ABSTRACT

Title of Dissertation: TRANSFER LEARNING IN NATURAL LANGUAGE PROCESSING THROUGH INTERACTIVE FEEDBACK

Michelle Yuan
Doctor of Philosophy, 2022

Dissertation Directed by: Professor Jordan Boyd-Graber
Department of Computer Science
College of Information Studies
Language Science Center
Institute for Advanced Computer Studies

Machine learning models cannot easily adapt to new domains and applications. This drawback becomes detrimental for natural language processing (NLP) because language is perpetually changing. Across disciplines and languages, there are noticeable differences in content, grammar, and vocabulary. To overcome these shifts, recent NLP breakthroughs focus on transfer learning. Through clever optimization and engineering, a model can successfully adapt to a new domain or task. However, these modifications are still computationally inefficient or resource-intensive. Compared to machines, humans are more capable at generalizing knowledge across different situations, especially in low-resource ones. Therefore, the research on transfer learning should carefully consider how the user interacts with the model. The goal of this dissertation is to investigate "human-in-the-loop" approaches for transfer learning in NLP.

First, we design annotation frameworks for inductive transfer learning, which is the transfer

of models across tasks. We create an interactive topic modeling system for users to find topics useful for classifying documents in multiple languages. The user-constructed topic model bridges improves classification accuracy and bridges cross-lingual gaps in knowledge. Next, we look at popular language models, like BERT, that can be applied to various tasks. While these models are useful, they still require a large amount of labeled data to learn a new task. To reduce labeling, we develop an active learning strategy which samples documents that surprise the language model. Users only need to annotate a small subset of these unexpected documents to adapt the language model for text classification.

Then, we transition to user interaction in transductive transfer learning, which is the transfer of models across domains. We focus our efforts on low-resource languages to develop an interactive system for word embeddings. In this approach, the feedback from bilingual speakers refines the cross-lingual embedding space for classification tasks. Subsequently, we look at domain shift for tasks beyond text classification. Coreference resolution is fundamental for NLP applications, like question-answering and dialogue, but the models are typically trained and evaluated on one dataset. We use active learning to find spans of text in the new domain for users to label. Furthermore, we provide important insights on annotating spans for domain adaptation.

Finally, we summarize the contributions of each chapter. We focus on aspects like the scope of applications and model complexity. We conclude with a discussion of future directions. Researchers may extend the ideas in our thesis to topics like user-centric active learning and proactive learning.

TRANSFER LEARNING IN NATURAL LANGUAGE PROCESSING THROUGH INTERACTIVE FEEDBACK

by

Michelle Yuan

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
2022

Advisory Committee:
Professor Jordan Boyd-Graber, Chair/Advisor
Professor Philip Resnik (Dean's Representative)
Professor Benjamin Van Durme
Professor Rachel Rudinger
Professor John Dickerson

Dedication

To my parents, my sister, and the mountains.

Acknowledgments

My completion of this thesis is only possible because of the support from loving family members and friends.

First, I would like to thank my advisor, Jordan Boyd-Graber, for his extensive mentorship. I joined his lab with minimal research experience and NLP knowledge. Luckily, Jordan had a very hands-on approach to advising students and taught me everything from scratch. From him, I have learned many essential research skills, like writing papers, analyzing experiments, and giving presentations. I have even learned to create engaging videos about my work! Most importantly, Jordan has taught me to have fun while conducting research. Sometimes, I become too bothered by the practical issues and forget to enjoy the journey. I will remember these valuable lessons and continue to develop the skills acquired under Jordan's tutelage.

Second, I want to thank Benjamin Van Durme for his guidance throughout these years. Ben has contributed many of the ideas that are present in this thesis. In our meetings, Ben would often provide an intriguing yet realistic perspective on research and connect our work to applications in industry. These insightful discussions have helped me structure and organize my thesis. Through Ben, I also have had the opportunity to work with fantastic people from the HLTCOE, such as Patrick Xia.

I also wish to thank the rest of the committee members. Philip Resnik gives some of the most informative and entertaining lectures. The content from his *Computational Linguistics*

course has influenced my work. John Dickerson's class on *Applied Mechanism Design for Social Good* has encouraged me to think about the social impacts of my thesis. Rachel Rudinger's feedback on my proposal has inspired me to consider more linguistically-informed approaches.

I owe much of my academic growth to the CLIP Lab at Maryland. The group is filled with talented NLP researchers who are willing to help each other at any given point in time. I am thankful to have received feedback on my work from the brilliant professors in CLIP, like Hal Daumé III and Marine Carpuat. I want to thank HyoJung Han for her support and accompanying me to various activities, such as rock climbing and hiking. Other students who have tremendously helped me are Weiwei Yang, Pedro Rodriguez, Shi Feng, Yogarshi Vyas, Mozhi Zhang, Pranav Goel, Sweta Agrawal, and Chenglei Si. I will cherish the moments spent with the people mentioned above, as well as Fenfei Guo, Andrew Mao, Naveen Raman, Connor Baumler, Navita Goyal, and Neha Srikanth.

The Computer Science Department at UMD has been conducive to my research. When we moved from the dark dungeons of A.V. Williams to the shining, shimmering, and splendid Iribe center, I was absolutely ecstatic. I will always miss my window seat in room 4108 with the stunning campus view. For administrative issues, I am grateful for the timely assistance from program director Tom Hurst. I want to thank Max Ehrlich for answering my every question about the dissertation defense and Manasij Venkatesh for valuable advice on surviving graduate school. Last but not least, I would like to thank Yi-Ting Chen for her friendship, comfort, and care.

My one-year research excursion in Taiwan was impactful and unforgettable. I finally improved my Chinese reading skills in order to navigate places on my own. At National Taiwan University, I was advised by Hsuan-Tien Lin. I am grateful for his guidance on active learning and information theory. I am thankful to have met the students in CLLab who have taught me how

to execute experiments more efficiently. It is because of Kuen-Han Tsai that I have become proficient at VIM and adopted good habits for implementing machine learning models. Moreover, I want to thank Chien-Min Yu and Si-An Chen for the fruitful research discussions.

The summer internships have expanded my professional network and exposed me to a wide range of machine learning problems. I am thankful for the people that I have met at ASAPP, Appier, and Adobe. I particularly want to thank Ani Nenkova and Rajiv Jain for their guidance.

Beyond the AI research community, there are several people who have helped me along the way. The Cornell Wushu club was my closest group of friends in college and are still like family up to this day. Thank you for the career advice, encouragement, and jiayous. At Maryland, I am very thankful for all the friends from rock climbing, badminton, and wushu. I will miss all the fun times spent together on campus. I am grateful for the companionship of Yihua Lee and Zhixin Jiang since my first year of graduate school. As always, I want to deeply thank my long-time backpacking friends, especially Ming-Chi Lin and Yi-Ling Cheng, for offering me solace in times of need.

Finally, I extend the deepest gratitude to the most important people in my life: my family. While I was in Taiwan, I became much closer with my grandparents, especially my grandpa who has been fully supportive of my life choices. I want to thank Ashley Yuan for being the best sister and my best friend. Although she is much younger than me, she is wise beyond her years and gives some of the most useful advice. Lastly, I am most grateful for my parents, Chao Yuan and Chi-Hsin Mou. They have sacrificed much of their time and energy to raise me. Whether it was completing the Ph.D. or finishing a difficult backpacking trip, they have always believed that a klutz like me could surpass herself. Through their unwavering support, I have finally achieved the dream of earning a doctoral degree.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
ALPS	Active Learning by Processing Surprisal
ALTO	Active Labeling with Topic Overviews
BADGE	Batch Active Learning by Diverse Gradient Embeddings
BART	Bidirectional and Auto-Regressive Transformers
BERT	Bidirectional Encoders from Transformers
CBOW	Continuous Bag-of-words
CCA	Canonical Correlation Analysis
CLIME	Classifying Interactively with Multilingual Embeddings
CLWE	Cross-lingual Word Embeddings
CR	Coreference Resolution
C2F-COREF	Coarse-to-Fine Coreference Resolution
ELMo	Embeddings from Language Models
EN	English
GLoVE	Global Vectors for Word Representations
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
HIT	Human Intelligence Task
ICOREF	Incremental Coreference Resolution
IDF	Inverse Document Frequency
IE	Information Extraction
LDA	Latent Dirichlet Allocation
LORELEI	Low Resource Languages for Emergent Incidents
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory Network
mBERT	Multilingual BERT

MCTA	Multilingual Cultural-common Topic Analysis
MTurk	Mechanical Turk
MTAnchor	Multilingual Topic Anchors
MUC-4	Message Understanding Conference
NER	Named Entity Recognition
NLP	Natural Language Processing
NPMI	Normalized Pointwise Mutual Information
NQ	Natural Questions
PMI	Pointwise Mutual Information
PLTM	Polylingual Topic Model
QA	Question Answering
RNN	Recursive Neural Network
RoBERTa	Robustly Optimized BERT Pre-training Approach
SI	Sinhalese
SQuAD	Stanford Question Answering Dataset
SST	Stanford Sentiment Treebank
SVD	Singular Value Decomposition
TFIDF	Term Frequency–Inverse Document Frequency
T5	Text-to-Text Transfer Transformer
XLM	Cross-lingual Language Model
XLM-R	Cross-lingual Language Model with RoBERTa
XMod	Cross-lingual Modular Model
ZH	Chinese

Chapter 1: Introduction

Language is a form of communication and a means of self-expression. Through language, we convey information, beliefs, and emotions. For better or worse, it is a medium that evolves with humans. As a result, we see variation in language across different geographic locations and time periods. Currently, there are 7,117 languages in the world and 3,982 of them have a written system (Eberhard et al., 2020). Over time, new words are added to the vocabulary and the semantics for existing ones often change. The dynamic nature of language makes it challenging yet compelling to study.

