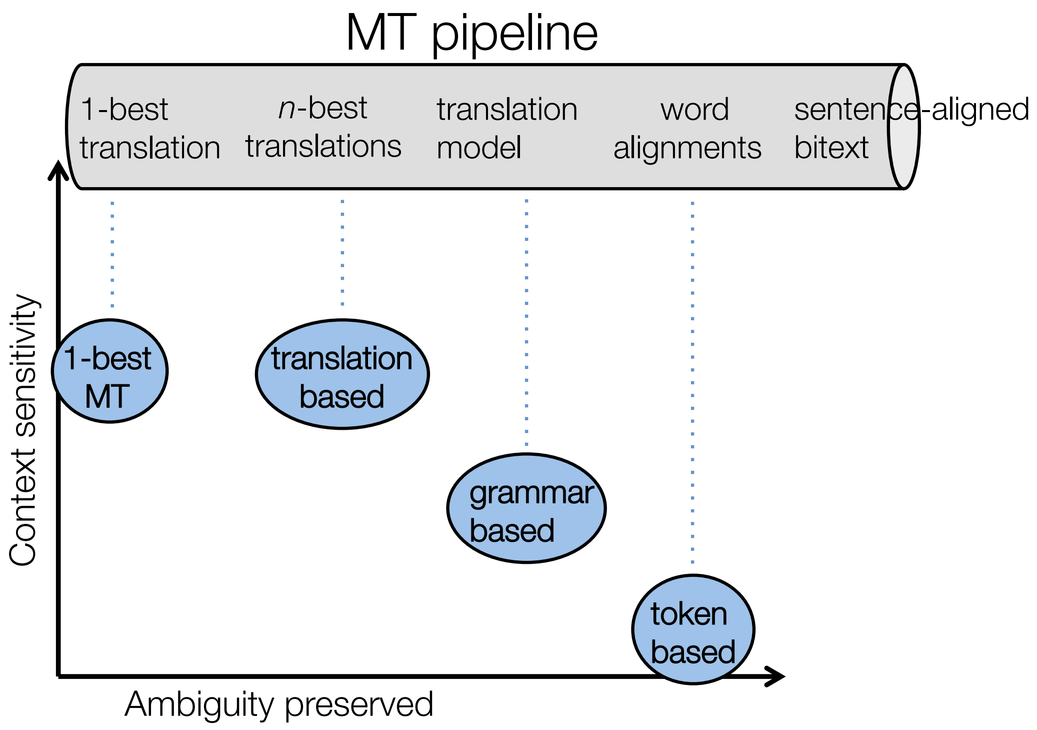


Figure 4.7: A comparison between the CLIR models and their correspondence within a hierarchical MT pipeline.

model by performing a linear interpolation of the three probability distributions:

$$\begin{aligned} Pr_{\text{c}}(t_i|s_j;\lambda_1,\lambda_2) = &\lambda_1 Pr_{\text{nbest}}(t_i|s_j) \\ &+\lambda_2 Pr_{\text{SCFG/PBMT}}(t_i|s_j) \\ &+(1-\lambda_1-\lambda_2)Pr_{\text{token}}(t_i|s_j) \end{aligned} \tag{4.18}$$

where λ_1 and λ_2 define how much weight is assigned to the translation-based and grammar-based models, respectively. Replacing Pr_{token} with Pr_{c} in Equations 4.4 and 4.5 gives us the document scoring formula for the interpolated model.

4.4 Evaluation

We evaluated our system on the latest available CLIR test collections for three languages: TREC 2002 English-Arabic CLIR, NTCIR-8 English-Chinese Advanced Cross-Lingual Information Access (ACLIA), and CLEF 2006 English-French CLIR. For the Arabic and French collections, we used title queries because they are most representative of the short queries that searchers frequently pose to web search engines. Chinese queries in the NTCIR-8 ACLIA test collection are in the form of complete syntactically correct questions, but for consistency we treated them as bag-of-words queries in our experiments with no special processing. The collections contain 383,872, 388,589 and 177,452 documents, and 50, 73, and 50 topics, respectively. Details of each collection are summarized in Table 4.1.

Language	**Collection**		**# topics**	**Training data**	
	Source	**Size**		**Source**	**Size**
Arabic	TREC 2002	383,872	50	GALE	3.4m
Chinese	NTCIR-8	388,589	73	FBIS	0.3m
French	CLEF 2006	177,452	50	Europarl	2.2m

Table 4.1: A summary of the collections used in our evaluations. Query language is English in all three cases.

An English-to-Arabic translation model was learned using 3.4 million aligned sentence pairs from the DARPA GALE evaluation [108], which consists of NIST and LDC releases. An English-to-Chinese translation model was trained on 302,996 aligned sentence pairs from the widely used Foreign Broadcast Information Service

(FBIS) corpus, which is a collection of radio newscasts, provided by LDC (catalog number LDC2003E14).[11] We trained an English-to-French translation model using 2.2 million aligned sentence pairs from the latest Europarl corpus (version 7) that was built from the European parliament proceedings.[12]

For the flat PBMT system, we used the Moses MT system [70], a state-of-the-art open-source toolkit. For the hierarchical MT approach, we used the `cdec` decoder, due to its support for SCFG-based models and its efficient C-based implementation, making it faster than most of the other state-of-the-art systems [36]. Word alignments were learned with GIZA++ [106], using 5 Model 1 and 5 HMM iterations. A Hiero-style SCFG serves as the basis for the hierarchical translation model [25], which was extracted from these word alignments using a suffix array [85]. A 3-gram language model was trained from the English side of the training bitext for Chinese and Arabic, using the SRILM toolkit [133]. For French, we trained a 5-gram LM from the monolingual dataset provided for WMT-12. The Chinese collection was segmented using the Stanford segmenter [138], English topics and the French collection were tokenized using the OpenNLP tokenizer,[13] and Arabic was tokenized and stemmed using the Lucene package.[14] For English and French, we also lowercased text, stemmed using the Snowball stemmer, and removed stop words.

As discussed in Section 4.3, we implemented three techniques to construct a term translation probability distribution: token-based (using Pr_{token}, as described

[11]http://projects.ldc.upenn.edu/TIDES/mt2003.html
[12]http://www.statmt.org/europarl
[13]http://opennlp.apache.org
[14]http://lucene.apache.org

by Equation 4.3), grammar-based (using Pr_{SCFG} or Pr_{PBMT}, as described by Equation 4.7), and translation-based (using Pr_{nbest}, as described by Equation 4.15).[15] We assessed these three approaches (as well as the three heuristics for one-to-many token alignments) by (i) comparing them against each other, and (ii) measuring the benefit of a linear combination, which we call the interpolated approach (using Pr_{c}, as described by Equation 4.18).

We used Mean Average Precision (MAP) as the evaluation metric. The baseline token-based model achieves a MAP of 0.271 for Arabic, 0.150 for Chinese, and 0.262 for French. Direct comparisons to results reported at TREC, NTCIR, and CLEF (respectively) are hard to make because of differences in experimental conditions, but the comparisons we are able to make suggest that these baseline MAP values are reasonable. The best results at those evaluation campaigns often employ blind relevance feedback, multiple lexical resources and/or very long queries. While these techniques can be useful in deployed applications, we have chosen not to run such conditions in order to avoid masking the effects that we wish to study. For Arabic, the best reported results from TREC 2002 were close to 0.400 MAP [43], but those results were achieved by performing query expansion and learning stem-to-stem mappings; our experiment design requires token-to-token mappings (which result in sparser alignments). For Chinese, the NTCIR-8 topics are in the form of questions, and systems that applied question rewriting performed better than those that did not. Also, 15 of the questions are about people, for which our vocabu-

[15]We fixed $C = 0.95, L = 0.005, H = 15, n = 10$ for all models after manually trying a range of values.

lary coverage was not tuned. If we disregard these 15 topics, our baseline system achieves 0.178, close to the best reported results with comparable settings, with a MAP of 0.181 [158]. For French, our baseline achieves a score close to the single reported result at CLEF 2006 that did not incorporate blind relevance feedback (0.261 MAP) [123].

In the remainder of this section, we present several aspects of our detailed evaluation. In Section 4.4.1, we describe a comparison between alternative variants of our implementation, and provide an argument on why we focused on a specific setting. In Section 4.4.2, we present a comparison between the different CLIR models, including effectiveness results and a thorough topic-by-topic analysis. Finally, in Section 4.4.3, we compare the various models in terms of efficiency, using the total running time as the metric for comparison.

4.4.1 A Comparison of System Variants

We performed an assessment of the following variants of our approach: (a) using either a flat or hierarchical MT model, and (b) using either one-to-many, one-to-one, or one-to-none as the method for accumulating probabilities when there are multiple alignments for a token. Experimental results are summarized in Table 4.2, and Arabic (ar), Chinese (zh), and French (fr) runs are presented in subsequent parts.[16] For each case, one row represents results with hierarchical PBMT (using `cdec`), and the other row represents results with flat PBMT (using Moses).

[16]For the one-best approach, one-to-one and one-to-many results are very close, so we discarded the latter for space reasons.

In order to obtain the MAP values labeled as "interpolated," we performed a grid search on the interpolation weights, λ_1 and λ_2, (in increments of 0.1, ranging from 0 to 1) using the interpolated model Pr_{c}. The setting with best MAP value is reported in the table. Notice that these MAP values correspond to the best our model can possibly achieve, since the interpolation weights are optimized on the same set of topics used for testing. We also present results with weights learned from cross-validation experiments in Section 4.4.2.

From Table 4.2, we notice that the two MT models exhibit large differences in the grammar-based approach. We performed a randomized significance test that has been shown to work well for CLIR [131]. According to this test, the `cdec`-based hierarchical MT approach statistically significantly outperforms the Moses-based flat MT approach (with 95% confidence, $p < 0.05$) for all nine settings from {ar,zh,fr}×{one-to-many, one-to-one, one-to-none}, using the grammar-based CLIR model. Furthermore, when we compare the best out of three for each MT approach (i.e., `cdec` and Moses) separately, p-value is still under 0.1. This supports the argument that the SCFG-based translation model is better at representing query translation alternatives for CLIR, possibly due to the more expressive representation (through use of non-terminal variables).

Another implication of using different MT models is grammar size: the translation grammar of a flat MT system is much larger than the SCFG of a hierarchical MT system. This is because certain linguistic transformations can be expressed with a single hierarchical rule, but need to be instantiated under many lexicalizations with a flat model. As a result, the query processing time for Pr_{PBMT} is an order of magnitude higher than Pr_{SCFG}. This might be counter-intuitive, since decoding with flat PBMT is usually faster than decoding with hierarchical MT systems,[17] due to constraints imposed by language modeling. However, we should emphasize that the grammar-based CLIR approach (see Section 4.3.1) uses the internal translation model representation without help from the decoder's optimized search capabilities.

In our experiments, the difference between flat and hierarchical MT models becomes most apparent for the Arabic collection, where the grammar-based model is the most effective. Due to the benefits of a hierarchical grammar, the best result with `cdec` is higher than the best result with Moses, with statistically significant differences for heuristics one-to-none and one-to-one (marked as ‡ in Table 4.2). For the other two collections, we observe a similarly superior performance with the interpolated approach using `cdec`, yet the differences are not statistically significant (i.e., $p > 0.5$).

It is also interesting that the differences between the two MT models are almost non-existent for the 10-best approach, and this is because the final translation output is very similar. Therefore, it might be better to use flat representations for the 10-best approach for efficiency, since the end-to-end translation process might be faster than hierarchical models. We discuss efficiency in more detail in Section 4.4.3.

The second system variant we evaluated is the heuristic for handling source tokens that are aligned to multiple target tokens, namely one-to-none, one-to-one, and one-to-many. Experimental results reveal that the one-to-many method seems

[17]Actual running time depends on system parameters that control how much pruning takes place during search.

to dominate the competition, with the best MAP score in 18 out of 24 cases. Four of the the six cases in which one-to-many is not the best are from Chinese runs. This might be due to the word segmentation task when preprocessing Chinese, as we see that the one-to-one heuristic performs relatively better in Chinese runs.

Based on this analysis, we decided to focus on using a SCFG-based MT approach (`cdec`) and enforce the one-to-many heuristic for the rest of our evaluation.

Language	**MT**	**token**	**grammar**			**1-best**		**10-best**			**interpolated**		
			many	one	none	many/one	none	many	one	none	many	one	none
ar	`cdec`	0.271	0.293	0.282	0.302	0.249	0.249	0.255	0.249	0.248	$0.293^{*\dagger}$	$0.282^{\ddagger}$	$0.302^{*\ddagger}$
	Moses		0.274	0.266	0.273	0.249	0.232	0.264	0.254	0.249	$0.280^{\dagger}$	0.274	0.276
zh	`cdec`	0.150	0.182	0.188	0.170	0.155	0.155	0.159	0.159	0.159	$0.192^{*\dagger}$	0.193^{*}	0.182^{*}
	Moses		0.156	0.167	0.151	0.155	0.146	0.169	0.163	0.163	$0.183^{*\dagger}$	0.177^{*}	0.188^{*}
fr	`cdec`	0.262	0.297	0.288	0.292	0.276	0.235	0.307	0.304	0.295	$0.318^{*\dagger}$	0.314^{*}	0.315^{*}
	Moses		0.264	0.257	0.262	0.297	0.242	0.289	0.300	0.282	0.307^{*}	0.301	0.300

Table 4.2: A summary of experimental results under different conditions, for all three CLIR tasks. Superscripts * and † indicate the result is statistically significantly better than token-based and one-best approach, respectively. Superscript ‡ indicates that using `cdec` is statistically significantly better than Moses (Each cell in the "grammar" column should be marked by ‡, even though we do not for space reasons.). For the 1-best model, heuristics one-to-one and one-to-many yield very close results, so we discarded the latter for space reasons.

4.4.2 A Comparison of CLIR Approaches

In this section, we present an evaluation of the three query translation methods: token-based, grammar-based, and translation-based CLIR, as well as the method based on a linear interpolation of the three individual models.