To analyze these shifts in human language, natural language processing (NLP) uses computers. NLP has its humble beginnings in symbolic systems with influential works like *Computing Machinery and Intelligence* (Turing and Haugeland, 1950) and *Syntactic Structures* (Chomsky, 1975). The symbolic systems are built from handwritten rules and could not make any decisions beyond what is hard-coded. In the 1990s, statistical NLP becomes prevalent after the expansion of the Internet and improved machinery. The Internet enables more text to be accumulated for data collection and the upgraded hardware provides more computing power to analyze language. With these resources, NLP researchers aggressively use statistics and machine learning to model the information content on the Web. Since the 2010s, NLP research has shifted its focus to GPU-powered neural networks. These deep learning models require less feature engineering

and encourage end-to-end training. Despite the ever-changing trends in the field, modeling the variations in language remains a challenge.

In NLP research, machine learning plays an important role for training models. A long-standing problem of machine learning is adapting the models to unobserved situations. Typically, these models are trained on a large, annotated dataset. Then, their predictions are quite accurate for data from the same training distribution. However, when it comes to out-of-domain data, these models may fail to generalize and the results are disappointing. Humans can quickly process new information and link them to past experiences, but artificial intelligence (AI) cannot easily acquire new knowledge if only trained on a few examples. This brittleness is especially harmful for NLP because language is constantly changing. A model that has only trained on English documents may fail to understand text in Shona, a written language of Zimbabwe. A model that has only trained on documents before 2019 may not detect a global pandemic as an important topic. How can NLP models overcome these dynamic shifts in language?

To adapt machine learning models to new problems, we look to *transfer learning* (Pan and Yang, 2010; Ruder, 2019). This area of research is concerned with transferring the knowledge from models over different domains and downstream tasks. In NLP, we care about accumulating general knowledge about language. One way to achieve this is through *pre-training*. The term refers to preliminary training on massive corpora with a self-supervised loss function that does not require labeled data. An example is the *language modeling* loss that measures the likelihood of a word given its context. After pre-training, the model is then adapted by further training it on data from the task and domain of interest. Given the success of these methods, most research in transfer learning focus on improving model architecture and optimization.

While these structural modifications are important, model adaptation still relies on labeled

data and computing power. During some emergent situations, we may not have the resources to label a large dataset, tune model parameters, and train the model on the entire dataset. How else could we quickly adapt the model to new domains and tasks? Previously, we have mentioned that language evolves with humans. Without much experience and thinking, humans can easily transfer their knowledge across several problems. On the other hand, machines must rely on abundant data and computing resources to achieve such a skill (Griffiths et al., 2019). Therefore, we should consider ways that a human could help redirect the model toward generalizing language understanding.

Ideally, we would want the user to briefly interact with the model. Through repeated interactions, the model acquires the relevant pieces of knowledge from the user. The type of user-AI interaction would depend on the task at hand. If we want to teach the English model to classify text in Shona, a bilingual user could align some keywords between the two languages. If we want an outdated model to learn about the COVID-19 pandemic, a user could label spans of text related to the Coronavirus. In both situations, we find missing gaps in knowledge for the user to fill in. However, it is unclear how to locate these missing gaps and update the model based on human input. Thus, this dissertation investigates the methodology for incorporating user feedback to facilitate model transfer in NLP.

1.1 Motivation

The origins of transfer learning in NLP can be found in latent semantic analysis (Deerwester et al., 1990), an algorithm that creates low-dimensional vector representations of documents based on word occurrences. The purpose of these vector representations, now also known

as *embeddings*, is to encode important, general knowledge about the data. Later on, text modeling follows a more statistical approach through topic modeling where the documents are defined as probabilistic mixtures of topics (Blei et al., 2003). After decades of statistical NLP, neural networks are used to train word embeddings (Mikolov et al., 2013c) that can then form vector representations for documents. As neural NLP progresses, researchers emphasize the importance of transferring knowledge through learning general representations (Bengio et al., 2013). In recent years, the transformer models, like BERT (Devlin et al., 2019), learn contextual word embeddings to capture semantics in different contexts. As a result, the embeddings can more effectively represent words in new contexts and help transfer the model to various applications.

The goal of transfer learning is to adapt models for shifts in the data domain or downstream task. This thesis is concerned with two types of transfer learning: inductive transfer learning and transductive transfer learning (Pan and Yang, 2010). *Inductive transfer learning* involves transferring models for different tasks. An example would be developing a model that can be used for multiple tasks like text classification, question answering, and natural language inference (Peters et al., 2018). *Transductive transfer learning* deals with transferring knowledge across different domains for the same task. An example would be building a sentiment classifier for different languages (Chen et al., 2018b). We formally define these two types of transfer learning in Section 2.2. Through transfer learning, we improve the model's ability to generalize across different settings.

The research in transfer learning primarily emphasizes the developments in model architecture and mechanisms. From topic models to word embeddings to transformer models, each breakthrough focuses on how text is encoded and how the model is trained. The results are impressive, but these innovations may be impractical for low-resource situations. What happens

when we lack resources, like data or computing power? Given a new task, we need to collect labeled data, which could possibly take years to preprocess and annotate (Lewis, 1992; Marcus et al., 1994). While Mechanical Turk has simplified dataset creation, collecting vast amounts of high-quality annotation from crowdsourced workers is still difficult (Callison-Burch and Dredze, 2010). After obtaining labeled data, we still need to adapt the model. Training a model like BERT can cost thousands of dollars in cloud compute and over a thousand of pounds in carbon emissions, which is nearly equivalent to a flight from New York to San Francisco (Strubell et al., 2019)!

In many real-life settings, there are not enough resources to label large amounts of data and train an enormous model on the data. During the 2010 Haiti Earthquake, international relief workers heavily rely on analyzing social media posts to mitigate destruction (Yates and Paquette, 2010). Early in 2020, language scientists gather reports on the Coronavirus pandemic to help with clinical research (Voorhees et al., 2020). For emerging situations like these, we require an approach that can immediately adapt the model for the current situation. If researchers only prioritize advancements in model architecture, the true potential of transfer learning may never be completely realized.

The model implementation is only a small component of AI systems (Sculley et al., 2015). In fact, much of the system design relies on the feedback loops. How do we update the model given new information or changes in the external world? A straightforward approach is having a human who can guide the process for updating models. Much of the research in human-in-the-loop machine learning stems from the need to explain model predictions (Swartout, 1983). For external users and even model developers, the AI system is often a black box. In the General Data Protection Regulation, the European Union emphasizes the "right to explanation" when using AI

for decision-making (European Parliament and Council of the European Union, 2016). For high-stakes situations, we should only use models that are interpretable (Rudin, 2019). Nowadays, machine learning research is focused on human-centric AI with the purpose of increasing transparency and control for users (Renner, 2020). Users are considered stakeholders of an AI system, so the models must consider their feedback and needs (Shneiderman, 2022).

Beyond transparency and control, we argue that transfer learning may also benefit from human-AI interaction. When humans discover a new problem, they can quickly apply past skills and experiences (Taatgen, 2013), but machine learning models fail to transmit knowledge as successfully as humans do (Griffiths et al., 2019). So, simple feedback from a human could provide the right signal for model adaptation. The idea of transferring models through interaction sounds reasonable, but the implementation and consequences are unclear. How should we design a user feedback loop for transfer learning? Therefore, the research in transfer learning should prioritize interactive approaches, especially when it comes to language.