In order to see the effectiveness of the interpolated model with respect to parameters λ_1 and λ_2, we performed a grid search by applying values in increments of 0.1 (ranging from 0 to 1) to the interpolated model Pr_{c}. Experimental results are summarized in Table 4.3 and illustrated in Figures 4.8, 4.9 and 4.10. In each figure, we provide a scatterplot of MAP scores within a range of values for λ_1 and λ_2. For readability, figures only include a representative subset of λ_2 settings, where different lines represent different values for λ_2. To distinguish the extreme settings of $\lambda_2 = 0$ and $\lambda_2 = 1$, we use a filled circle and square, respectively.

The left edge represents $\lambda_1 = 0$, meaning that we do not use probabilities learned from the n-best derivations (i.e., Pr_{nbest}) in our interpolation. Along the y-axis on the left edge, we see results for various settings of λ_2, which controls how much weight is put on Pr_{SCFG} and Pr_{token}. Within these settings, a particularly interesting one is when λ_2 is set to 0. In this case, the approach is solely based on context-independent translation probabilities (i.e., Pr_{token}), which is the baseline model (call this condition A). When λ_2 is set to 1, we rely on grammar-based term translation probabilities (i.e., Pr_{SCFG}, call this condition B). By contrast, at the right edge, $\lambda_1 = 1$, so we rely only on Pr_{nbest} when translating query terms (call this condition C). For reference, the dotted horizontal line represents simply taking the one-best translation from the MT system (i.e., described by Equation 4.14, call this condition D).

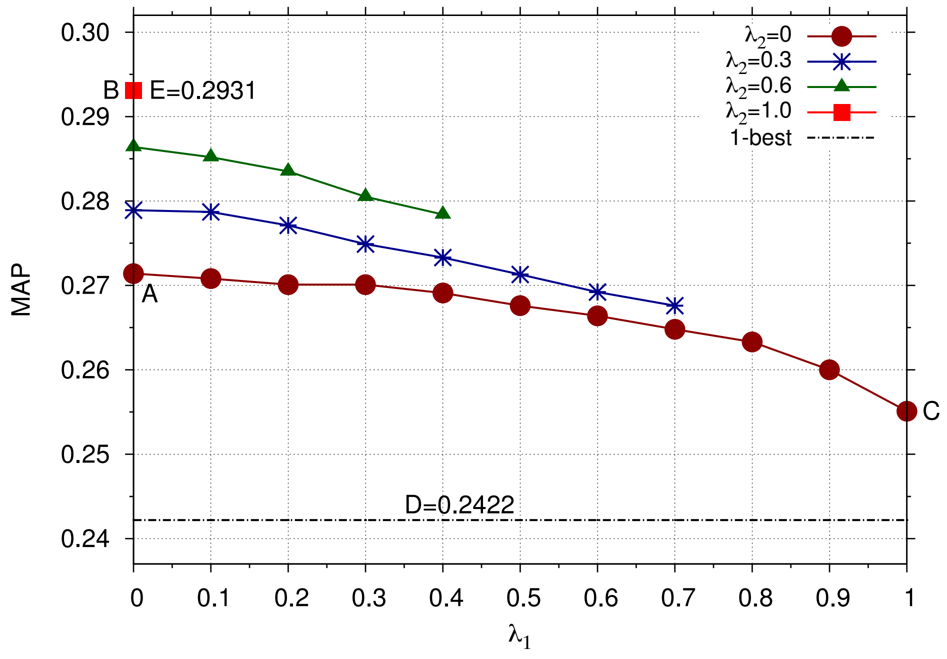


Figure 4.8: Results of a grid search on the parameters of our interpolated CLIR model, evaluated on TREC 2002 English-Arabic CLIR task.

In the case of the Arabic collection, we observe a strictly decreasing trend for the MAP scores as λ_2 decreases, and the best results are obtained when λ_1 is 0 and λ_2 is 1.0 (call the condition with the best MAP score E). In other words, the interpolation yields a maximum 0.293 MAP when it was based entirely on Pr_{SCFG}, ignoring distributions Pr_{token} and Pr_{nbest}. For the Chinese collection, although MAP values are not as high as in the Arabic case, we observe better performance as the weight on Pr_{SCFG} rises. The setting with λ_1=0.1 and λ_2=0.8 yields the best result (MAP=0.192) in our experiments. Experimental results on the French collection are more balanced between the three approaches, with peak effectiveness at λ_1=0.5

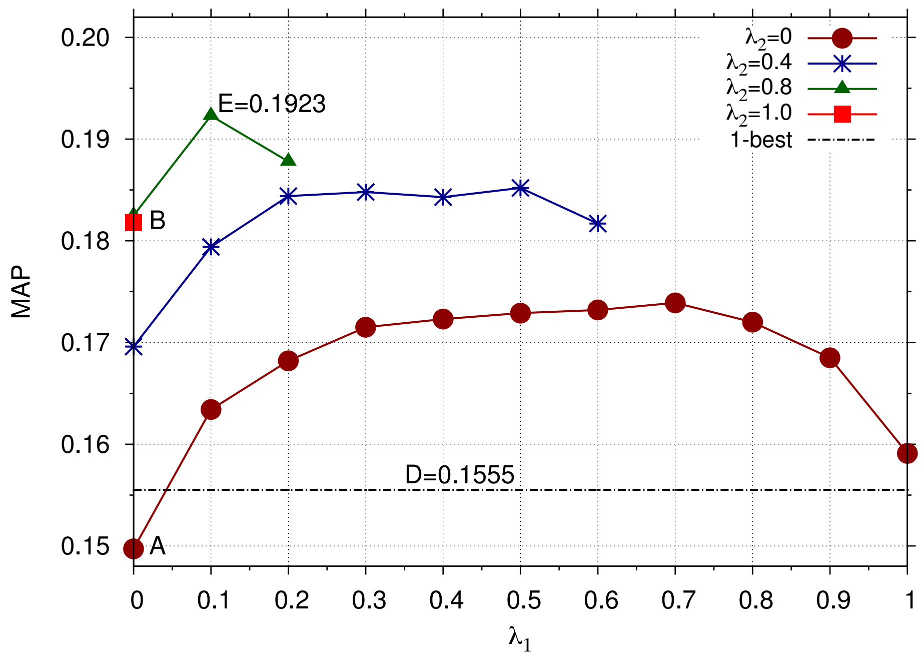


Figure 4.9: Results of a grid search on the parameters of our interpolated CLIR model, evaluated on NTCIR-8 English-Chinese CLIR task.

and λ_2=0.3, resulting in a MAP score of 0.318.

Based on the same randomized significance test proposed by Smucker et al. [131], the interpolated approach (E) outperforms all models with 95% confidence in the Arabic collection, except for the grammar-based approach (B). When we ran the same test on the other two collections, we found that the interpolated approach (E) is significantly better than the baseline (A) and 1-best (D) approaches for French, whereas MAP is significantly higher than all of the individual approaches (A, B, C, and D) for Chinese. These results confirm that the complementary advantages of each model can be combined into a single superior model using our approach.

When the three individual models (conditions A, B and C) are compared (i.e., ignoring the interpolated results), the grammar-based model (B) is statistically

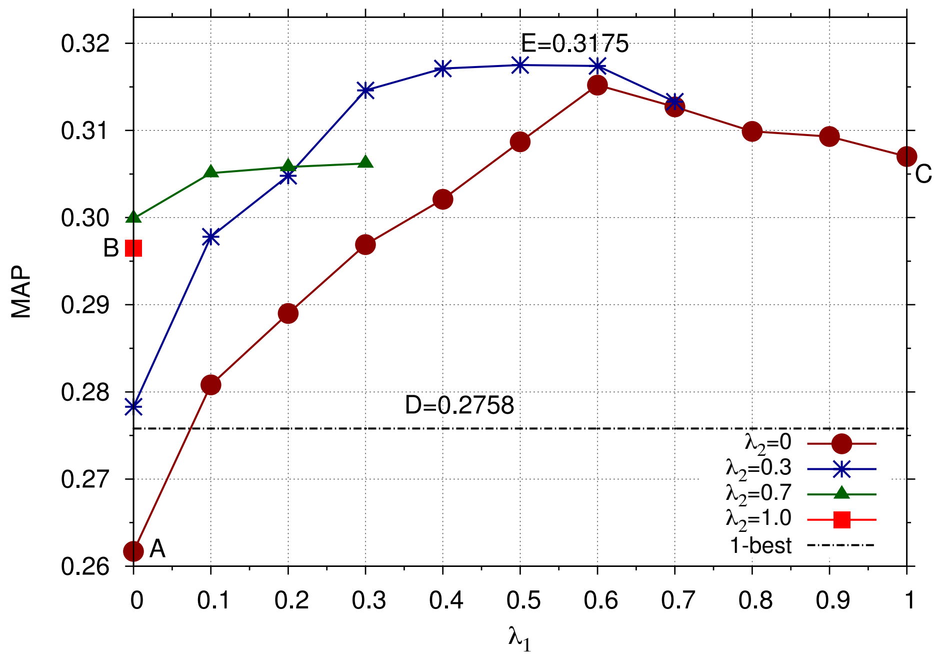


Figure 4.10: Results of a grid search on the parameters of our interpolated CLIR model, evaluated on CLEF 2006 English-French CLIR task.

significantly better than the token-based baseline (A) with 95% confidence for Arabic and Chinese, but statistically indistinguishable from the same baseline model in French. For French, the best retrieval effectiveness results from the n-best full query translation model (C), which is significantly better than the baseline model (A). On the other hand, Pr_{nbest} does not produce significantly better results for Arabic and Chinese. There might be a variety of reasons behind very different results for different languages. The fact that the decoder output in Arabic and Chinese MT systems corresponds to a poorer query translation model than the translation grammar can be explained by a weak language model, a sub-optimal search process in decoding, or inappropriately tuned MT parameters. In any case, this result indicates that there is no "one-size-fits-all" model that outperforms the rest in all

Condition	**Parameters**	**MAP**		
		Arabic	**Chinese**	**French**
A: token-based (Pr_{token})	λ_1=0, λ_2=0	0.271	0.150	0.262
B: grammar-based (Pr_{SCFG})	λ_1=0, λ_2=1	0.293	0.182	0.297
C: translation-based (Pr_{nbest})	λ_1=1, λ_2=0	0.255	0.159	0.307
D: 1-best	-	0.242	0.155	0.276
E: interpolated (Pr_{c})	best $\{\lambda_1, \lambda_2\}$	$0.293^{a,c,d}$	$0.192^{a,b,c,d}$	$0.318^{a,d}$

Table 4.3: A summary of experimental results under different conditions, for all three CLIR tasks. Superscripts indicate if the best result is statistically significantly better than conditions A, B, C, and D.

three collections. The real strength of our approach is therefore to introduce a combination-of-evidence approach that can combine all of these different approaches in a principled manner.

Topic-Specific Analysis For a detailed analysis, we looked at the distribution of the average precision (AP) differences between the various models. We observe that our best interpolated model (E) yields better AP than the token-based baseline model (A) for 36 of the 43 topics (84%) in the Arabic collection, after discarding topics in which there was no noticeable difference (i.e., 7 of the 50 topics exhibited differences of 0.001 or less). For the Chinese collection, the same was true for 41 of 57 topics (72%), with 16 exhibiting a negligible difference. For the French case, the comparable statistic is 30 of 46 (65%), since 4 of the topics exhibited differences of 0.001 or less.

Figures 4.11, 4.13, and 4.15 plot the AP improvement of the best interpolated model (E) and the one-best MT approach (D) over (or the average degradation

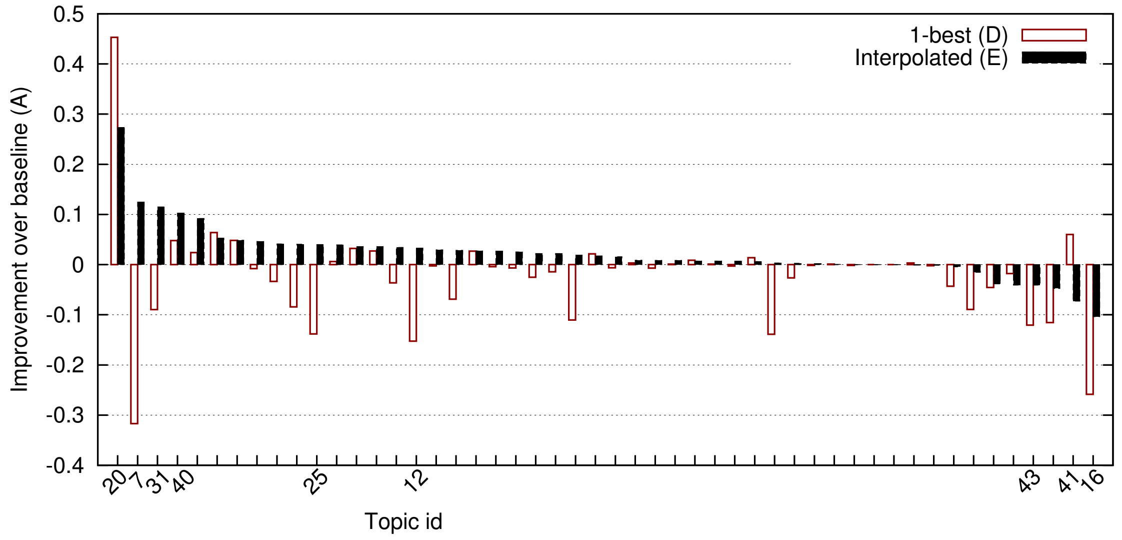


Figure 4.11: Per-topic AP improvement over token-based baseline (condition A) for Arabic: Interpolated and 1-best models.

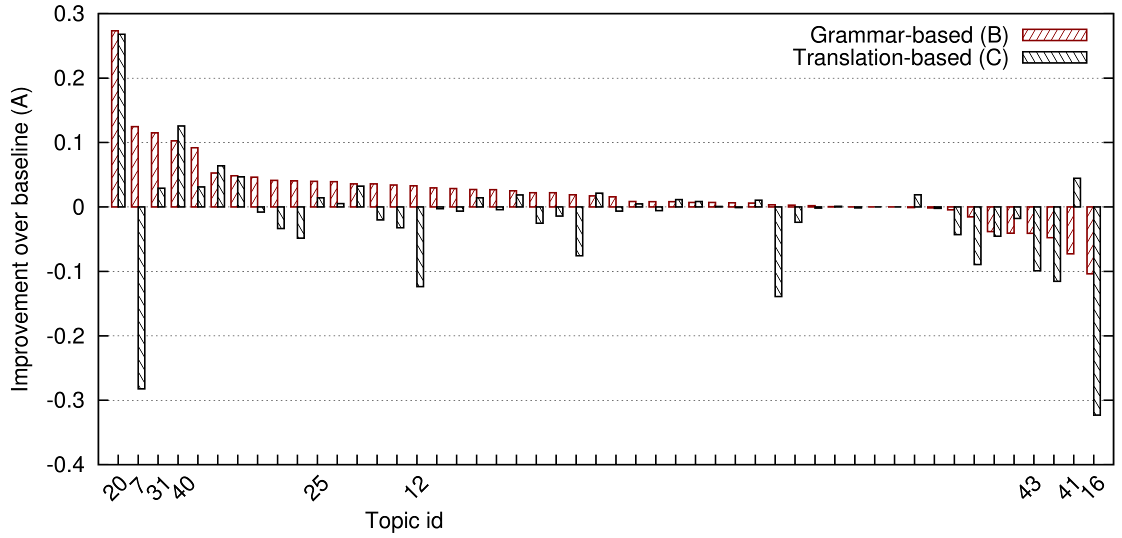


Figure 4.12: Per-topic AP improvement over token-based baseline (condition A) for Arabic: Grammar- and translation-based (i.e., n-best) models.

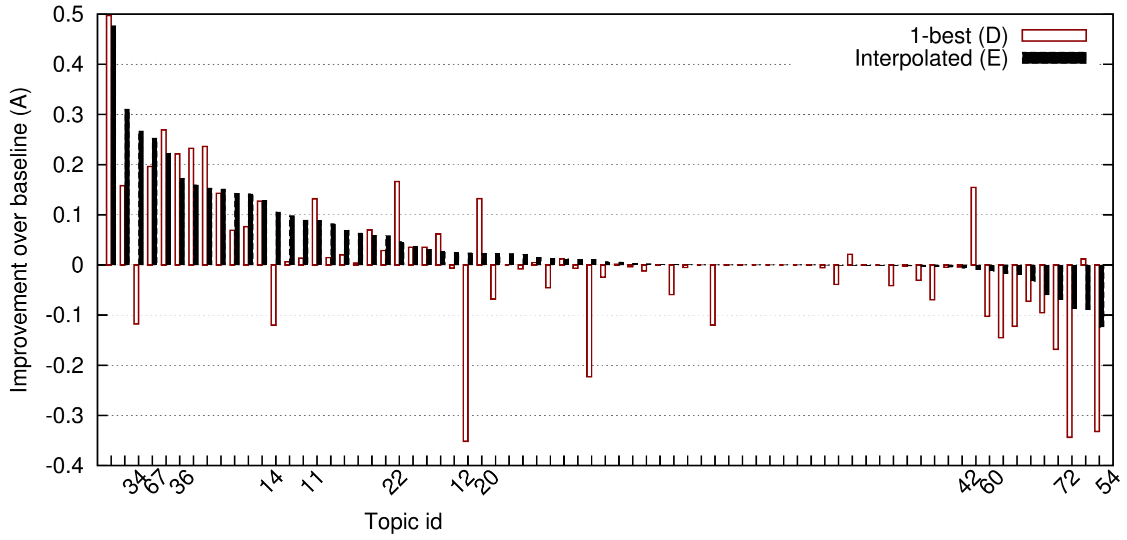


Figure 4.13: Per-topic AP improvement over token-based baseline (condition A) for Chinese: Interpolated and 1-best models.

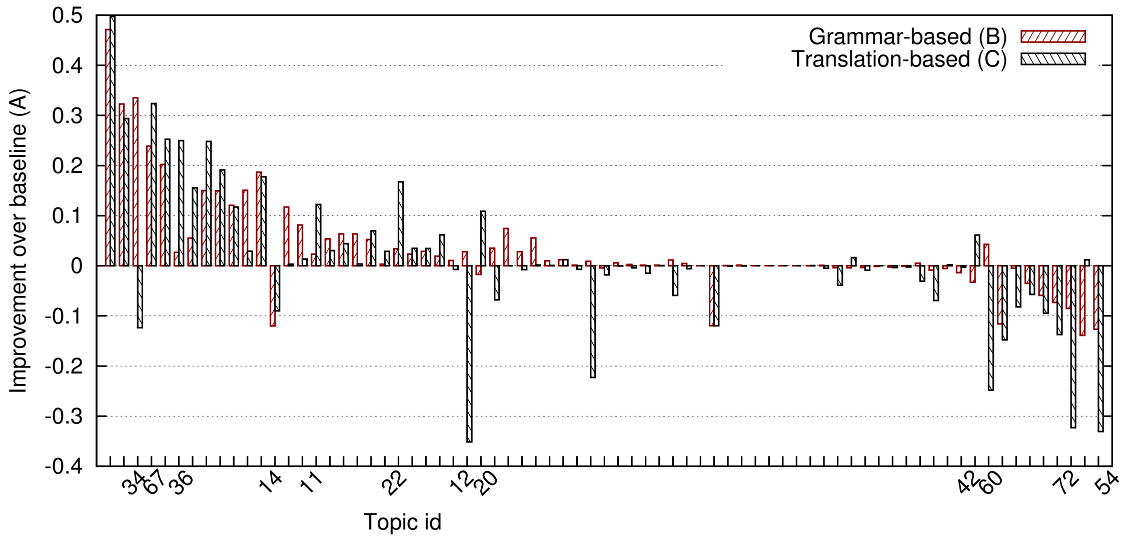


Figure 4.14: Per-topic AP improvement over token-based baseline (condition A) for Chinese: Grammar- and translation-based (i.e., n-best) models.

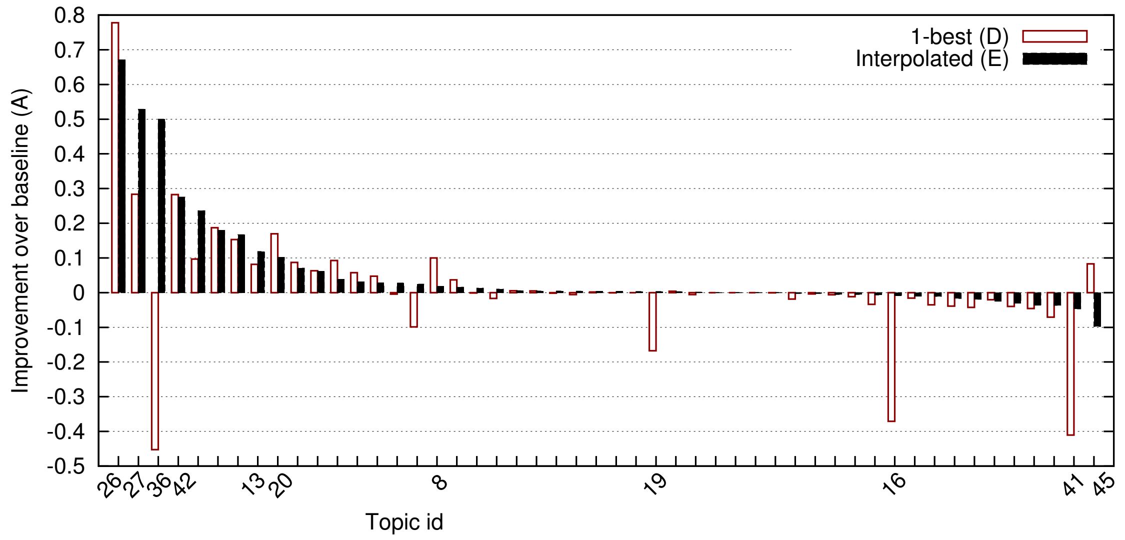


Figure 4.15: Per-topic AP improvement over token-based baseline (condition A) for French: Interpolated and 1-best models.

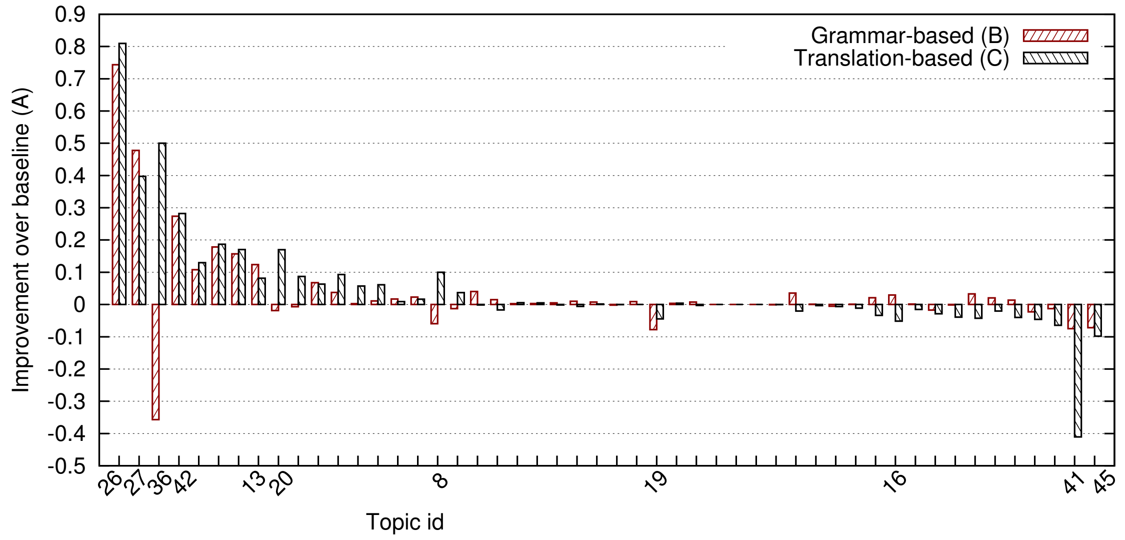


Figure 4.16: Per-topic AP improvement over token-based baseline (condition A) for French: Grammar- and translation-based (i.e., n-best) models.

below) the token-based baseline (A), sorted left to right by decreasing AP improvement for the interpolated model (E). Figures 4.12, 4.14, and 4.16 similarly plot the same AP differences for the grammar-based (B) and n-best full query translation approaches (C), again in reference to the token-based baseline (A), with topics sorted in the same order to facilitate comparison.

These plots make it quite clear that the three approaches vary in their per-topic effectiveness. Rather than slight variations across all of the topics, we see several topics in which one or two of the models is much better than the rest. For instance, for the Arabic collection, the n-best full query translation approach (C) is a clear winner for topics 40 and 41, whereas the same is true for topics 67, 36, 11 and 22 for Chinese, and topics 36, 20 and 8 for French. The grammar-based approach (B) outperforms the rest for topics 7 and 31 in the Arabic collection, 34 and 12 for Chinese, and topics 27 and 13 in French. As expected, the performance of one-best (D) is correlated with the translation-based approach (C), yet there are topics in which the latter outperforms the former (e.g., topics 25 and 31 in the Arabic collection, 25 and 67 in Chinese, and 36 and 16 for French), and vice versa (e.g., topic 20 in Arabic, 60 in Chinese, and 45 in French). All of these topics are marked on the respective figures. Additionally, despite its drawbacks, there are topics in which the token-based model is superior to our more sophisticated approaches, notably topics 16 and 43 for Arabic, topics 54 and 72 for Chinese, and topics 41 and 45 for French. Once again, this analysis supports our argument that combining these three probabilistic models into one unified approach can capture some of the best of each. In general, we expect the interpolated model to be more robust, since it has access to more evidence than the individual models.

We observe that the translation-based model (C) outperforms the rest when the decoder does well in finding appropriate translations. As an example, topic #36 in CLEF 2006 is *NBA labor conflict*, which is translated into the PSQ shown in Figure 4.17.

```
#comb(#weight(1.0 nba)
      #weight(1.0 travail)
      #weight(0.96 confl, 0.04 conflit))
```

Figure 4.17: A PSQ, representing the translation of query "NBA labor conflict" using Pr_{nbest}.

In the grammar-based (B) and token-based (A) approaches, the PSQ contains other translation alternatives, such as *contradiction* instead of *conflit*, or words related to *labor*, such as *social*. In this case, these alternative translations do not benefit retrieval effectiveness, and introduce more noise instead. On the other hand, the 1-best MT approach (D) suffers because the top translation of the query omits the translation of *labor* altogether. The second top scoring translation does not have this issue, therefore this is an excellent example of the benefits of using top n translations instead of only the single best.

Another interesting case in CLEF 2006 is topic #13, *centenary celebrations*, which is translated correctly into French as *centenaire* by the decoder. However, as opposed to MT, the task of CLIR is not only aboardut finding the most appropriate translation, but retrieving and ranking relevant documents. The grammar-based approach (B) includes *célébrations* (Eng. *celebration*) in addition to *centenaire*, which increases the relevance score of some relevant documents, improving the average precision.

Topic #41, *theft of "The Scream"* displays a weakness of relying on language models. In this particular case, the translation candidate *vol du "Cri"* is down-weighted by the language model since it is a sequence of words never seen before. Instead, the decoder picks *vol de "Scream"* as the top translation, which causes lower retrieval effectiveness. When the MT system fails to find an appropriate translation, the token-based model (A) outperforms easily because it does not prematurely discard any alternative, a result of the ambiguity-preserving nature of this model. Any possible translation of the query (that can be found in the training data) is represented in the PSQ, producing a representation with many translation alternatives; some of these might be irrelevant, but it is likely that the correct translation is there as well. The grammar-based model (B) does worse than the token-based (A), but much better than the translation-based (C) for this topic, supporting the argument that it is a compromise between the two.

Parameter Tuning In practice, we would like to select model parameters without having access to the test topics and relevance judgements. Therefore, we ran 10-fold cross-validation experiments on each collection, by selecting the parameters that maximize MAP on nine folds and evaluating on the remaining one. This method yields a MAP of 0.293 for Arabic (same as the best run in Table 4.3), 0.191 for Chinese, and 0.311 for French, all significantly better than the token-based baseline (A) with 95% confidence. In the case of Arabic and Chinese, this cross-validation model is statistically significantly better than the 1-best (D) and 10-best MT (C) approaches as well.