1.2 Objectives

The dissertation develops methods for interactive transfer learning and analyzes the effect of human feedback on natural language understanding. Human interaction is either conducted in a controlled user study or simulated by computer algorithms. The specific goals and research questions are as follows:

1. **Scope of Applications:** As transfer learning indicates shift in either domain or task, the interactive learning component differs in each situation. We explore human interaction for inductive and transductive transfer learning problems. How should users interact with the

model to adapt it to a new task? How should users interact with the model to adapt it to a new domain? What are the similarities and differences between these interactions?

2. **Low-resource Settings:** Many languages are *low-resource*, which means that data and knowledge resources are scarce. For low-resource NLP, transfer learning might rely even more on human feedback. Along with having insufficient training data, computational resources may also be limited. To conserve resources, user feedback needs to provide the most useful information about the task. How can we design the user interaction to overcome situations in which we lack resources like data, time, or computing power?
3. **Model Complexity:** NLP models show various levels of complexity. A logistic regression model may only have one hundred parameters while GPT-3 has 175 billion of them. Large language models can solve more difficult tasks and are more adept at transfer learning. Compared to smaller models, they may depend less on user feedback or require a different form of interaction. As model complexity increases, how should we change the interaction between the user and the model?

1.3 Organization

Chapter 2 provides background information on transfer learning and interactive learning. First, we discuss the failures of supervised learning to motivate the need for transfer learning. Then, we describe various problems and settings in transfer learning. The two main divisions are inductive transfer learning, which involves shift in task, and transductive transfer learning, which refers to shift in the domain space. The chapter then introduces interactive learning by describing the ways for the machine learning models to learn from user feedback. We discuss how active

learning selects data for humans to label and describe work in human-AI collaboration.

Chapters 3 to 6 are split into two parts. Part I presents application of interactive learning on inductive transfer learning. Part II focuses on user interaction for transductive transfer learning. In each part, we organize the chapters in increasing order of model complexity and task difficulty. By doing this, we can observe changes in the user feedback loop as the problem of knowledge transfer becomes more complicated.

Part I starts the discussion on inductive transfer learning with Chapter 3. In this chapter, we create a system called MTAnchor for users to cross-lingually align topics for text classification. While topics help transfer knowledge across languages, human interaction can help refine topics for classification tasks. In Chapter 4, we describe our work on ALPS, which is an active learning algorithm that helps adapt pre-trained language models for downstream tasks. By finding text that surprises the model, we can focus labeling efforts on these unpredictable examples. With the surprisal values, we reduce the amount of labeled data needed to quickly adapt language models for text classification.

Next, we transition to transductive learning in Part II. In Chapter 5, we discuss CLIME, a human-in-the-loop framework that aligns the cross-lingual word embedding space according to user feedback. CLIME can be combined with active learning to further improve classification accuracy. This shows that both document-level and word-level annotations help with transfer learning. Chapter 6 focuses on domain shift for more complicated tasks like coreference resolution. For coreference resolution, models are typically developed with one dataset and may not generalize to new domains. We use active learning to label spans for quickly adapting models. For sampling data, we look at spans that confuse mention detection or clustering. We conclude with insights about labeling spans for adapting coreference resolution models to new domains.

Finally, Chapter 7 summarizes contributions and discusses future research directions. Future work can develop user-centric active learning where the sampling strategy considers data that users prefer to label. Beyond active learning, there is proactive learning, a framework that not only determines which examples to label but also matches each example to the most appropriate annotator. Additionally, we mention the importance of online, interactive updates. Rather than adapt models through retraining, more work should develop online algorithms to immediately update models and reduce latency in the user feedback loop.

Chapter 2: Background

The dissertation is about interactive approaches to transfer learning in NLP, so this chapter provides background information on transfer learning and interactive learning. Before we discuss transfer learning, we first introduce supervised learning in Chapter 2.1. Supervised learning is the traditional framework for training machine learning models. We analyze the failures of supervised learning during sudden shifts in tasks and data domains. Next, we delve into transfer learning, which aims to overcome these issues in supervised learning (Section 2.2). Transfer learning can be inductive (Section 2.2.1) or transductive (Section 2.2.2). We explore the taxonomy and history of transfer learning in NLP by examining seminal models and algorithms. Finally, we look at interactive learning to understand the effect of human feedback on machine learning models (Section 2.3). We focus on active learning (Section 2.3.1), a structured framework for obtaining instance-level annotations, and discuss innovative human-in-the-loop interfaces for NLP tasks (Section 2.3.2).

2.1 Supervised Learning

Machine learning is a computer program using experience to improve upon performance, as measured by certain metrics, on some class of tasks (Mitchell, 1997). Supervised learning is the traditional way of training machine learning models on large amounts of labeled data. Through

supervised learning, AI systems can automate several tasks across various fields. Goodfellow et al. (2016) provides an overview of such breakthroughs, like AlexNet, a a neural network that can successfully recognize more than 10,000 objects in the ImageNet database (Krizhevsky et al., 2012). While supervised learning shows tremendous success, the caveat is that the training set must resemble the data that appears at test time. In the following sections, we will describe details and limitations of supervised learning.

Suppose that we have a training dataset of n instances, $\boldsymbol{X} = \boldsymbol{x}_1, \ldots \boldsymbol{x}_n$, with corresponding labels $\boldsymbol{y} = y_1, \ldots, y_n$. We will also refer to an instance $\boldsymbol{x}_i$ as an example. Assume each instance $\boldsymbol{x}_i$ belongs to feature space $\mathcal{X}$ and each label y_i belong to label space $\mathcal{Y}$. For classification problems, the label space $\mathcal{Y}$ is a finite set of items. For regression problems, the label space $\mathcal{Y}$ is the set of real numbers $\mathbb{R}$. For simplicity, we will assume the task is a classification problem where $\mathcal{Y}$ is a finite set of labels. Now, the goal of training is to learn a mapping $g : \mathcal{X} \to \mathcal{Y}$ such that g assigns the correct label y_i to instance $\boldsymbol{x}_i$.

A common framework in supervised learning is to learn a function $f : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}|}$ and let $g(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} f(\boldsymbol{x})_y$. An interpretation of $f(\boldsymbol{x})_y$ is that it scores how likely instance $\boldsymbol{x}$ is assigned as label y. Then, g assigns the label to $\boldsymbol{x}$ such that score $f(\boldsymbol{x})_y$ is maximized. To learn function f, we look to minimize *expected risk*, which we define as

$$R(f) = \mathbb{E}_{p(\boldsymbol{x},y)} \left[L(f(\boldsymbol{x}), y) \right] = \int L(f(\boldsymbol{x}), y) \, dp(\boldsymbol{x}, y). \tag{2.1}$$

The expected risk measures the *loss* L of our model prediction for an arbitrary instance $\boldsymbol{x}$ with label y drawn from the data distribution $p(\boldsymbol{x}, y)$. The loss function L measures the amount of error in model prediction. In other words, the risk measures how poorly the function f learns the

task and the loss computes the difference between one prediction $f(\boldsymbol{x})$ and the actual label y. A common choice for L is cross-entropy loss,

$$L_{CE}(f(\boldsymbol{x}), y) = -\log f(\boldsymbol{x})_y. \tag{2.2}$$

Intuitively, machine learning should aim to minimize expected risk. Low expected risk implies that the model should correctly label most instances on average. Unfortunately, $R(f)$ is impossible to compute because we do not know the true distribution $p(\boldsymbol{x}, y)$. So, we instead use *empirical risk* to approximate expected risk. The empirical risk is the average loss over examples in the training dataset,

$$\widehat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(\boldsymbol{x}_i), y_i). \tag{2.3}$$

For this approximation to work, we assume that the instance-label pairs $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ are i.i.d, distributed independently and identically to the true distribution $p(\boldsymbol{x}, y)$. Under this assumption, we can train models through minimizing empirical risk $\widehat{R}(f)$ rather than having to compute expected risk $R(f)$. Consequently, the bounds on this approximation depends on the size of the training dataset. As we approach an infinite number of training instances, the difference between empirical risk $\widehat{R}(f)$ and expected risk $R(f)$ decreases. Thus, supervised learning has higher chance of success if trained on large, labeled datasets.

In practice, we usually have a validation dataset. After we train the model on the training dataset, we evaluate on the validation dataset. The results help us tune model hyperparameters and show us if there is *overfitting*. This term means that the model is only learning the task for the training dataset but does not generalize to arbitrary instances $\boldsymbol{x}$ from the true data distribution.

After model development with the validation dataset, we then perform the final evaluation on the test dataset and report results.

Having a validation dataset may reduce overfitting, but the size of the training dataset is still a bottleneck. Without training on enough data, the empirical risk $\widehat{R}(f)$ cannot closely approximate the expected risk $R(f)$. Intuitively, this makes sense because the model needs to learn from as much data as possible. Then, it can correctly make predictions for any example at test time because a similar example has likely been observed before.