We also explored if we could use two of the collections to tune parameters for the third collection. For this, we first ranked each (λ_1, λ_2) pair by MAP on each collection. In order to select the parameters for a particular collection, we added the ranks from the other two collections and picked the one with the lowest sum. Using this method, the selected parameters were $(0.1, 0.8)$ for Arabic, $(0.3, 0.5)$ for Chinese, and $(0.1, 0.1)$ for French. When compared to the token-based baseline (A), this approach showed significant improvements only for Chinese.

From this analysis, we conclude that the optimal combination of models depends on the collection, language, and resources. Once these are fixed, we can use a subset of the topics to appropriately tune parameters for the rest. However, better tuning methods need to be devised for a truly robust approach to combining these CLIR models.

4.4.3 Efficiency

We compared the various CLIR approaches in terms of efficiency (query evaluation time), performing experiments on a server running Red Hat Linux on a 2.4 GHz processor. We processed the Arabic topics using each model and measured running time per query in milliseconds. Average values over three repeated runs are reported in Table 4.4 (with 95% confidence intervals).

As described before, there are three processes in the MT pipeline: word alignment, translation modeling (i.e., grammar extraction), and decoding. Word alignment is query-independent and required for all three approaches, so we did not include it in our comparison of running times. For the construction of Pr_{SCFG}, we only need to extract grammar rules that apply to each given query,[18] whereas Pr_{nbest} also requires decoding.[19] We discussed the need for MT tuning before, but we do not include tuning cost in this evaluation, since it is a one-time query-independent cost that will become negligible as number of queries increases.

The remaining processes that we need to consider are part of the IR pipeline: initialization of the CLIR model, generation of query representations in the target language, and ranking of the most relevant documents in the collection. We only count query-dependent initialization costs, since other costs such as loading the bilingual dictionary need to be done only once, and therefore are considered negligible. The generation step takes the source-language query as input, and outputs a PSQ that represents that query in the target language. In the grammar-based method, this step takes a negligible amount of time, because the probability distribution is already in memory at the beginning of this step, and it is very small (i.e., probabilities for a few query terms only). For Pr_{nbest}, generation time rises linearly as n is increased.

Ranking time depends on the complexity of the query representation. With more complex representations, it is possible to increase effectiveness, but at the cost of efficiency. Therefore, a desirable CLIR approach would express all the relevant in-

[18]We should also note that extracting rules from the grammar takes relatively less time than well-formed sentences for this particular task because the queries are very short.

[19]It is reasonable to assume that the decoder time to find top n translations is the same as finding the one-best result.

formation and nothing more. The distributions Pr_{token} and Pr_{SCFG} tend to include more translation alternatives per query term, resulting in a more complex representation and longer ranking time. As a result, interpolating all three distributions generates a complex representation as well.

	Process	$Pr_{\textbf{token}}$	$Pr_{\textbf{SCFG}}$	$Pr_{\textbf{nbest}}$			$Pr_{\textbf{c}}$
				1-best	**5-best**	**10-best**	
MT	Extraction	-	7.6				
	Decoding	-	-	134.9			
IR	Initialization	negl.	64.4	negl.			64.4
	Generation	48.1	negl.	5.8	59.5	62.3	49.1
	Ranking	545.6	514.2	97.6	158.8	179.0	602.0
Total time (in ms)		594±22	586±13	246±15	361±28	383±22	858±20

Table 4.4: Average running time per query (in ms) for the TREC 2002 English-Arabic task, shown separately for each process in the MT+CLIR pipeline.

When we look at the total running times in Table 4.4, we observe that the n-best approach is significantly more efficient than the token-based baseline, even though it requires additional MT processes to fully translate the queries. When $n = 1$, the reduction in total running time is nearly 60%. The savings become more modest as n increases, approximately 39% and 35% for 5-best and 10-best MT approaches. Increasing n also improves effectiveness, thus there is a tradeoff to consider when deciding on the value for n. There is a similar tradeoff for the token-based approach: the representation can be simplified if more aggressive thresholding is used (e.g., if C or H increased in Equation 4.1); however, this may result in a less effective model.

We do not see the same efficiency improvements from reduction in query complexity with the grammar-based model; the query complexity is similar to the baseline approach. As a result, the grammar-based approach runs in about the same total time. However, the MAP score improves considerably for all of the collections, so we can say that Pr_{SCFG} is preferable to Pr_{token}.

The combined model Pr_{c} yields the highest MAP scores but also takes the longest time to complete. When compared to the baseline model, running time increases by 44%, which might be acceptable given the consistently significant improvements, but depends on the application. We should also note that our implementation is not fully optimized, and is open to further improvements in the future.

For a better understanding of the tradeoffs between efficiency and effectiveness, Figure 4.18 shows MAP as a function of total running time, for all three CLIR tasks. Instead of the absolute values, we plotted percentage values relative to the baseline. Therefore, negative values indicate a decrease with respect to the baseline, and positive values indicate an increase. As a result, the token-based approach is located on (0,0). The lower right section contains settings in which the running time was higher than the token-based approach, and MAP was lower. On the other hand, the upper left part shows approaches that were both more efficient and effective, when compared to the token-based model. The 1-best and 10-best models appear in this section for the Chinese and French experiments, as well as the grammar-based model for the English-Arabic task. The other two sections represent a compromise between better efficiency and effectiveness.

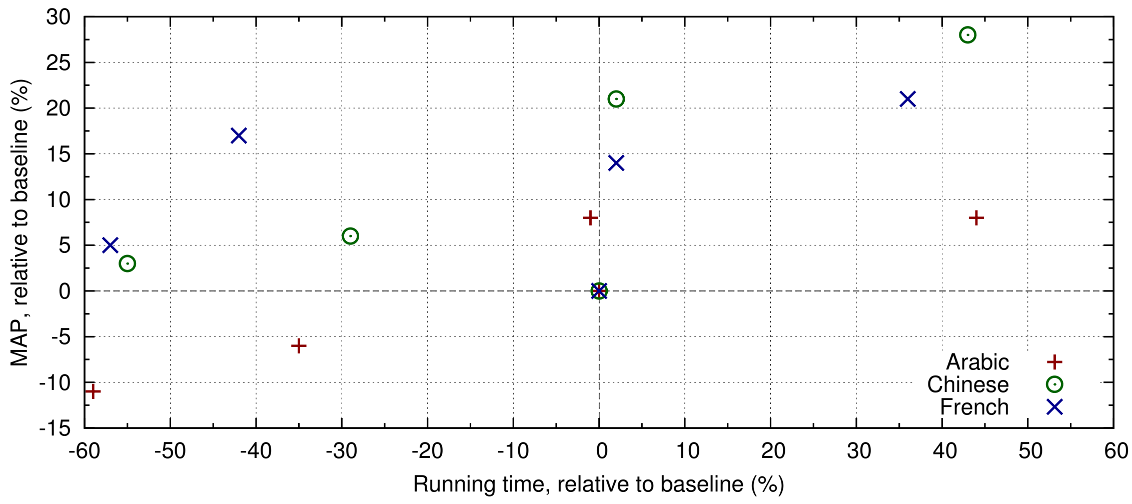


Figure 4.18: An empirical analysis of the efficiency-effectiveness tradeoff for proposed CLIR models.

Efficiency values in Table 4.4 and Figure 4.18 were all computed using a hierarchical MT system (i.e., `cdec`). Although flat MT approaches are considered faster than hierarchical MT, we did not observe much difference in the MT running time when Moses was used instead of `cdec`.[20] Individual timing results were not included in Table 4.4 since it is difficult to make a fair comparison between the processes of `cdec` and Moses, especially due to a particular implementation of the cdec pipeline on our server.

In contrast with the MT side of efficiency results, the IR side changes significantly when a flat grammar is used instead of a hierarchical one If we use the flat grammar of Moses for the grammar-based approach, the generation step takes more than an order of magnitude longer. As discussed before, this is due to the relatively inflated translation models of flat PBMT systems.

[20]As noted before, the decoding time of MT systems is highly dependent on certain system parameters.

As a summary of our evaluation, we believe that the best choice depends on user expectations. For a faster and possibly more effective model, Pr_{nbest} and Pr_{SCFG} seem to be good alternatives to Pr_{token}. For best effectiveness, the interpolation of the three probability distributions is a good choice, providing significantly better results at the cost of additional complexity.

4.5 Conclusions and Future Work

In this chapter, we introduced a framework that uses a statistical MT system for cross-language information retrieval. Our approach combines the representational advantage of probabilistic structured queries with the richness of the intermediate information produced by translation and language models in MT systems. We proposed two ways of exploiting the internal representation of translation models to learn context-sensitive and ambiguity-preserving term translation probabilities: (1) extract the subset of the translation grammar that applies to a given query, and use the word alignments in each rule to construct a probability distribution, or (2) aggregate information from the n-best translation output by an MT decoder. Our implementation covers two of the most widely adapted MT approaches: hierarchical phrase-based and flat phrase-based.

We evaluated our models on an English-Arabic task from TREC 2002, an English-Chinese task from NTCIR-8, and an English-French task from CLEF 2006. In terms of efficiency, we showed that our framework provides a set of choices, allowing a beneficial tradeoff between improving efficiency and effectiveness. In terms of effectiveness, we found in all three cases that an optimal linear combination of the three approaches can significantly improve MAP (regardless of the underlying MT approach), but that the optimal parameters vary by collection. Since we used only one collection per language, experiments with multiple collections for the same language will be needed before we can begin to speculate on whether these differences are language-dependent, collection-dependent, or some combination of the two. Additionally, we would like to try this approach on more languages to further study the consistency in improvements, and also with different parallel corpora and monolingual language modeling collections, in order to decide whether the differences we are seeing in the optimal combination weights are resource-dependent (varying principally with different parallel corpora and/or language models).

In terms of modeling, our experiments with different MT models revealed that the differences become most apparent when the translation model is used directly. Once the decoder is introduced, the improvements due to the more advanced modeling of hierarchical MT shrink substantially. We plan to revisit the rather ad hoc way we have incorporated multi-word expressions, exploring ways of leveraging them in each model separately rather than at the final evidence combination stage. Also, since the benefit of performing MT would be expected to increase as available context increases, we would like to explore the potential for translating *documents* in addition to *queries*. Following the same methods described in this paper, we could learn a new set of probability distributions from the document translations, which could be combined with the current three approaches to construct an even richer and possibly more accurate CLIR model.

In conclusion, we have introduced ways of using statistical translation models for CLIR that take greater advantage of the capabilities of current MT systems, and we hope that our encouraging results will spur the community for further research.

Chapter 5: **Searching to Translating to Search Again: An Architecture for Connecting IR and MT**

5.1 Introduction

In Chapters 3 and 4, we described two approaches, showing how one can use search techniques to improve translation, and how one can use translation techniques to improve search. In this chapter, we present a system architecture that combines these two approaches into a single circular process, with a potential to improve both of the underlying MT and IR models after a full cycle. We also present evidence to support this hypothesis, by implementing a simplified version of the proposed process, which is a bootstrapping approach extending the work in Chapter 3.

5.2 Overview of "Searching to Translate" and "Translating to Search"

Let us first illustrate the approaches in Chapters 3 and 4 in one big picture. In both cases, our starting point is a parallel corpus, containing sentence pairs from the language pair we are focusing on.

Two existing pipelines are used in this work:

1. MT pipeline builds a statistical MT translation model from a parallel corpus

(Section 2.1).

2. CLIR pipeline builds a CLIR translation model from token translation probabilities (Section 4.2), which are learned from a parallel corpus (Section 2.1.1).

Following the terminology above, in Chapter 3, we showed how to use a *CLIR translation model* to apply cross-language translation techniques for extracting parallel text from comparable corpora. As a result, we concluded that using the extracted parallel text for training (in addition to the baseline corpus) can improve the *MT translation model*. Figure 5.1 illustrates this pipeline, where solid/black-colored shapes represent existing components, and dotted/red-colored shapes represent approaches introduced in this dissertation and the resulting improved models. Based on our experiments, this approach yielded BLEU improvements to strong MT baseline systems for five out of six language pairs we tested.

In Chapter 4, we showed how to use the rich internal representation of an *MT translation model* to provide ambiguity-preserving and context-sensitive query translation in the *CLIR translation model*. As a result of improved query translation, experiments indicate statistically significant improvements in MAP, over a context-independent baseline approach, for three different collections (in three different languages). Using the same coloring scheme as Figure 5.1, we illustrate this pipeline in Figure 5.2.

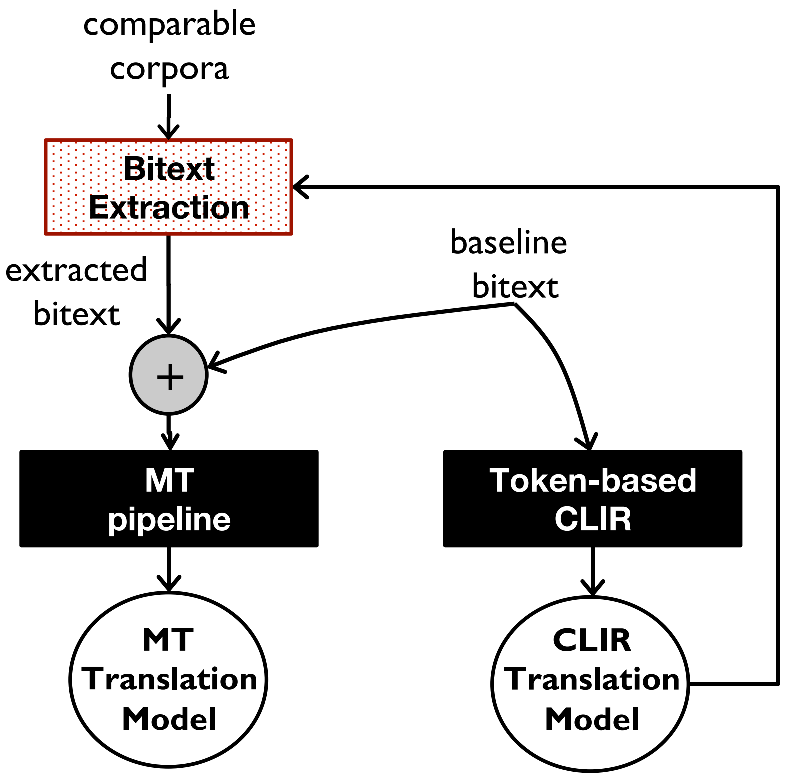


Figure 5.1: Illustration of our approach in Chapter 3.

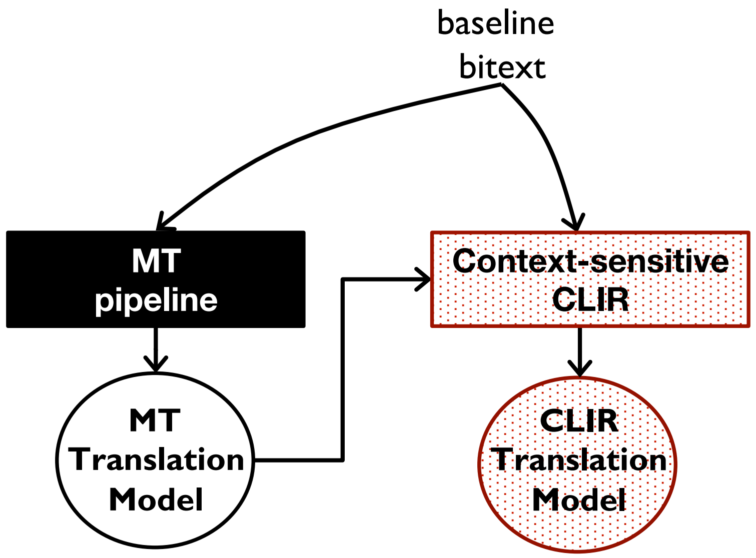


Figure 5.2: Illustration of our approach in Chapter 4.

5.3 Connecting "Searching to Translate" and "Translating to Search" via a Circular Architecture

These two chapters exhibit a nice symmetry; this creates an opportunity to extend the integration of IR and MT, by combining the two into a single architecture, as shown in Figure 5.3. In this proposed architecture, the circular process starts by adapting our context-sensitive CLIR model (Chapter 4) into our parallel text extraction approach (Chapter 3). The extracted sentence pairs can be added to the MT pipeline to build an improved MT translation model. We can use the *improved* MT model in another run of the context-sensitive CLIR, bringing back improvements to the initial CLIR model. This CLIR model can be subsequently applied to the parallel text extraction pipeline to get more and better bitext.

Therefore, the proposed cycle can potentially result in improvements to both IR and MT models, and can be repeated until some convergence is reached (i.e., until we reach the limit of information we can extract from the comparable corpus, as discussed in Chapter 3). Different variations of this idea can be implemented in practice. We present experimental results from a simplified implementation, in order to show the feasibility of this idea.

5.4 A Bootstrapping Approach for Bitext Extraction

Our simplified implementation is a bootstrapping approach, as illustrated in Figure 5.4. Since this is an iterative process, we used solid, dashed and dotted lines

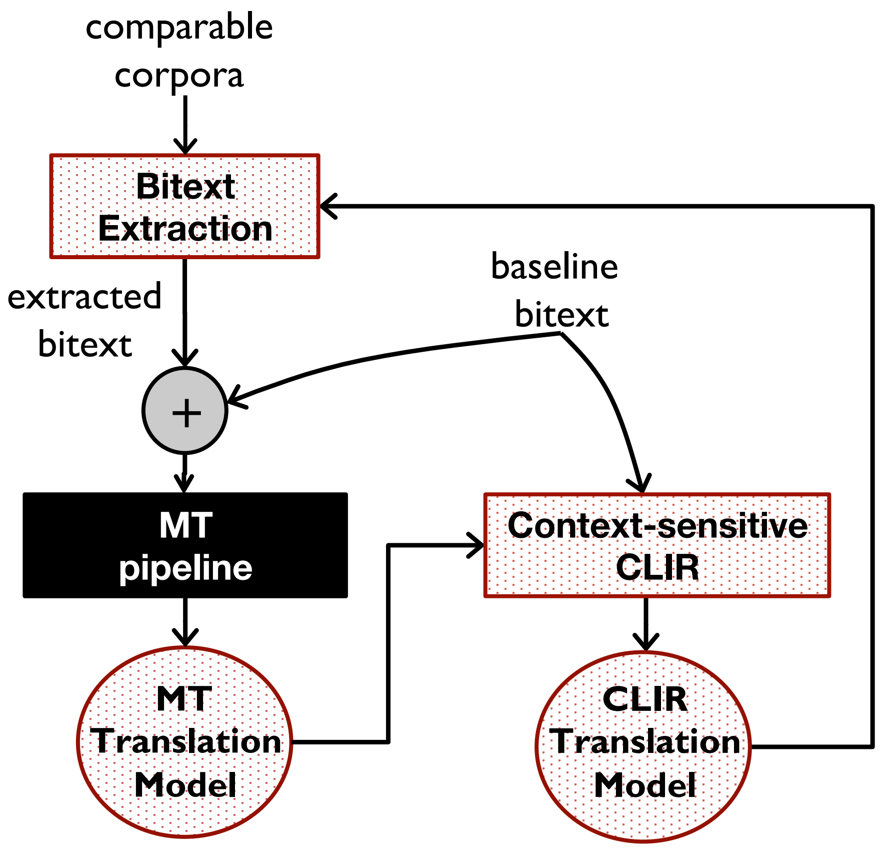


Figure 5.3: Proposed approach to unify approaches in two chapters, which improves both IR and MT models in an iterative process.

to indicate the execution order: Starting from the baseline bitext, solid lines show the process to extract bitext as described in Chapter 3, referred to as D_1. Starting from D_1, following dashed lines shows how we piped D_1 back into the CLIR pipeline, instead of using it to improve MT. As a result of adding D_1 to the baseline corpus, we obtained a modified CLIR model with an expanded vocabulary and updated translation probabilities.

In order to quantify the effect of D_1 on the CLIR models, Table 5.1 displays statistics before and after the bitext is included, for the same six language pairs evaluated in Chapter 3: German-English (de-en), Spanish-English (es-en), Chinese-English (zh-en), Arabic-English (ar-en), Czech-English (cs-en), and Turkish-English (tr-en). In each row, columns show the vocabulary size (i.e., number of distinct words) for source and target languages, respectively, and the translation table size (i.e., number of distinct translation pairings). Note that these parameters depend on the training corpus size and content, as well as the language pair. For example, the Chinese-English training corpus is the largest and contains text from forums, resulting in much larger vocabularies. The Chinese vocabulary is especially large due to the characteristics of the language.

When D_1 is added, the increase in these statistics ranges from 2% for Turkish-English, up to over 400% for Spanish-English. This increase cannot be directly interpreted as an improvement, since the additional words and translation pairings might have two conflicting effects: (a) providing coverage of previously unknown words and (b) increasing noise by adding gibberish words to the vocabulary. One way to compare the two CLIR models is to directly evaluate both on the task of

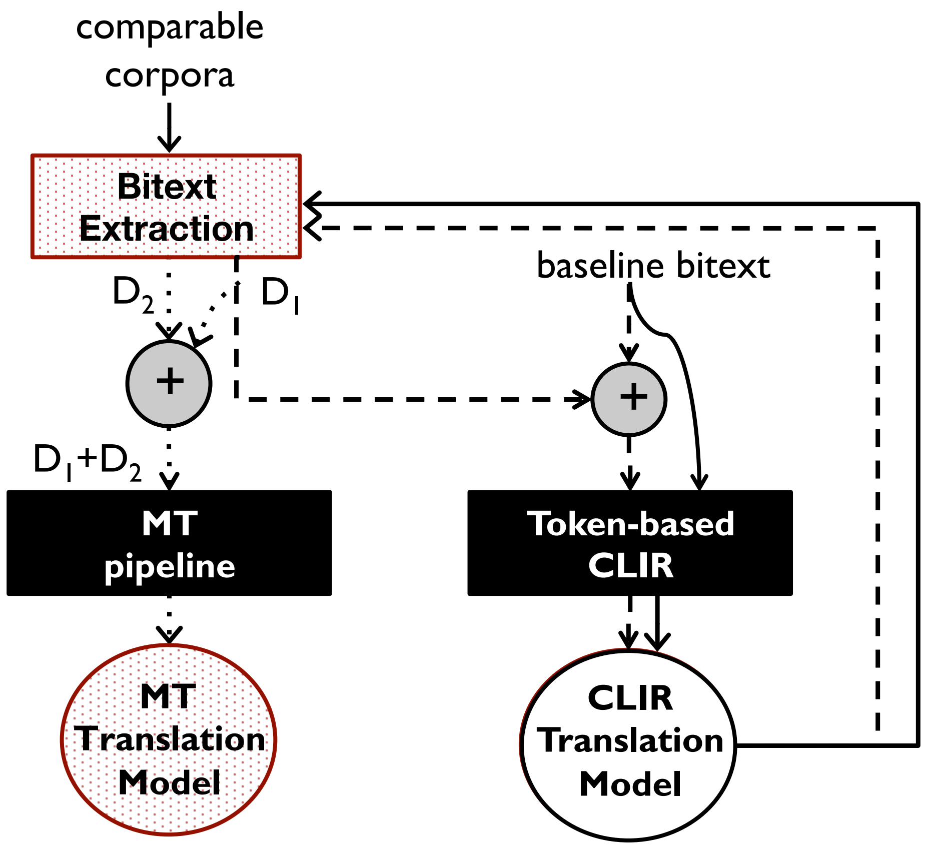


Figure 5.4: Simplified version of the proposed approach, where we bootstrap the improvements from extracted bitext D_1 into the CLIR pipeline and obtain a better translation model (shown by dashed line), producing bitext D_2, which is used for MT training (shown by dotted line).

cross-language retrieval. We could not pursue this direction since we do not possess labeled data for all language pairs. Instead, we decided to perform an extrinsic evaluation on MT.

The dotted lines show how we used the updated CLIR translation model to find more sentence pairs from the comparable corpus. We applied the new CLIR model to the our bitext extraction pipeline, potentially finding sentence pairs not extracted in the first iteration due to vocabulary limitations. We added this new parallel text (called D_2) to D_1 and the baseline corpus, and trained a new MT model from this combined data.

Language	**Baseline**			**Baseline+D_1**		
	Vocabulary		Translation Table	Vocabulary		Translation Table
	Source	Target		Source	Target	
de-en	271.2	80.2	1098.4	695.1	352.8	2378.7
es-en	72.8	68.6	338.1	573.6	442.2	1760.7
zh-en	1991.6	700.7	6297.7	2064.3	715.5	6515.7
ar-en	219.2	134.0	1088.1	286.9	176.1	1299.6
cs-en	104.2	42.4	532.8	284.6	153.7	1288.4
tr-en	27.9	15.3	125.0	28.9	15.8	127.7

Table 5.1: Size of the CLIR model (in thousands): before and after first batch of parallel data extracted from Wikipedia (D_1) is included.

With the additional data, the updated MT system produces higher-quality translations (in terms of BLEU on the same test set) than by just adding D_1. Improvements in each iteration, for each language pair, are reported in Table 5.2. Although the increase in the second iteration is less substantial than the first, the results are promising for future exploration of such circular approaches within this proposed architecture.

We observe that the bootstrapping is more effective for language pairs with limited parallel text to start with. This outcome supports our hypothesis that Turkish-English and Czech-English systems suffered from weak initial resources (i.e., translation model and vocabulary). Therefore, running this bootstrapping cycle within the architecture might be especially useful for low-resource language pairs.

Although the second iteration improves BLEU for all language pairs, the additional improvements are not as substantial as the first iteration. One possible

Language	**Baseline**	$+D_1$	$+D_2$
German-English	24.50	25.00	25.08
Spanish-English	33.44	35.15	35.30
Chinese-English	27.51	28.01	28.11
Arabic-English	63.15	66.66	66.67
Czech-English	23.11	23.54	23.72
Turkish-English	27.22	27.58	27.79

Table 5.2: BLEU score after first and second iterations of bitext extraction.

reason for this is the fact that the test sentences are from the same domain as the initial parallel corpus. This is an important factor, since translation models are highly tuned to the domain from which they are trained. While obtaining more data from a different domain might help for some out-of-vocabulary terms, it is also likely to hurt by biasing the translation model into less appropriate translations. For example, the phrase "developing countries" has a single specific sense in the news domain, but we might find many other uses of these two words in Wikipedia. Therefore, including these sentence pairs might introduce noise to the translation model. Evaluating this bootstrapping approach on test sets from other domains would determine the effect of this domain mis-match.

5.5 Conclusions and Future Work

In this chapter, we presented a system architecture with MT and CLIR components. We showed that the approaches presented in Chapters 3 and 4 represent different paths within this architecture. Finally, we introduced a bootstrapping approach to increase the benefits of bitext extraction, by performing an additional iteration after expanding the underlying CLIR model.

The bootstrapping method did not yield substantial improvements in the limited number of experiments we conducted. However, it serves as an example to the many new directions one can explore through this architecture. We proposed creating a circular flow by combining multiple paths in the architecture, and there might also be value in adding new paths and connections into the picture. For instance, feedback from the MT model can be sent back to the bitext extraction pipeline, in order to provide a better learning process. Both phases of the bitext extraction approach might benefit from such feedback, since it would provide information on what would add to the existing MT model, as opposed to just finding good translation pairs. One aspect of this would be to reduce redundancy, since the MT model could provide what it already “knows”, and prevent the classifier from emitting redundant pairs.

In conclusion, the main purpose of introducing this architecture was to present new possibilities once various components, techniques, pipelines from MT and IR are put together. We also demonstrated that the two approaches described in previous chapters fit perfectly within this architecture. We hope this will serve as a basis for future research paths, and guide researchers to find other ways to integrate search and translation technologies.

Chapter 6: **Conclusions and Future Work**

6.1 Summary

This dissertation explores solutions to two tasks, search and translation, with a particular focus on how these two technologies can be better integrated. There are two sides to this integration: (1) exploring how translation models can be improved by exploiting the capabilities of current search approaches, and (2) exploring how search processes can be aided by state-of-the-art translation models and techniques. Corresponding to these two sides, we presented solutions to two problems in this dissertation, namely the problems of *searching to translate* and *translating to search*.

Searching to Translate In Chapter 3, we explored how approximate search techniques can be used to efficiently find parallel text within large semi-comparable collections. We presented a pipeline to extract parallel text, which is a crucial resource for training statistical MT systems, from a semi-comparable corpus, which can be found in multi-lingual collections such as Wikipedia. In the first phase, we adapted locality-sensitive hashing techniques to efficiently identify pairs of documents in different languages that have very similar content. Since translating all queries using an MT system is computationally infeasible (given our resources), we performed vector translation based on word-based CLIR techniques.