2.1.1 Model Complexity

Obtaining a large amount of labeled data is important for training machine learning models. Another critical factor is the **model complexity**. It generally refers to the model's expressive power, or in other words, the ability to capture the complexity of the function f that it is learning (Hu et al., 2021). If f is a simple function (e.g. linear), then we do not need a complex model to learn it. Sometimes, a complex model may overfit to random noise from the data. However, if f is a complicated function (e.g. high-degree polynomial), we would need a complex model. A simple model may underfit and not properly capture the complexity of f. There are various ways to measure model complexity. In statistical learning, Rademacher complexity is commonly used to measure the degree of a hypothesis space fitting random noise (Bartlett and Mendelson, 2002). Other measures may be model-specific. For example, the complexity of decision trees could be determined by tree depth or number of leaf nodes.

Hu et al. (2021) indicate that model size is an important factor to consider when quantifying model complexity. A model with more parameters has more expressive power and is more prone

to overfitting on the training data. *For this thesis, we will measure model complexity by the number of parameters.* When we discuss the contributions in Parts I and II, we order each work according to the model complexity to understand its connection to transfer learning. While more complex models may generalize to more domains and tasks, they may not be practical for low-resource situations (Chapters 3, 5). Therefore, we consider models of varying complexity in this thesis.

2.1.2 Poverty of the Stimulus

As long as we curate a large training dataset that resembles the situation at test time, we can use supervised learning to train a robust machine learning model for a task. In reality, it is impractical to create such a training dataset for every possible task. Complete and accurate annotations are expensive to build a machine learning system (Roh et al., 2019). For example, an annotator may take fifteen hours or more to label entities and relations for only a hundred of news articles (Settles et al., 2008a), which is not nearly enough to train a model. In emergency situations, we need to quickly train a model with limited resources and cannot afford the costs associated with labeling. For some languages, there may not be enough unlabeled data to begin with! With scarcity in labeled data, how can we train machine learning models?

In linguistics, the "Poverty of the Stimulus" argument claims that people do not learn language exclusively through experience (Chomsky, 1980; Pullum and Scholz, 2002). The evidence is straightforward: young children acquire linguistic abilities despite limited exposure to their environment. Therefore, humans must be born with basic knowledge of language. From a statistical modeling perspective, Mitchell (1980) frame this innate knowledge as *inductive bias*, which

is "any basis for choosing one generalization over another, other than strict consistency with the observed training instances". Training AI systems may be viewed as automatically learning inductive bias (Baxter, 2000).

Today's research is largely dominated by data-driven deep learning, but these state-of-the-art models require an abundance of labeled data. While machine learning systems use enormous amount of computing power and data to automate specific tasks, humans solve several problems with limited computation capacity and experience (Griffiths et al., 2019). For cognitive reasoning problems, humans make more robust decisions than language models, especially on out-of-distribution data (Collins et al., 2022). Therefore, we need to rethink supervised learning and develop models that can also learn under a "poverty of the stimulus". In the next section, we show how transfer learning can bridge knowledge between machine learning models.

2.2 Transfer Learning

The success of supervised learning is limited by the amount of labeled data. While humans can learn new concepts quickly (Taatgen, 2013), AI fails to instantly generalize across new domains. Transfer learning emerges to mitigate this issue in supervised learning. In a 2016 NeurIPS tutorial, Andrew Ng prophesizes, "Transfer learning will be next driver of machine learning success". If we want to deploy AI systems across all fields and sectors, we must innovate approaches beyond supervised learning.

In transfer learning, we assume that the *source* model is reliable to use for a certain domain and task. Our goal is to adapt the source model into a *target* model. The target model is difficult to train on its own, so we need to rely on the source model. For example, the source model

could be a topic classifier for news articles in English. The model can correctly predict whether the article is mainly about sports, medicine, technology, etc. Now, we want to use this model for two other applications: classifying news articles in Shona and predicting the author of the English articles. If we have no labeled data in Shona, how may we modify the English classifier to make predictions for Shona? If we labeled data about authorship is scarce, how do we apply the knowledge about topics to predict the author?

Transfer learning is machine learning with a shift in domain $\mathcal{D}$ or task $\mathcal{T}$ (Pan and Yang, 2010). A domain $\mathcal{D} = \{\mathcal{X}, p(\boldsymbol{X})\}$ is defined by a feature space $\mathcal{X}$ and a marginal probability distribution $p(\boldsymbol{X})$ over $\mathcal{X}$. The random matrix $\boldsymbol{X}$ represents the examples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ where $\boldsymbol{x}_i \in \mathcal{X}$. A task $\mathcal{T} = \{\mathcal{Y}, p(\boldsymbol{Y}), p(\boldsymbol{Y} \mid \boldsymbol{X})\}$, is defined by the label space $\mathcal{Y}$, a prior distribution on the label space $p(\boldsymbol{Y})$, and a conditional probability distribution $p(\boldsymbol{Y} \mid \boldsymbol{X})$. The random vector $\boldsymbol{Y}$ represents labels $y_1, \ldots, y_n$ where $y_i \in \mathcal{Y}$. During training, the model usually learns the label distribution $p(\boldsymbol{Y} \mid \boldsymbol{X})$.

Now, suppose that the source model is trained on source domain $\mathcal{D}_S$ for source task $\mathcal{T}_S$. In the above example, the source domain is English news articles and the source task is topic categorization. The model can reasonably compute $p_S(\boldsymbol{Y}_S \mid \boldsymbol{X}_S)$, which is the distribution of the topic labels given the English article. However, it has difficulty transferring information to either a target domain $\mathcal{D}_T$ or task $\mathcal{T}_T$. The target domain is news articles in Shona. The target task is predicting the author of the news article. How can the model learn $p_T(\boldsymbol{Y}_T \mid \boldsymbol{X}_T)$ from $\mathcal{D}_S$? and $\mathcal{T}_S$ where either $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$? In other words, how would the English topic classifier help us predict topics for Shona articles or determine the authors of the English articles?

Prior work defines different types of transfer learning (Pan and Yang, 2010; Ruder, 2019). This thesis focuses on the two main types: inductive transfer learning and transductive transfer

learning. Inductive learning is concerned with training a model from labeled data. For **inductive transfer learning**, the source task $\mathcal{T}_S$ and target task $\mathcal{T}_T$ are different, but labeled data may be available in the target domain $\mathcal{D}_T$. Like inductive learning, the goal is to learn $p_T(\boldsymbol{Y}_T \mid \boldsymbol{X}_T)$ with the labeled data in the target domain. However, for transfer learning, we want the source model to learn this distribution. *Thus, in inductive transfer learning, the source model needs to transfer knowledge from the source task $\mathcal{T}_S$ to the target task $\mathcal{T}_T$.*

On the other hand, transductive learning involves making predictions for unseen data by considering all data (labeled and unlabeled). For **transductive transfer learning**, the source task $\mathcal{T}_S$ and target task $\mathcal{T}_T$ are the same, but labeled data is only available in the source domain $\mathcal{D}_S$. Like transductive learning, we can use unlabeled data in the target domain $\mathcal{D}_T$ to learn $p_T(\boldsymbol{Y}_T \mid \boldsymbol{X}_T)$. However, since this is transfer learning, the goal is for the source model to learn this new distribution, rather than training a new model from scratch (Arnold et al., 2007). *Therefore, transductive transfer learning uses the knowledge learned about the source domain to help make predictions for the target domain.*

In the following sections, we cover prior work in these two types of transfer learning. Section 2.2.1 is about models for inductive transfer learning, which often involve learning general representations from a source task to help transfer knowledge to the target task. Section 2.2.2 looks at methods for transductive transfer learning that help align information across domains or languages.

2.2.1 Inductive Transfer Learning

The objective of inductive transfer learning is to adapt a model, which is trained on the source task, for a target task. That is, we assume $\mathcal{Y}_S \neq \mathcal{Y}_T$, a shift in the label space. As a result, this typically implies $p_S(\boldsymbol{Y}_S) \neq p_T(\boldsymbol{Y}_T)$ and $p_S(\boldsymbol{Y}_S \mid \boldsymbol{X}_S) \neq p_T(\boldsymbol{Y}_T \mid \boldsymbol{X}_T)$, unless there is the rare chance of a one-to-one mapping between source and target labels. In many situations, the target task is challenging to learn because of scarcity in labeled data. If a related task is easier to learn, we can first train the model on the related task and then modify it for the target task. We have mentioned the example before of using a topic classifier to help predict authorship. There is likely some correlation between topics and authors, so the knowledge about topics may help with the authorship task.