The second phase operates on the output of the first, and generates candidate sentence pairs within the similar document pairs. Labeling each candidate sentence pair as "parallel" or not involves a very large number of decisions, so we introduced a 2-step classification approach for balancing efficiency and effectiveness.

The entire pipeline is implemented in the MapReduce programming model for increased scalability and parallelization [33]. Experimental results indicate that the parallel text we extract for a diverse set of language pairs is beneficial to MT.

Translating to Search In Chapter 4, we proposed techniques to incorporate various parts of a modern statistical translation model into the cross-language search process. Query and/or document translation is the core of this task, and little has been done previously to exploit the advanced MT approaches for CLIR. We presented a set of methods to construct a term translation probability distribution from different internal representations of a modern statistical MT pipeline.

We compared these approaches to existing methods in the CLIR literature: context-independent token-to-token translations and using MT as a black box for one-best translation. Based on a thorough evaluation on three cross-language retrieval tasks, we found that an interpolation between our proposed models and existing approaches yields best results. We also concluded that optimizing the interpolation weights is a great challenge, yet very important for achieving improved results.

6.2 Limitations

There are several limitations of the methods presented in this dissertation, as listed below:

Resources and Languages As mentioned in Chapter 5, both of our approaches assume the availability of an initial bilingual sentence-aligned corpus. In Chapter 3, experimental results indicated that quality of this initial bitext is very important for a successful application of our bitext extraction approach. This is a limitation since the amount of such resources varies significantly among language pairs.

Since related work suffered from the same drawback, researchers have explored solutions for low-resource language pairs. The bootstrapping approach we introduced in Chapter 5 is one such example, although our experiments did not indicate substantial benefits. Another very recent paper discusses a crowd-sourcing approach without relying on any initial data, which might alleviate this issue [130].

We have shown that our approach can successfully extract useful parallel text from semi-comparable text. While this is an improvement over previous work, our approach may not apply to arbitrary multi-lingual collections. One of our future goals is to scale to larger web collections, yet it is unclear whether the current approach will succeed when documents are even less similar. A deeper analysis of this matter is required before we can be confident that bilingual text can be extracted from arbitrary multi-lingual web collections using our approach in Chapter 3.

Although we tried to include a diverse set of languages in this dissertation, we could only cover a very small portion of world languages. Therefore, applying our methods to other languages requires additional modifications to Ivory, the open-source project that hosts our implementation. In addition to changes to the code, adding a new language to our pipeline also requires a parallel corpus for training the initial models (for both problems), and a comparable corpus that should at least satisfy the properties of a semi-comparable corpus (only for bitext extraction).

Wikipedia has served as a perfect collection for our experiments due to free access and a wide coverage of languages, however these properties might not hold for other resources. For example, we are not aware of any general web collection for Turkish, which limits the applicability of our approach to extract Turkish-English parallel text from the web. A related issue with our current implementation is the assumption that the target language is English. We should emphasize that this is not a limitation our *methods*, which can be applied to any language pair, given the resources.

Finally, some of our conclusions in Chapter 3 are based on the availability of a Hadoop cluster.

Parameter Estimation A major drawback of our approaches is the dependence on many parameters, which need to be carefully tuned for optimal performance. This limitation applies to several parts of our work.

In the first phase of our bitext extraction pipeline, we need to tune many parameters of the LSH-based sliding window algorithm. Our evaluation stressed the importance of parameter selection in analyzing the various tradeoffs. To address this issue, we presented an analytical model, relating how parameters influence the efficiency and effectiveness of the task. While this analytical model overcomes this limitation for the first phase, we still suffer from parameter estimation in the second phase. The MT evaluation revealed that finding the right thresholds for the classifiers is essential. A threshold too high might cause the algorithm to miss many good sentence pairs, whereas a threshold too low might cause more noise in the output. Therefore, finding a way to estimate appropriate thresholds without running experiments is essential to a fully operational bitext extraction pipeline.

In the context-sensitive CLIR framework, we discussed the importance of the weights of our combination-of-evidence approach. From experimental results, we concluded that the best parameters varied significantly among the three collections we tested. We showed that these parameters could be estimated properly if we had a comparable set of queries and relevance judgments to start with. We also showed that parameter estimation was not possible from other collections (i.e., tuning parameters on the Chinese collection and applying on Arabic was not successful). Therefore, our approach cannot guarantee the significant improvements without a more robust way of optimizing parameters. An initial step for solving this problem might be to understand the problem better by experimenting with a wider variety of collection types and languages.

Extracting Machine-Translated Text Our bitext classification approach cannot distinguish between human and machine translation. Since the web contains a considerable amount of machine-translated text, this might cause the algorithm to learn from low-quality translations. A paper by Google presents a solution to this: the authors "watermark" translations of their own MT system (i.e., Google Translate), to make sure these do not influence their efforts to extract new parallel text [142]. We can argue that our output does not include machine-translated text due to the edited content of Wikipedia. Wikipedia has editors that strictly enforce high quality throughout the collection, and machine-translated text is either fixed or marked as poor quality. Therefore, the parallel corpora we extracted from Wikipedia should not contain a noticeable amount of machine-translated text, if any.

Context Window In this dissertation, we assume fixed context windows for operation: the first phase of the bitext extraction pipeline finds *document* pairs and the second phase identifies parallel text at the *sentence* level. However, this limits the capabilities of our approach, as we are discarding potentially useful information at any level in between these two pre-defined windows. For instance, there might be two documents very different, but each of them might have a very similar paragraph about a certain subject. Our current implementation might miss these cases; however, our flexible code interface would allow modifications for redefining the level of context.

6.3 Future Work

In the rest of this chapter, we present a few directions that might be interesting to pursue in the future, complementing the work described throughout the dissertation.

6.3.1 Extracting Parallel Text from the Web

Extracting bitext automatically from the web is one of the most promising approaches to improve the state of the art of statistical MT systems. As we have shown in Chapter 3, this idea can improve MT models by extracting high-quality training data continuously, as more multi-lingual text is added to the web.

In order to strengthen this argument even further, we would like to extend our approach to many more language pairs, and to collections much larger than Wikipedia. The ClueWeb dataset,[1] containing 1 billion web pages from 10 different languages, might be a good starting point.

The approach we have implemented shows significant gains in translation quality, however it can be improved further in several ways. There are many other features that can be added to the bitext classifier, such as syntactic relations, fluency of sentences, and the use of punctuation, named-entities, multi-token phrases and more. Also, we do little to clean up the noisy data, which may cause many sentence pairs to contain HTML code and other mangled characters. Further refinements should be made to address these issues.

Another promising direction for improving our approach is to generalize the two-step classifier into a cascade of many classifiers, possibly adding some of the features suggested above. Instead of determining thresholds for these classifiers in an ad-hoc manner as before, we can turn it into a learning problem, where the objective is to minimize the cascade classification error for a labeled set of sentence pairs. In

[1]http://www.lemurproject.org/clueweb09.php

addition, as seen from the domain-specific evaluation in Table 3.6, obtaining labeled pairs in a wider range of domains is critical for the model to generalize well. Hence, approaches to collect such data (e.g., crowd-sourcing) might be worth exploring.

As discussed in Section 6.2, a drawback of the current approach is the lack of a mechanism to determine the optimal classifier thresholds. We can consider this as the following problem: In the second step of classification, the classifier assigns a score to each candidate sentence pair. Instead of fixing a threshold on this score, we can do the following: Starting from the highest scored pair, we iterate through the candidates until we know there is no additional gain from adding more. Predicting this stopping point would be very valuable in reducing the effort on tuning parameters of our approach. Also, since our evaluation in Section 3.4.1 reveals that there is a optimal amount of parallel text to be extracted from any collection (see Figures 3.17 and 3.18), knowing when the BLEU is going to peak would eliminate the need to perform many time-consuming MT experiments.

6.3.2 Context in Natural Language Processing

In natural language processing (NLP) problems, it is often that we are interested in making a (probabilistic) decision (e.g., determine part-of-speech tag, find possible translations, etc.) on some part of the text (say, a word), given any knowledge available to our model. Ideally, we would like to build models that can use all available world knowledge; however, this is not possible in practice, and we need to reduce the computational complexity by constraining our model. This is usually achieved by limiting the amount of local and global context available when processing a word (or any other part of the text). Therefore, it is an open problem to understand how to correctly adjust the amount of context, when processing language at various windows of text.

Although not the primary focus of this dissertation, we provide insights on this aspect in both chapters. In Chapter 3, we compared two alternative approaches for cross-lingual pairwise similarity, and concluded that limiting the context of the translation model to the word-level (by using CLIR techniques as opposed to MT) is comparable in effectiveness, yet orders of magnitude better in efficiency. These results have strengthened the hypothesis that statistical translation models perform well in retrieval tasks, even with strong independence assumptions. On the contrary, in Chapter 4, we showed that term translation probabilities conditioned on wider context (i.e., query) perform significantly better than context-independent probabilities for cross-language IR, with little or no additional complexity.

As part of future plans, it might be interesting to explore different levels of context in other NLP problems. For example, apart from the work described in this dissertation, we explored using document-level context in MT. We concluded that encouraging consistency at the document-level provides better translations [139]. These "consistency features" introduced in our approach were based on tf and idf weights, showing once more that putting together ideas from IR and MT is useful. This has opened new avenues in MT research, with an increase of focus on exploiting context beyond the sentence-level. One such extension is to group documents to identify topics or genres inherent in text, and provide a wider context when translating phrases. Both monolingual and cross-language IR techniques should be useful in finding relevant documents for such applications.

6.3.3 Domain Adaptation in MT

When translating text, it is important to build models specifically for the domain, since the vocabulary and translation choices differ based on the domain. This so-called “domain adaptation” problem has received a fair amount of attention in MT research literature, in which adaptation methods have been applied to both language models [156, 88, 67, 126] and translation models [12, 60, 120, 136, 124], and has been shown to improve results in many cases.

Since the main focus of this dissertation is towards the translation model (TM) of an MT system, a natural extension is to explore approaches for adapting the TM to new domains. Previous work on domain adaptation for the TM can be separated into two threads: On one hand, there are approaches that select passages from the existing training data that are most relevant to the input text, and use those to adapt the TM. The gains are limited because the scope from which new translation rules can be learned is limited to existing parallel corpora.

On the other hand, some approaches exploit the vast amounts of available monolingual text (i.e., in either source or target language) for domain adaptation (e.g. Snover et al. [132]). Given an input sentence, relevant passages are retrieved from the monolingual (source or target language) data source, and translated into the corresponding language (with an MT system). Although this expands the scope of possible resources to exploit (e.g., one could simply find relevant data from the web and translate it), using the un-adapted MT system to translate the adaptation text may result in poor translations. Furthermore, this self-learning process may cause the system to reinforce previous errors.

We can use our pairwise similarity pipeline to combine the best of these two domain adaptation approaches. Given an input sentence for translation, we can find parallel text for TM adaptation as follows: Assuming there exists an in-domain comparable corpus (e.g., Wikipedia), we run our approach from Chapter 3 to find cross-lingual document pairs. Then, we query the input sentence to retrieve the most relevant subset of these document pairs. For this retrieval task, we can run either monolingual or cross-language IR, or we can run both and combine the relevance scores. Once relevant document pairs are retrieved, we run the bitext classifier to select a set of bilingual sentence pairs within these documents. This approach yields an in-domain parallel corpus that is relevant to the input sentence.

For domain adaptation, we can simply add this in-domain bitext to the baseline training data, and measure improvements. However, a more direct adaptation would involve introducing a special feature into the MT log-linear model, similar to the consistency features in Section 6.3.2. By doing this, we can let the translation model learn how much weight to put on the in-domain text.

6.3.4 Representing Phrases in CLIR

In Section 4, we focused on the translation of single-token terms in the source query. On the target side, we also experimented with multi-token expressions, or "phrases," through the one-to-many heuristic, and this method outperformed the others in most cases. Representing phrases on the source side might be beneficial as well: we can represent the fact that *maternity leave* translates into *congé de maternité*, so that the retrieval model should prefer documents in which tokens *congé*, *de*, and *maternité* appear consecutively. Notice that this is different than having these three tokens separately in the translation, since the latter does not distinguish documents that contain *congé*, *de*, and *maternité* separately from documents with the phrase *congé de maternité*.

We used the n-best translation list to explore how source phrases are translated in context. The right hand side of the rules in the n most probable derivations provides us with statistically meaningful target-language phrases (aligned to source-side phrases), along with their associated probabilities.

Within our CLIR framework, it is not clear how to incorporate such phrases into the structured query representation (i.e., Pr_{c} in Chapter 4). We experimented with a rather ad-hoc approach as follows: We first generate a separate probability distribution (called Pr_{multi}), representing the probabilities of phrasal translations, obtained from the n-best list (as described above). We then score each document by a weighted average of a score based on Pr_{c} and this alternative model based on multi-token terms, Pr_{multi}:

$$\begin{aligned} \text{Score}(d|s;\gamma,\lambda_1,\lambda_2) \ &= \ \gamma\ \text{Score}_{\text{c}}(d|s;\lambda_1,\lambda_2) \qquad (6.1)\\ &+(1-\gamma)\sum_{\text{phrase } p} \text{BM25}(\text{tf}(p,d),\text{df}(p))Pr_{\text{multi}}(p) \\ Pr_{\text{multi}}(p) &= \frac{1}{\psi}\sum_{k=1}^{n}\sum_{\substack{\text{rule } r\in D^{(k)}\\ p\in RHS(r)}} \ell(r) \qquad (6.2) \end{aligned}$$

where ψ is the normalization factor and $D^{(k)}$ is the derivation of the k^{th} best translation. Figure 6.1 illustrates the representation of the example query, *Maternal leave in Europe*, under this model, with γ set to 0.8.

```
#combweight(
       0.8 #comb(#weight(0.81 matern, 0.12 maternel, ... )
                 #weight(0.45 cong, 0.25 laiss, 0.10 quitt, ... )
                 #weight(0.95 europ, 0.04 européen, ... ))
       0.1 “en europ”, 0.08 “cong de”, 0.01 “cong matern”, ... )
```

Figure 6.1: Probabilistic representation of the translation of query “Maternal leave in Europe” using 80% weight from Pr_{c} and 20% weight from Pr_{multi}.