Another reason for inductive transfer learning is shift in the scope of application. A situation that often occurs is refining the label set after training a classifier and realizing that the original set of classes is too broad. Then, we would need to relabel the entire dataset and train the model all over again. An example application is out-of-scope intent detection (Zhan et al., 2021). Intent detection is the task of predicting intents from utterances and out-of-scope refers to unknown intents. The model is initially trained to predict which type of billing issue is mentioned in the customer complaints. However, the company then realizes that service issues are prevalent. To avoid unhappy customers, how can the company researchers quickly adapt the model to refine the intent predictions?

The common framework for inductive transfer learning involves two stages: *pre-training* and *adaptation*. During pre-training, we train a model on the source task. For adaptation, we refine the source model for the target task. Many works for transfer learning in NLP execute this

two-stage scheme, including early works like LSA (Deerwester et al., 1990) and recent breakthroughs like BERT (Devlin et al., 2019). For LSA, the algorithm that uses SVD to learn low-dimensional vector representations of documents can be viewed as pre-training. Then, the adaptation occurs when using these representations for tasks like text classification. For BERT, the pre-training task is language modeling and the adaptation method is fine-tuning, which we will explain later in this section.

Ideally, the pre-trained model should be applied to not only one but several target tasks. Similar to representation learning (Bengio et al., 2013), the goal is to understand documents in a *universal* context. For language, we need to somehow encode semantic and syntactic features about the documents. Thus, pre-training typically requires extensive time and resources to gain general knowledge about language. While pre-training is costly, the second adaptation step tends to be quicker and less expensive. So, we only have to perform pre-training once and reuse the pre-trained model for each adaptation. The two-stage model development overall reduces resources and improves accuracy. For the rest of the section, we will focus on NLP methods that first pre-train a source model and then adapt it for target tasks.

Traditional Text Modeling Many traditional pre-training methods use statistics on word occurrences to create document representations. Thus, this eliminates the need for labeled data to pre-train these initial representations. One of the earliest works to embed documents is *latent semantic analysis* (Deerwester et al., 1990, LSA). First, the algorithm computes the term-document matrix $\boldsymbol{X}$ such that $\boldsymbol{X}_{i,j}$ is the number of times that word w_i occurs in document d_j of the training corpus. Then, we decompose $\boldsymbol{X}$ with singular value decomposition so that $\boldsymbol{X} = \boldsymbol{U}\Sigma\boldsymbol{V}^\top$. To obtain the rank k approximation, we take the largest k singular values in $\boldsymbol{\Sigma}$ to get submatrix

$\boldsymbol{\Sigma}_k$. Then, we select corresponding rows in $\boldsymbol{U}$ and $\boldsymbol{V}$ to get submatrices $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$, respectively. The k-dimensional representation for word i is $\boldsymbol{E}_i = \boldsymbol{\Sigma}_k \boldsymbol{u}_i^\top$ where $\boldsymbol{u}_i$ is the ith row of $\boldsymbol{U}_k$. The k-dimensional representation for document d_j is $\boldsymbol{D}_j = \boldsymbol{\Sigma}_k \boldsymbol{v}_j^\top$ where $\boldsymbol{v}_j$ is the jth column of $\boldsymbol{V}_k$. LSA uses linear algebra to obtain low-dimensional, dense representations of textual data.

An issue with LSA is that words similar in meaning may rarely co-occur. To handle this problem, one approach is *Brown clustering* (Brown et al., 1992) which groups words into different classes c_i. With a class-based bigram model, the conditional probability of word w_{i+1} following word w_i is computed as:

$$p(w_{i+1} \mid w_i) = p(c_{i+1} \mid c_i) p(w_{i+1} \mid c_i), \tag{2.4}$$

where $p(c_{i+1} \mid c_i)$ is the class transition probability and $p(w_{i+1} \mid c_i)$ is the word emission probability. As in n-gram models, the goal is to maximize log likelihood of sequence $w_1, \ldots, w_n$. After substituting Equation 2.4, we can show that maximizing log likelihood is equivalent to optimizing average mutual information. Therefore, Brown clustering uses hierarchical clustering to iteratively increase average mutual information by merging words into classes. The algorithm overcomes data sparsity in NLP as words similar in meaning may rarely co-occur. From the Brown word classes, we can derive word representations useful for downstream applications like part-of-speech tagging.

The downside to Brown clustering is that a word can only be assigned to one cluster. An approach that groups similar words together but does not enforce hard assignments is *topic modeling*. A topic model assumes documents are mixtures over topics and topics are distributions over words. The most popular method for building a topic model is *latent Dirichlet allocation* (Blei et al., 2003, LDA), which sets the topic-word distribution with a Dirichlet prior (Figure 2.1). LDA

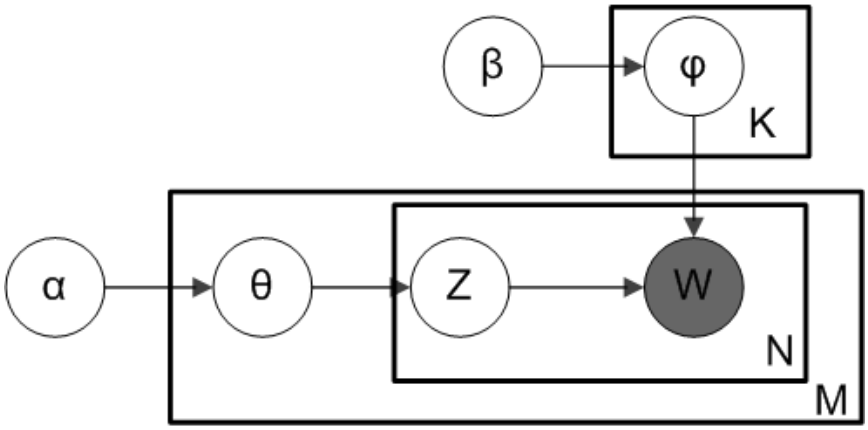


Figure 2.1: LDA topic model where word w is generated from a topic z with distribution ϕ_z, a Dirichlet prior with parameter β. Topic z is generated from θ, the document-topic Dirichlet distribution with parameter α. The generative model uses latent topics to transfer knowledge about language across different data domains.

is *generative*, meaning that it jointly models the feature space and the label space (in this case, topics). The challenge with topic models is that topics are never observed and must be *inferred*. Inference for LDA is computed with either Gibbs sampling or variational Bayes. To apply topic modeling to text classification, we typically use document-topic likelihood as features to represent documents in downstream tasks. While several works in topic modeling are generative, an *anchor-based topic model* is a computationally quick, spectral approach (Arora et al., 2012, 2013). The method depends on anchor words, which are words that appear with high probability in only one topic. In documents, we often observe anchor words like "lipstick" for a topic about cosmetics and "carburetor" for a topic about automobiles. Using co-occurrence between anchor words and other words, the algorithm can determine the distribution over words for each topic. Then, we fix the topic-word distribution and use an inference algorithm (e.g. Variational Bayes) to learn topics for each document. Chapter 3 covers the advantages of anchor-based topic models in an interactive setting.

How do we adapt these models for the target task? First, we use these methods to engineer

representations for documents or words without using labeled data. Then, these representations can be passed into a classifier for a downstream task. This type of model adaptation is an instance of *feature extraction*. By computing word likelihood and co-occurrence, we obtain features that help transfer knowledge for simple NLP tasks.

Neural Word Embeddings Traditional text modeling relies on matrix factorization and word occurrence statistics. With the advent of neural networks, research focuses on applying deep learning to compute representations for text. While LSA and LDA embed documents as vectors, *word2vec* embeds word types as vectors. For the word2vec models, Mikolov et al. (2013a) train *word embeddings* with neural networks on large, unlabeled corpus. They propose two choices of architecture: continuous bag-of-words (CBOW) and skip-gram. For each target word w_t in corpus $\mathcal{D}$, the CBOW model tries to predict w_t in a window of m context words with a loss function defined as

$$L_{\text{CBOW}} = -\frac{1}{|\mathcal{D}|}\sum_{t=1}^{|\mathcal{D}|}\log p(w_t \mid w_{t-m},\ldots,w_{t-1},w_{t+1},\ldots,w_{t+m}), \tag{2.5}$$

$$p(w_t \mid w_{t-m},\ldots,w_{t+m}) = \frac{\exp\left\{\boldsymbol{C}_t{}^\top \boldsymbol{S}_t\right\}}{\sum_{i=1}^{|\mathcal{V}|}\exp\left\{\boldsymbol{C}_i{}^\top \boldsymbol{S}_t\right\}}. \tag{2.6}$$

Here, $\boldsymbol{C}_i$ is the context embedding for word w_i, $\boldsymbol{E}_i$ is the word embedding for word w_i, and $\boldsymbol{S}_t$ is the sum of embeddings over words in the context window of target word w_t,

$$\boldsymbol{S}_t = \sum_{-m\le j\le m,\, j\neq 0} \boldsymbol{E}_{t+j}. \tag{2.7}$$

By training on the L_{CBOW} objective, the model learns $\boldsymbol{C}_i$ and $\boldsymbol{E}_i$ for each word in the vocabulary $\mathcal{V}$. On the other hand, the skip-gram model optimizes the log-likelihood of corpus $\mathcal{D}$ so that

$$L_{\text{SKIP}} = -\frac{1}{|\mathcal{D}|} \sum_{t=1}^{|\mathcal{D}|} \sum_{-m \leq j \leq m,\, j \neq 0} \log p(w_{t+j} \mid w_t), \tag{2.8}$$

$$p(w_{t+j} \mid w_t) = \frac{\exp\left\{\boldsymbol{C}_{t+j}{}^{\top} \boldsymbol{E}_t\right\}}{\sum_{i=1}^{|\mathcal{V}|} \exp\left\{\boldsymbol{C}_i{}^{\top} \boldsymbol{E}_t\right\}}. \tag{2.9}$$

The skip-gram model also learns a context representation $\boldsymbol{C}_i$ and a word representation $\boldsymbol{E}_i$ for each word w_i. For both models, the conditional likelihood $p(w_j \mid w_i)$ is computed using softmax. The denominator in this expression is expensive to compute, so Mikolov et al. (2013c) propose *negative sampling* for a faster approximation:

$$p(w_{t+j} \mid w_t) = \log \sigma(\boldsymbol{C}_{t+j}{}^{\top} \boldsymbol{E}_t) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim p_n} \left[\log \sigma(\boldsymbol{C}_i{}^{\top} \boldsymbol{E}_t)\right] \tag{2.10}$$

where k negative samples are sampled from negative distribution p_n. Levy and Goldberg (2014) discover that the skip-gram model with negative sampling is an implicit matrix factorization of the word-context matrix $\boldsymbol{C}^{\top}\boldsymbol{E}$.

After word2vec, GLoVE vectors (Pennington et al., 2014) show similar success with easier parallelization. Whereas word2vec aims to predict target word, GLoVE optimizes for co-occurrence through matrix factorization. FastText (Bojanowski et al., 2017) is a subword extension of the word2vec skip-gram model. The model trains an embedding for each character n-gram, rather than for each word. Then, the embedding for each word is the sum over the embeddings of the character n-grams. Through encoding morphological features, fastText provides

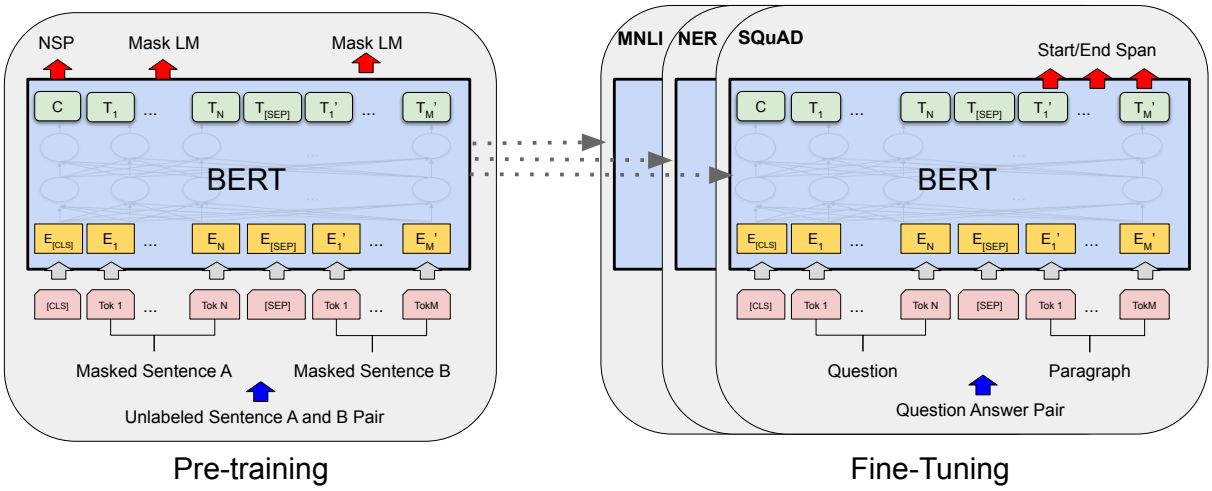


Figure 2.2: BERT (Devlin et al., 2019) is a bidirectional transformer model. The model is pre-trained with masked language modeling and next sentence prediction (left). After pre-training the model, it can be fine-tuned for many downstream tasks (right).

accurate representations for rare words in the vocabulary.

The neural architecture of these models require pre-training on large corpora. Word2vec pre-trains on a Google News corpus with six million words. GLoVE uses Wikipedia dumps, Gigaword corpus, and Common Crawl data. FastText trains on Wikipedia dumps in nine languages. These pre-trained word embeddings are used as features to represent text (Turian et al., 2010). So, similar to traditional text modeling, the way to adapt pre-trained embedding models is through feature extraction. However, unlike traditional text representations, neural word embeddings can encode distributional semantics that are not captured before.

Pre-trained Language Models Neural word embeddings are generated from shallow networks. As computing power increases with the rise of GPUs, model capacity also skyrockets. New NLP models contain more layers than before and datasets become even more massive. ELMo introduces *contextualized* word embeddings (Peters et al., 2018). Words can share different meanings (e.g. "training" a model is not exactly the same as "training" for a sport), so their representations

should not be static. ELMo uses a bidirectional LSTM and a language modeling objective for pre-training. The contextualized embedding is formed by summing over weighted and concatenated hidden states. Radford et al. (2018) introduce GPT which uses the *transformer model* as a decoder. Unlike RNNs, transformers do not have to process sequential data in order (Vaswani et al., 2017). The model relies on multi-head self-attention to learn long range-dependencies between items in a sequence. Thus, transformers can be parallelized to reduce computing time and costs.

Another innovation of these contextualized models lies in how they are adapted. *Fine-tuning* is introduced as another way to adapt pre-trained models for downstream tasks (Mou et al., 2016; Howard and Ruder, 2018). In feature extraction, weights of the pre-trained model are frozen. When adapting a model through fine-tuning, these weights are updated as the model trains on data for the target task. For these transformer models, fine-tuning can further increase accuracy over static feature extraction. In later chapters, we will see how fine-tuning improves model adaptation.

These works lead up to the seminal BERT model (Devlin et al., 2019), a bidirectional encoder-decoder transformer. The model has two pre-training loss objectives: masked language modeling and next sentence prediction (Figure 2.2). For masked language modeling, a subset of input tokens are masked, and the model has to predict masked tokens from the context. For next sentence prediction, the model has to predict whether a pair of sentences are contiguous. The pre-training data consists of the BooksCorpus and English Wikipedia. BERT reaches state-of-the-art performance in several NLP tasks, like text classification, question answering, and natural language inference.

RoBERTa improves upon BERT by removing the next sentence prediction objective and trains on larger mini-batches (Liu et al., 2019). XLNet uses autoregressive language modeling to

pre-train by maximizing log-likelihood of sequence with respect to all possible permutations of the factorization order (Yang et al., 2019).

After the success of BERT, many researchers attempt to allocate as many resources as possible during pre-training. The massive T5 transformer model contains eleven billion parameters and pre-trains on the "Colossal Clean Crawled Corpus", which is twice as large as Wikipedia (Raffel et al., 2020). OpenAI releases GPT-3, an autoregressive transformer model with 175 billion parameters (Brown et al., 2020). GPT-3 shows astounding capabilities in text generation and attempts to learn target task with as few examples as possible. People have difficulty distinguishing text generated from GPT-3 and writings from actual humans (Clark et al., 2021).

2.2.2 Transductive Transfer Learning

The goal of transductive transfer learning is to transmit information across domains. In Section 2.2, we define a domain $\mathcal{D}$ by its feature space $\mathcal{X}$ and a marginal probability distribution $p(\boldsymbol{X})$. We assume that the source model is trained on a source domain $\mathcal{D}_S$ and labeled data is limited in the target domain $\mathcal{D}_T$. So, we must somehow adapt the source model for the target domain. During domain shift, we typically observe the two following situations:

1. The marginal probability distributions of source and target domains differ, $p_S(\boldsymbol{X}_S) \neq p_T(\boldsymbol{X}_T)$. This scenario is known as *domain adaptation* (Section 2.2.2.1). For instance, the distribution of words is different between news articles and scientific reports.
2. The feature spaces of source and target domains differ, $\mathcal{X}_S \neq \mathcal{X}_T$. In this dissertation, we refer to this as *cross-lingual learning* because feature spaces are different across various languages (Section 2.2.2.2).