The `#combweight` operator corresponds to the weighted averaging in Equation 6.1. The `#comb` structure represents Pr_{c} and the remaining multi-token terms represent Pr_{multi}, all extracted from the top n derivations. In the example, notice that 20% weight is shared among all multi-token terms that appear on the RHS of the derivation rules, each weighted by the normalized translation likelihood, Pr_{multi}.

As an evaluation of this approach, we repeated the CLIR experiments, using the updated relevance model in Equation 6.1. After trying a range of values between 0 and 1 for γ, we empirically found that 0.8 worked best. This change yielded an insignificant 0.005 MAP increase for French, and no change for the Arabic and Chinese collections. As a result, we concluded that introducing phrasal translations this way does not really benefit retrieval. Hence, more principled approaches should be explored in the future, potentially resulting in even better models for CLIR.

6.3.5 Combining Document and Query Translations For CLIR

The translation techniques described in Chapter 4 can be adapted to document translation in CLIR, as well as query translation. In the future, it might be interesting to compare document and query translation approaches within the CLIR framework discussed in this dissertation. Furthermore, it might be beneficial to combine the translation probabilities learned from translating documents into query language, and vice versa. Once we construct a term translation probability distribution from each translation direction, we can merge them into bidirectional translation probabilities, following the work of Wang and Oard [145].

Of course, the main benefit of document translation over query translation is the ability to generate better translation choices by using the additional context available in the document. However, current MT systems perform translation at the sentence-level, ignoring any useful information that may be available throughout the containing document. According to Nie, this is one of the main reasons document translation has not received as much attention as query translations in CLIR [104]. This provides a perfect situation for applying our document-level MT approach (Section 6.3.2) for CLIR, another promising direction worth exploring in the future.

Bibliography

[1] Sisay Fissaha Adafre and Maarten de Rijke. Finding similar sentences across multiple languages in Wikipedia. In *Proceedings of the EACL Workshop on New Text: Wikis and Blogs and Other Dynamic Text Sources*, EACL'06, pages 62–69, 2006.

[2] Mirna Adriani and C. J. Van Rijsbergen. Phrase identification in Cross-Language Information Retrieval. In *Proceedings of Content-Based Multimedia Information Access*, RIAO'00, 2000.

[3] Maik Anderka, Benno Stein, and Martin Potthast. Cross-language high similarity search. In *Proceedings of ECIR*, ECIR'10, pages 640–643, 2010.

[4] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM (CACM)*, 51(1):117–122, 2008.

[5] A. T. Arampatzis, T. Tsoris, C. H. A. Koster, and P. Van Der Weide. Phrase-based Information Retrieval. *Information Processing & Management*, 34(6):693–707, 1998.

[6] Bahman Bahmani, Ashish Goel, and Rajendra Shinde. Efficient distributed Locality Sensitive Hashing. In *Proceedings of CIKM*, CIKM'12, pages 2174–2178, 2012. ACM.

[7] Lisa Ballesteros and W. Bruce Croft. Phrasal translation and query expansion techniques for cross-language Information Retrieval. In *Proceedings of SIGIR*, SIGIR'97, 1997.

[8] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of ACL*, ACL'01, pages 26–33, 2001. Association for Computational Linguistics.

[9] Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *Proceedings of WWW*, WWW'07, pages 131–140, 2007. ACM.

[10] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *Proceedings of SIGIR*, SIGIR'99, pages 222–229, 1999. ACM.

[11] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March 1996.

[12] Nicola Bertoldi and Marcello Federico. Domain adaptation for statistical Machine Translation with monolingual resources. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT'09, pages 182–189, 2009. Association for Computational Linguistics.

[13] Rohit G. Bharadwaj and Vasudeva Varma. Language independent identification of parallel sentences using Wikipedia. In *Proceedings of WWW*, WWW'11, pages 11–12, 2011. ACM.

[14] Jeff A. Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of NAACL*, NAACL'03, pages 4–6, 2003. Association for Computational Linguistics.

[15] Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences (SEQUENCES)*, SEQUENCES'97, pages 21–29. IEEE Computer Society, 1997.

[16] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to Machine Translation. *Computational Linguistics (CL)*, 16(2):79–85, June 1990.

[17] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical Machine Translation: parameter estimation. *Computational Linguistics (CL)*, 19(2):263–311, June 1993.

[18] Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 Workshop on statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT'12, pages 10–51, 2012. Association for Computational Linguistics.

[19] Chris Callison-Burch and Miles Osborne. Bootstrapping parallel corpora. In *Proceedings of the NAACL Workshop on Building and using parallel texts*, PARALLEL'03, pages 44–49, 2003. Association for Computational Linguistics.

[20] Guihong Cao, Jian-Yun Nie, and Jing Bai. Constructing better document and query models with markov chains. In *Proceedings of CIKM*, CIKM'06, pages 800–801, 2006. ACM.

[21] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of STOC*, STOC'02, 2002.

[22] Surajit Chaudhuri, Venkatesh Ganti, and Raghav Kaushik. A primitive operator for similarity joins in data cleaning. In *Proceedings of ICDE*, ICDE'06, pages 5–16, Washington, DC, 2006. IEEE Computer Society.

[23] Aitao Chen. Phrasal translation for english-chinese cross language information retrieval. In *Proceedings of the International Conference on Chinese Language Computing*, SIGIR'00, 2000.

[24] David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, ACL '05, 2005.

[25] David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33:201–228, 2007.

[26] Steve Chien and Nicole Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of WWW*, WWW'05, pages 2–11, 2005. ACM.

[27] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary McCabe. Collection statistics for fast duplicate document detection. *ACM Transactions on Information Systems (TOIS)*, 20(2):171–191, 2002.

[28] Ondrej Chum, James Philbin, and Andrew Zisserman. Near Duplicate Image Detection: min-Hash and tf-idf Weighting. In *British Machine Vision Conference*, 2008.

[29] Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. Better hypothesis testing for statistical Machine Translation: controlling for optimizer instability. In *Proceedings of ACL: Human Language Technologies*, ACL-HLT'11, pages 176–181, 2011. Association for Computational Linguistics.

[30] Josep M. Crego and François Yvon. Factored bilingual n-gram language models for statistical machine translation. *Machine Translation*, 24(2):159–175, June 2010.

[31] Kareem Darwish and Douglas Oard. Probabilistic structured query methods. In *Proceedings of SIGIR*, SIGIR'03, 2003.

[32] Gerard de Melo and Gerhard Weikum. Untangling the cross-lingual link structure of Wikipedia. In *Proceedings of ACL*, ACL'10, 2010.

[33] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proceedings of OSDI*, 2004.

[34] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society (Series B)*, 39(1):1–38, 1977.

[35] Yonggang Deng. *Bitext alignment for statistical Machine Translation.* PhD thesis, Johns Hopkins University, Baltimore, MD, 2006. AAI3213688.

[36] Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. cdec: a decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL System Demonstrations*, ACL-Demos'10, pages 7–12, 2010. Association for Computational Linguistics.

[37] Christopher James Dyer. *A formal model of ambiguity and its applications in machine translation.* PhD thesis, University of Maryland at College Park, College Park, MD, 2010. AAI3443442.

[38] Vladimir Eidelman, Chris Dyer, and Philip Resnik. The University of Maryland statistical Machine Translation system for the fifth Workshop on Machine Translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics*, WMT'10, pages 72–76, 2010. Association for Computational Linguistics.

[39] Tamer Elsayed, Jimmy Lin, and Douglas Oard. Pairwise document similarity in large collections with MapReduce. In *Proceedings of NAACL: Human Language Technologies*, NAACL-HLT'08, 2008. Association for Computational Linguistics.

[40] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of SIGMOD*, SIGMOD'03, pages 301–312, 2003. ACM.

[41] Marcello Federico and Nicola Bertoldi. Statistical Cross-Language Information Retrieval using n-best query translations. In *Proceedings of SIGIR*, SIGIR'02, pages 167–174, 2002. ACM.

[42] Message P Forum. Mpi: A message-passing interface standard. Technical report, Knoxville, TN, 1994.

[43] Alexander Fraser, Jinxi Xu, and Ralph Weischedel. TREC 2002 cross-lingual retrieval at BBN. In *Proceedings of TREC-11*, 2002.

[44] Norbert Fuhr. Probabilistic models in Information Retrieval. *The Computer Journal*, 35:243–255, 1992.

[45] Pascale Fung, Pascale Fung, and Percy Cheung. Multilevel bootstrapping for extracting parallel sentences from a quasi-comparable corpus. In *Proceedings of COLING*, COLING'04, pages 1051–1057, 2004.

[46] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of SIGIR*, SIGIR'88, pages 465–480, 1988. ACM.

[47] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, June 1998.

[48] William A. Gale, Kenneth W. Church, and David Yarowsky. One sense per discourse. In *Proceedings of the HLT Workshop on Speech and Natural Language*, HLT'91, pages 233–237, 1992.

[49] Jianfeng Gao, Jian-Yun Nie, Guangyuan Wu, and Guihong Cao. Dependence language model for Information Retrieval. In *Proceedings of SIGIR*, SIGIR'04, pages 170–177, 2004. ACM.

[50] Jianfeng Gao, Jian-Yun Nie, Endong Xun, Jian Zhang, Ming Zhou, and Changning Huang. Improving query translation for Cross-Language Information Retrieval using statistical models. In *Proceedings of SIGIR*, SIGIR'01, pages 96–104, 2001. ACM.

[51] Jianfeng Gao, Shasha Xie, Xiaodong He, and Alnur Ali. Learning lexicon models from search logs for query expansion. In *Proceedings of EMNLP*, EMNLP'12, pages 666–676, 2012.

[52] Sanjay Ghemawat, Howard Gobioff, and Shun Leung. The google file system. *ACM SIGOPS Operating Systems Review*, 37(5):29–43, 2003.

[53] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of VLDB*, VLDB'99, pages 518–529, San Francisco, CA, 1999. Morgan Kaufmann Publishers Inc.

[54] Michel Goemans and David Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal for the ACM (JACM)*, 42:1115–1145, 1995.

[55] Jonathan Goldstein, John C. Plat, and Christopher J. C. Burges. Redundant bit vectors for quickly searching high-dimensional regions. In *Proceedings of the First international conference on Deterministic and Statistical Methods in Machine Learning*, pages 137–158, Berlin, Heidelberg, 2005. Springer-Verlag.

[56] Miniwatts Marketing Group. http://www.internetworldstats.com/stats7.htm, June 2010.

[57] Marios Hadjieleftheriou, Amit Chandel, Nick Koudas, and Divesh Srivastava. Fast indexes and algorithms for set similarity selection queries. In *Proceedings of ICDE*, ICDE'08, 2008.

[58] Kenneth Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT'11, 2011. Association for Computational Linguistics.

[59] Monika Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of SIGIR*, SIGIR'06, 2006.

[60] AS Hildebrand, M Eck, S Vogel, and Alex Waibel. Adaptation of the translation model for statistical machine translation based on Information Retrieval. In *Proceedings of EAMT*, EAMT'05, 2005.

[61] Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of CIKM*, CIKM'12, pages 545–554, 2012. ACM.

[62] Lian'en Huang, Lei Wang, and Xiaoming Li. Achieving both high precision and high recall in near-duplicate detection. In *Proceedings of CIKM*, CIKM'08, pages 63–72, 2008. ACM.

[63] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of STOC*, STOC'98, pages 604–613, 1998. ACM.

[64] Frederick Jelinek. *Statistical methods for speech recognition.* MIT Press, Cambridge, MA, 1997.

[65] Yan Ke, Rahul Sukthankar, and Larry Huston. Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia*, pages 869–876, 2004.

[66] Tim Kiefer, Peter Benjamin Volk, and Wolfgang Lehner. Pairwise element computation with MapReduce. In *Proceedings of the ACM International Symposium on High Performance Distributed Computing*, HPDC'10, pages 826–833, 2010.

[67] Woosung Kim and Sanjeev Khudanpur. Language model adaptation using cross-lingual information. In *Proceedings of INTERSPEECH*, INTERSPEECH'03, 2003.

[68] Katrin Kirchhoff and Mei Yang. Improved language modeling for statistical Machine Translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, ParaText'05, pages 125–128, 2005. Association for Computational Linguistics.

[69] Philipp Koehn. *Statistical Machine Translation.* Cambridge University Press, 2010.

[70] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan

Herbst. Moses: open source toolkit for statistical Machine Translation. In *Proceedings of the ACL System Demonstrations*, ACL-Demos'07, pages 177–180, 2007. Association for Computational Linguistics.

[71] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of NAACL*, NAACL'03, pages 48–54, 2003. Association for Computational Linguistics.

[72] Aleksander Kolcz, Abdur Chowdhury, and Joshua Alspector. Improved robustness of signature-based near-replica detection via lexicon randomization. In *Proceedings of KDD*, KDD'04, 2004.

[73] Wessel Kraaij, Jian yun Nie, and Michel Simard. Embedding web-based statistical translation models in cross-language Information Retrieval. *Computational Linguistics*, 29:381–419, 2003.

[74] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Proceedings 12th International Conference on Computer Vision*, pages 2130–2137, 2009.

[75] K. L. Kwok. English-chinese cross-language retrieval based on a translation package. In *Workshop of Machine Translation for Cross Language Information Retrieval (Machine Translation Summit VII)*, pages 8–13, 1999.

[76] Victor Lavrenko, Martin Choquette, and W. Bruce Croft. Cross-lingual relevance models. In *Proceedings of SIGIR*, SIGIR'02, pages 175–182, 2002. ACM.

[77] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of SIGIR*, SIGIR'01, pages 120–127, 2001. ACM.

[78] Nicholas Lester, Alistair Moffat, William Webber, and Justin Zobel. Space-limited ranked query evaluation using adaptive pruning. In *Proceedings of the International Conference on Web Information Systems Engineering*, WISE'05, pages 470–477, Berlin, Heidelberg, 2005. Springer-Verlag.

[79] Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. Variational decoding for statistical Machine Translation. In *Proceedings of ACL*, ACL'09, pages 593–601, 2009.