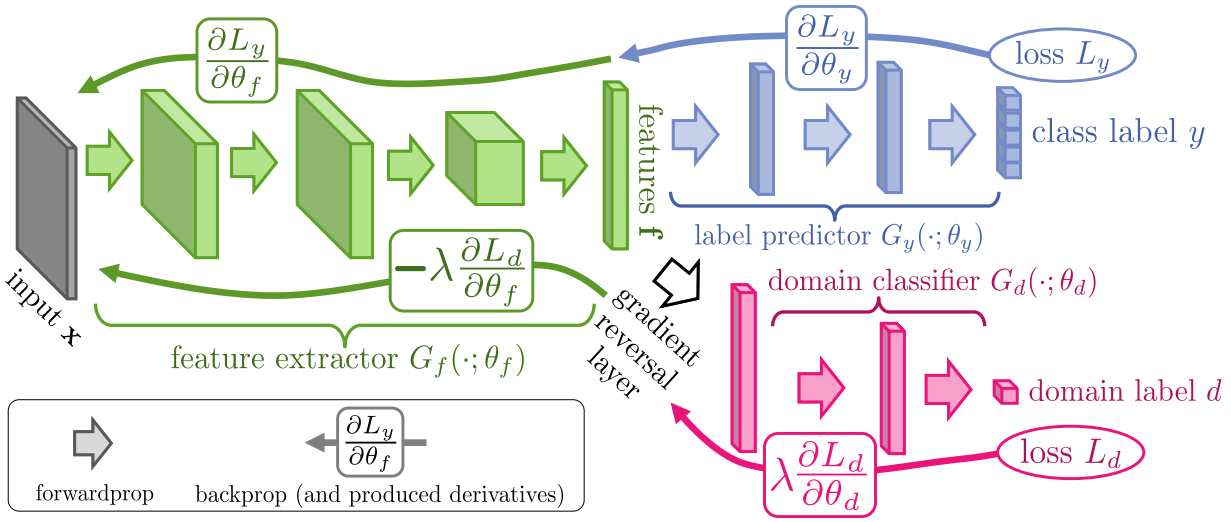


Figure 2.3: Figure of domain-adversarial training from Ganin et al. (2016) for training a feature extractor. The learned representations are passed into the label classifier and domain classifier to jointly minimize classification loss and maximize domain loss. Adversarial training has influenced many works in domain adaptation. It also shows a great amount of engineering effort spent toward transferring knowledge.

2.2.2.1 Domain Adaptation

Supervised models are less robust when they encounter shifts in the data domain. The goal of domain adaptation is to accommodate these changes in marginal distribution $p(\boldsymbol{X})$. Whereas inductive transfer learning has access to some labeled data for the target task, domain adaptation typically assumes very few or no labeled data for the target domain. An example of a problem in domain adaptation is taking a topic classifier trained on news articles and now applying it to scientific reports in the same language. The feature space $\mathcal{X}$ is the same, but the marginal probability distribution has shifted from $p_S(\boldsymbol{X}_S)$ to $p_T(\boldsymbol{X}_T)$. For scientific reports, we see more occurrences of technical terms that may rarely appear in news articles.

Distribution similarity Many works in domain adaptation focus on finding representations that minimize the divergence between source and target marginal distributions, $p_S(\boldsymbol{X}_S)$ and $p_T(\boldsymbol{X}_T)$.

The high-level idea is that increasing similarity between the feature distributions will help the model detect common characteristics between the domains. For example, if we consider news articles and scientific reports, we want to emphasize terms that appear in both domains. That way, we can use these terms to transfer knowledge from one domain to another. If COVID-19 frequently occurs in news articles labeled as "medicine", then the scientific report that also discusses COVID-19 should be labeled as "medicine" too. There are different ways to measure divergence between two probability distributions, such as Jensen-Shannon divergence and Kullback-Leibler divergence. Ben-David et al. (2007) propose using the $\mathcal{A}$ distance Kifer et al. (2004) to measure divergence,

$$d_{\mathcal{A}}(p_S, p_T) = 2 \sup_{A \in \mathcal{A}} p_S(A) - p_T(A) \tag{2.11}$$

where $\mathcal{A}$ is the collection of all subsets of feature space $\mathcal{X}$. In other words, the $\mathcal{A}$ distance measures the largest possible change between p_S and p_T over a data subset of instances from $\mathcal{X}$. The $\mathcal{A}$ distance is also the minimum empirical risk of a classifier that is supposed to predict whether instances are from the source or target domain. For ease of computation, we use a variant called proxy $\mathcal{A}$ distance to measure divergence between p_S and p_T. Blitzer et al. (2007) apply the proxy $\mathcal{A}$ distance to sentiment classification. Another measure for comparing probability distributions is maximum mean discrepancy,

$$MMD(p_S, p_T) = \left\| \mathbb{E}_{\boldsymbol{x} \sim p_S} \left[f(\boldsymbol{x}) \right] - \mathbb{E}_{\boldsymbol{x} \sim p_T} \left[f(\boldsymbol{x}) \right] \right\|_{\mathcal{H}} \tag{2.12}$$

where f maps $\boldsymbol{x}$ to a reproducing kernel Hilbert space. The function f can be a kernel or neural network. So, maximum mean discrepancy measures the difference between means of p_S and p_T.

This distance function is also used in domain adaptation (Chen et al., 2009; Pan et al., 2010).

Domain-adversarial training Methods for increasing distributional similarity involve statistical measures. A more recent and common framework for domain adaptation is *domain-adversarial training* (Ganin et al., 2016). Here, neural networks are used to train representations for the source and target domain. Inspired by the proxy $\mathcal{A}$ distance, domain-adversarial training seeks to confuse a domain classifier. While training for a classification task, the model has to simultaneously optimize the adversarial loss (Figure 2.3). Therefore, the model trains representations that are both label-discriminative and domain-agnostic. In the previous example, domain-adversarial training would aim to train representations such that the domain classifier cannot distinguish a news article from a scientific report. At the same time, these representations should be obvious enough so that the label predictor still accurately predicts the topic of the news articles.

Domain-adversarial training has been employed in NLP tasks, like relation extraction (Fu et al., 2017) and duplicate question detection (Shah et al., 2018). Chen et al. (2018b) apply adversarial training in cross-lingual sentiment classification to learn language-agnostic representations. This is a crossover between domain adaptation and cross-lingual learning, another problem in transductive transfer learning.

Recently, researchers have scrutinized domain-adversarial training and discover that other self-supervised techniques are more effective for domain adaptation. Wang et al. (2020) notice that to continue training BERT with domain-adversarial layers does not improve accuracy. Simply fine-tuning BERT in the usual fashion already achieves similar results across domains. Furthermore, domain-adversarial training is expensive and unstable to use with transformer models. Karouzos et al. (2021) confirm the instability of domain-adversarial training for large language

models. They propose a new domain adaptation method where the source model simultaneously trains on the target domain with language modeling loss and source domain with the task-specific (e.g. text classification) loss. Shen et al. (2022) theoretically prove that domain-invariant features are unnecessary for domain adaptation and show benefits of contrastive pre-training.

Domain adaptation is an important problem that has received much attention over the years. Pre-training seems to accomplish similar results that past techniques, like domain-adversarial training, has achieved. Thus, future work may focus more on pre-training techniques for improved domain adaptation. In the next section, we discuss adapting models across languages. The challenge is that the languages may share few words in common, which makes it difficult to transfer knowledge.

2.2.2.2 Cross-lingual Learning

Given the linguistic diversity in the world, building AI systems for multiple languages is a critical goal in NLP. The issue is that many languages lack data needed to train robust machine learning models. In this setting, there is at least one language with plenty of data. The high-resource language is the *source language* and the low-resource language is the *target language*. An appropriate choice of source language could be one with lots of data and related to the target language (Lin et al., 2019). The source language has vocabulary $\mathcal{V}_S$ and the target language has vocabulary $\mathcal{V}_T$. The goal is to train models on the source language and then adapt them for the target language. Aside from labeled data, we may also have other resources such as a dictionary or parallel sentences. Therefore, we apply monolingual models from inductive transfer learning (Section 2.2.1) in the cross-lingual setting through word-level, document-level, or topic-level

alignment. For the rest of the section, we talk about cross-lingual alignment for three models: topic modeling, word embeddings, and transformer models.