[80] Jimmy Lin. Brute force and indexed approaches to pairwise document similarity comparisons with MapReduce. In *Proceedings of SIGIR*, SIGIR'09, 2009.

[81] Jimmy Lin and Chris Dyer. Data-intensive text processing with MapReduce. In *Proceedings of NAACL*, NAACL-Tutorials'09, pages 1–2, 2009. Association for Computational Linguistics.

[82] Jimmy Lin, Donald Metzler, Tamer Elsayed, and Lidan Wang. Of Ivory and Smurfs: Loxodontan MapReduce experiments for web search. In *Proceedings of TREC*, 2009.

[83] Michael Littman, Susan T. Dumais, and Thomas K. Landauer. Automatic Cross-Language Information Retrieval using latent semantic indexing. In *Cross-Language Information Retrieval, chapter 5*, pages 51–62. Kluwer Academic Publishers, 1998.

[84] Yi Liu, Rong Jin, and Joyce Y. Chai. A maximum coherence model for dictionary-based cross-language Information Retrieval. In *Proceedings of SIGIR*, SIGIR'05, pages 536–543, 2005. ACM.

[85] Adam Lopez. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of EMNLP-CoNLL*, EMNLP-CoNLL'07, pages 976–985, 2007.

[86] Adam Lopez. Statistical Machine Translation. *ACM Computing Surveys*, 40(3):8:1–8:49, August 2008.

[87] Yucheng Low and Alice X. Zheng. Fast top-k similarity queries via matrix compression. In *Proceedings of CIKM*, CIKM'12, pages 2070–2074, 2012. ACM.

[88] Yajuan Lü, Jin Huang, and Qun Liu. Improving statistical Machine Translation performance by training data selection and optimization. In *Proceedings of EMNLP*, EMNLP'07, pages 343–350, 2007.

[89] Walid Magdy and Gareth J. F. Jones. Should MT systems be used as black boxes in CLIR? In *Proceedings of ECIR*, ECIR'11, pages 683–686, Berlin, Heidelberg, 2011. Springer-Verlag.

[90] Gurmeet Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of WWW*, WWW'07, 2007.

[91] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[92] Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*, EMNLP'02, pages 133–139, 2002. Association for Computational Linguistics.

[93] Paul McNamee, James Mayfield, and Charles Nicholas. Translation corpus source and size in bilingual retrieval. In *Proceedings of NAACL*, NAACL'09, pages 25–28, 2009. Association for Computational Linguistics.

[94] Yashar Mehdad, Matteo Negri, and Marcello Federico. Detecting semantic equivalence and information disparity in cross-lingual documents. In *Proceedings of ACL*, ACL'12, pages 120–124, 2012. Association for Computational Linguistics.

[95] Helen M. Meng, Berlin Chen, Sanjeev Khudanpur, Gina-Anne Levow, Wai Kit Lo, Douglas W. Oard, Patrick Schone, Karen Tang, Hsin-Min Wang, and Jianqiang Wang. Mandarin-English information (MEI): Investigating translingual speech retrieval. *Computer Speech & Language*, 18(2):163–179, 2004.

[96] Donald Metzler and W. Bruce Croft. Combining the language model and inference network approaches to retrieval. *Information Processing & Management*, 40:735–750, September 2004.

[97] Donald Metzler and W. Bruce Croft. A markov random field model for term dependencies. In *Proceedings of SIGIR*, SIGIR'05, pages 472–479, 2005. ACM.

[98] Kimberly Moravec and Ingemar J. Cox. A comparison of extended fingerprint hashing and locality sensitive hashing for binary audio fingerprints. In *Proceedings of the ACM International Conference on Multimedia Retrieval*, ICMR'11, pages 31:1–31:8, 2011. ACM.

[99] Christopher Moretti, Jared Bulosan, Douglas Thain, and Patrick J. Flynn. All-Pairs: An abstraction for data-intensive cloud computing. In *Proceedings of IPDPS*, IPDPS'08, 2008.

[100] Dragos Stefan Munteanu and Daniel Marcu. Processing comparable corpora with bilingual suffix trees. In *Proceedings of EMNLP*, EMNLP'02, pages 289–295, 2002. Association for Computational Linguistics.

[101] Dragos Stefan Munteanu and Daniel Marcu. Improving Machine Translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504, 2005.

[102] Dragos Stefan Munteanu and Daniel Marcu. Extracting parallel sub-sentential fragments from non-parallel corpora. In *Proceedings of ACL*, COLING-ACL'06, pages 81–88, 2006. Association for Computational Linguistics.

[103] Dong Nguyen. Query translation for CLIR using Wikipedia. In *Twente Student Conference on IT*, 2008.

[104] Jian-Yun Nie. *Cross-Language Information Retrieval.* Morgan and Claypool Publishers, 2010.

[105] Vassilina Nikoulina, Bogomil Kovachev, Nikolaos Lagos, and Christof Monz. Adaptation of statistical Machine Translation model for Cross-Language Information Retrieval in a service context. In *Proceedings of EACL*, EACL'12, 2012.

[106] Franz Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics (CL)*, 29(1):19–51, 2003.

[107] Franz Josef Och and Hermann Ney. The alignment template approach to statistical Machine Translation. *Computational Linguistics (CL)*, 30(4):417–449, December 2004.

[108] Joseph Olive, Caitlin Christianson, and John McCary. *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[109] J. Scott Olsson and Douglas W. Oard. Combining LVCSR and vocabulary-independent ranked utterance retrieval for robust speech search. In *Proceedings of SIGIR*, SIGIR'09, pages 91–98, 2009.

[110] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of Machine Translation. In *Proceedings of ACL*, ACL'02, pages 311–318, 2002.

[111] Michael Persin, Justin Zobel, and Ron Sacks-Davis. Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society for Information Science and Technology (JASIST)*, 47(10):749–764, September 1996.

[112] John Platt, Kristina Toutanova, and Wen-Tau Yih. Translingual document representations from discriminative projections. In *Proceedings of EMNLP*, EMNLP'10, 2010.

[113] Jay M. Ponte and W. Bruce Croft. A language modeling approach to Information Retrieval. In *Proceedings of SIGIR*, SIGIR'98, pages 275–281, 1998.

[114] Martin Potthast, Benno Stein, and Maik Anderka. A Wikipedia-based multilingual retrieval model. In *Proceedings of ECIR*, ECIR'08, pages 522–530, Berlin, Heidelberg, 2008. Springer-Verlag.

[115] Michael O. Rabin. Fingerprinting by random polynomials. Technical Report TR-15-81, Center for Research in Computing Technology (Harvard University), 1981.

[116] Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering. In *Proceedings of ACL*, ACL'05, 2005.

[117] Philip Resnik and Noah A. Smith. The web as a parallel corpus. *Computational Linguistics (CL)*, 29:349–380, September 2003.

[118] Stephen Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *TREC-3*, 1994.

[119] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM (CACM)*, 18(11):613–620, November 1975.

[120] Germán Sanchis-Trilles and Francisco Casacuberta. Bayesian adaptation for statistical Machine Translation. In *Proceedings of the Joint IAPR International conference on Structural, Syntactic, and Statistical Pattern Recognition*, SSPR'10, pages 620–629, 2010.

[121] Sunita Sarawagi and Alok Kirpal. Efficient set joins on similarity predicates. In *Proceedings of SIGMOD*, SIGMOD'04, pages 743–754, 2004. ACM.

[122] Venu Satuluri and Srinivasan Parthasarathy. Bayesian Locality Sensitive Hashing for fast similarity search. *Proceedings of the VLDB Endowment*, 5(5):430–441, January 2012.

[123] Jacques Savoy and Samir Abdou. Experiments with monolingual, bilingual, and robust retrieval. In *Workshop of the Cross-Language Evaluation Forum*, CLEF'06, pages 137–144, 2006.

[124] Holger Schwenk. Investigations on large-scale lightly-supervised training for statistical Machine Translation. In *Proceedings of the International Workshop on Spoken Language Translation*, IWSLT'08, pages 182–189, 2008.

[125] Hee-Cheol Seo, Sang-Bum Kim, Hae-Chang Rim, and Sung-Hyon Myaeng. Improving query translation in english-korean cross-language Information Retrieval. *Information Processing & Management*, 41(3):507–522, May 2005.

[126] Abhinav Sethy, Panayiotis Georgiou, and Shrikanth Narayanan. Selecting relevant text subsets from web-data for building topic specific language models. In *Proceedings of NAACL*, NAACL'06, pages 145–148, New York City, June 2006. Association for Computational Linguistics.

[127] Claude E. Shannon and Warren Weaver. *A Mathematical Theory of Communication.* University of Illinois Press, Champaign, IL, 1963.

[128] Amit Singhal, Chris Buckley, Mandar Mitra, and Gerard Salton. Pivoted document length normalization. Technical report, Ithaca, 1995.

[129] Jason Smith, Chris Quirk, and Kristina Toutanova. Extracting parallel sentences from comparable corpora using document level alignment. In *Proceedings of NAACL: Human Language Technologies*, NAACL-HLT'10, 2010.

[130] Jason Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. Dirt cheap web-scale parallel text from the common crawl dirt cheap web-scale parallel text from the common crawl dirt cheap web-scale parallel text from the common crawl dirt cheap web-scale parallel text from the common crawl. In *Proceedings of ACL*, ACL'13, 2013.

[131] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of CIKM*, CIKM'07, pages 623–632, 2007. ACM.

[132] Matthew Snover, Bonnie Dorr, and Richard Schwartz. Language and translation model adaptation using comparable corpora. In *Proceedings of EMNLP*, EMNLP'08, pages 857–866, 2008. Association for Computational Linguistics.

[133] Andreas Stolcke. Srilm - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, ICSLP'02, pages 901–904, 2002.

[134] Trevor Strohman and W. Bruce Croft. Efficient document retrieval in main memory. In *Proceedings of SIGIR*, SIGIR'07, pages 175–182, 2007. ACM.

[135] Martin Theobald, Jonathan Siddharth, and Andreas Paepcke. SpotSigs: robust and efficient near duplicate detection in large web collections. In *Proceedings of SIGIR*, SIGIR '08, 2008.

[136] Jörg Tiedemann. Context adaptation in statistical Machine Translation using models with exponentially decaying cache. In *Proceedings of the ACL Workshop on Domain Adaptation for Natural Language Processing*, DANLP'10, pages 8–15, 2010.

[137] Jörg Tiedemann. *Bitext Alignment.* Morgan & Claypool Publishers, 1st edition, 2011.

[138] Huihsin Tseng, Pi-Chuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. A conditional random field word segmenter. In *SIGHAN Workshop on Chinese Language Processing*, 2005.

[139] Ferhan Ture, Douglas W. Oard, and Philip Resnik. Encouraging consistent translation choices. In *Proceedings of NAACL: Human Language Technologies*, NAACL-HLT'12, pages 417–426, 2012. Association for Computational Linguistics.

[140] Howard Turtle and James Flood. Query evaluation: strategies and optimizations. *Information Processing & Management*, 31(6):831–850, November 1995.

[141] Jakob Uszkoreit, Jay M. Ponte, Ashok C. Popat, and Moshe Dubiner. Large scale parallel document mining for Machine Translation. In *Proceedings of COLING*, COLING'10, pages 1101–1109, 2010. Association for Computational Linguistics.

[142] Ashish Venugopal, Jakob Uszkoreit, David Talbot, Franz J. Och, and Juri Ganitkevitch. Watermarking the outputs of structured prediction with an application in statistical Machine Translation. In *Proceedings of EMNLP*, EMNLP'11, pages 1363–1372, 2011. Association for Computational Linguistics.

[143] Rares Vernica, Michael Carey, and Chen Li. Efficient parallel set-similarity joins using MapReduce. In *Proceedings of SIGMOD*, SIGMOD'10, 2010.

[144] Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proceedings of COLING*, COLING'96, 1996.

[145] Jianqiang Wang and Douglas W. Oard. Combining bidirectional translation and synonymy for cross-language Information Retrieval. In *Proceedings of SIGIR*, SIGIR'06, pages 202–209, 2006. ACM.

[146] Dan Wu and Daqing He. Exploring the further integration of Machine Translation in multilingual information access. In *Proceedings of iConference '10*, 2010.

[147] Jinxi Xu and Ralph Weischedel. Empirical studies on the impact of lexical resources on CLIR performance. *Information Processing & Management*, 41:475–487, May 2005.

[148] Jinxi Xu, Ralph Weischedel, and Chanh Nguyen. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of SIGIR*, SIGIR'01, pages 105–110, 2001. ACM.

[149] Hui Yang and Jamie Callan. Near-duplicate detection by instance-level constrained clustering. In *Proceedings of SIGIR*, SIGIR'06, 2006.

[150] Reyyan Yeniterzi and Kemal Oflazer. Syntax-to-morphology mapping in factored phrase-based statistical Machine Translation from english to turkish. In *Proceedings of ACL*, ACL'10, 2010.

[151] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc Information Retrieval. In *Proceedings of SIGIR*, SIGIR'01, pages 334–342, 2001. ACM.

[152] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to Information Retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, April 2004.

[153] Jiaqi Zhai, Yin Lou, and Johannes Gehrke. ATLAS: a probabilistic algorithm for high dimensional similarity search. In *Proceedings of SIGMOD*, SIGMOD'11, pages 997–1008, 2011. ACM.

[154] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. Self-taught hashing for fast similarity search. In *Proceedings of SIGIR*, 2010.

[155] Wei Zhang, Shuang Liu, Clement Yu, Chaojing Sun, Fang Liu, and Weiyi Meng. Recognition and classification of noun phrases in queries for effective retrieval. In *Proceedings of CIKM*, CIKM'07, pages 711–720, 2007. ACM.

[156] Bing Zhao, Matthias Eck, and Stephan Vogel. Language model adaptation for statistical Machine Translation with structured query models. In *Proceedings of COLING*, COLING'04, 2004.

[157] Bing Zhao and Stephan Vogel. Adaptive parallel sentences mining from web bilingual news collection. In *Proceedings of ICDM*, ICDM'02, pages 745–748, Washington, DC, 2002. IEEE Computer Society.

[158] Dong Zhou and Vincent Wade. The effectiveness of results re-ranking and query expansion in Cross-Language Information Retrieval. In *Proceedings of NTCIR-8 Workshop Meeting*, 2010.