Multilingual topic modeling Topic modeling is a fully unsupervised method (Section 2.2.1), so many early works extend these models for languages with no labeled data. The Polylingual Topic Model, PLTM, is a cross-lingual version of LDA where there is a set of topics for each language (Mimno et al., 2009). PLTM requires a one-to-one alignment between documents of different languages such that the documents are *comparable*. The definition of comparable documents is a pair of texts that cover similar content (McEnery and Wilson, 2003). For example, today's articles from the New York Times and Beijing Daily are not parallel translations of each another. They are comparable because they broadly cover the same set of topics. JointLDA is another cross-lingual extension of LDA that does not require aligned documents (Jagarlamudi and Daumé, 2010). However, a bilingual dictionary is needed to align topics across languages. Interestingly, the multilingual model of JointLDA has less perplexity than the monolingual LDA models for each language. Therefore, modeling cross-lingual topics helps transfer information between source and transfer languages. Hu et al. (2014b) introduce a polylingual tree-based topic model to improve machine translation with domain knowledge from both source and target languages.

Cross-lingual word embeddings Compared to topic-level alignment, cross-lingual research more commonly uses word-level alignment. By projecting words of different languages into one vector space, cross-lingual word embeddings help transfer information across languages. Rapp (1999) propose one of the first cross-lingual word embedding models that depend on dic-

tionary entries and co-occurrence patterns. Following the success of neural word embeddings (Section 2.2.1) in English, Mikolov et al. (2013b) use word2vec models to train monolingual models and learn a translation matrix with dictionary entries that can map vectors from source to target languages. Specifically, they learn translation matrix $\boldsymbol{W}$ such that the following loss is minimized,

$$L_{\text{MSE}} = \sum_{i=1}^{n} \left\| \boldsymbol{W}\boldsymbol{E}_i^S - \boldsymbol{E}_i^T \right\|^2 \tag{2.13}$$

where $\boldsymbol{E}_i^S$ is the word embedding for source word w_i and $\boldsymbol{E}_i^T$ is the word embedding of its translation in the target language. Many models for cross-lingual word embeddings follow this sort of algorithm. Then, Xing et al. (2015) place orthogonality constraints on $\boldsymbol{W}$ so that all embeddings are unit length. That way, the cross-lingual word embeddings can be compared with dot product for downstream tasks like machine translation.

The downside to these approaches is the reliance on dictionary entries. Vulić and Korhonen (2016) note that high-quality cross-lingual embeddings are hard to train for languages that have scarce or inaccurate English translations. Artetxe et al. (2017) develop a method that learns cross-lingual word embeddings with as few translations as possible. They use a self-learning approach that iteratively learns the mapping and more dictionary entries. The initial dictionary can be as few as twenty-five entries that are easy to translate.

Other ways to learn cross-lingual mapping include CCA (Faruqui and Dyer, 2014) and *retrofitting* (Faruqui et al., 2015). Retrofitting approaches treat dictionary entries as edges connecting word nodes and learn parameters to optimize over this graph. Ettinger et al. (2016) generalize the approach to retrofit word alignments from parallel corpora. Thus, retrofitting does not learn a translation matrix $\boldsymbol{W}$ and instead trains on individual constraints for each word. In

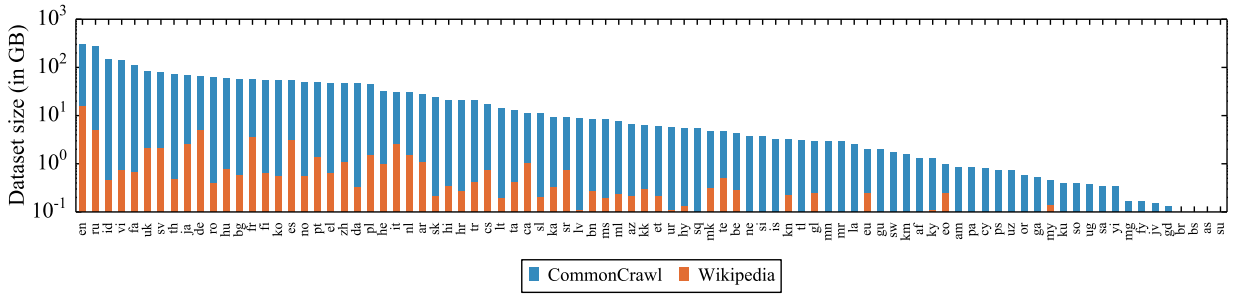


Figure 2.4: The size of pre-training data (GiB) across eighty-eight languages for multilingual transformers (Conneau et al., 2020). The MBERT and XLM models are pre-trained on the Wiki-100 corpus (orange). XLM-R trains on the CC-100 dataset (blue), which is much larger and covers more languages than Wiki-100. Extremely vast amounts of data are already used to develop large multilingual models, so further adaptation should involve cheaper methods.

the Attract-Repel algorithm, antonymy constraints are also used for retrofitting word embeddings (Mrkšić et al., 2017). An issue with retrofitting is that it never changes words that are unobserved in the set of constraints. So, explicit retrofitting uses an end-to-end neural network to refine all word embeddings (Glavaš and Vulić, 2018). Chapter 5 explores retrofitting word embeddings from linguistic constraints given by users.

Multilingual transformers Recent development in cross-lingual learning rely on transformer models, which have astounding capability in inductive transfer learning (Section 2.2.1). Like in the monolingual case, multilingual transformer models have the advantage of contextualization over the static, cross-lingual word embeddings. While releasing BERT, Devlin et al. (2019) additionally provide a multilingual variant called MBERT. Wu and Dredze (2019) find that MBERT is stronger than cross-lingual word embeddings for zero-shot cross-lingual tasks like natural language inference, document classification, and dependency parsing. Thus, efforts in multilingual NLP shift toward building cross-lingual transformer models.

XLM improves cross-lingual pre-training by adding a translation language modeling objective (Lample and Conneau, 2019). First, it takes in parallel sentences as one input. Like masked

language modeling, it masks the words in the parallel sentences for the model to predict. XLM-R (Conneau et al., 2020) makes further improvements on XLM by using RoBERTa rather than BERT and increasing size of training data (Figure 2.4). From experiments, they vary the number of pre-training languages and notice that accuracy for low-resource languages increases until a certain point. This is known as the *curse of multilinguality*. Their solution is to increase model capacity by increasing hidden size. Pfeiffer et al. (2022) propose an alternative way without scaling the number of model parameters. Their XMOD model uses modular components during pre-training so that information about each individual language is preserved. While XLM-R and XMOD show state-of-the-art performance on low-resource languages, training these models are computationally expensive. Future work in cross-lingual learning should focus on reducing additional costs to further adapt these models for specific domains.

This section has introduced a variety of problems in transfer learning. The current solutions for transfer learning are quite costly. The trend is to pre-train the model on as much information as possible so that it can be easily adapted in different situations. In the next section, we begin to discuss prior work on interactive learning in NLP. The background work may inspire us to develop interactive methods in transfer learning that are cheaper and more effective.

2.3 Interactive Learning

Machine learning practitioners understand the theory and implementation of AI systems, but external users may not understand and trust machines. If we want people to use AI, then we need to make machine learning *interpretable* (Swartout, 1983). As Biran and Cotton (2017) loosely define, "systems are interpretable if their operations can be understood by a human, either

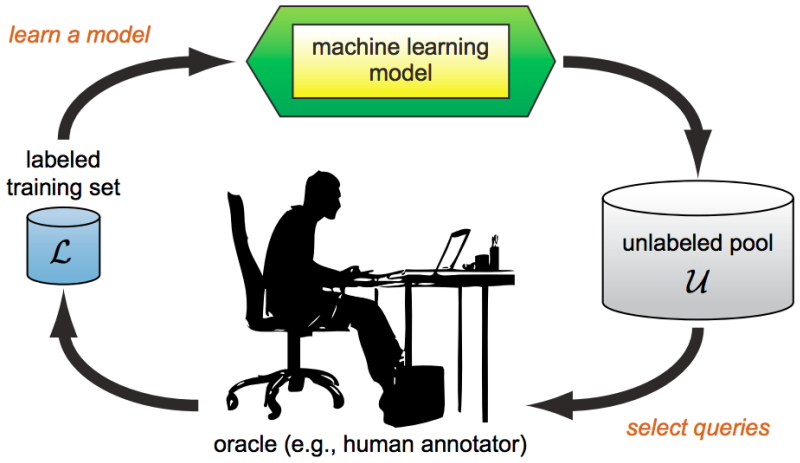


Figure 2.5: Figure from (Settles, 2009) showing the active learning cycle. We repeatedly select examples in the unlabeled data pool, have an oracle annotate selected examples, add these examples to the training set, and train the model on the new dataset.

through introspection or through a produced explanation". If people can justify model outputs, then they are more inclined to deploy AI in real-life situations. So, human-centric AI is primarily motivated by the need for transparency and control (Renner, 2020). Through human-AI collaboration, the human can refine the model for downstream tasks and particular needs (Feng and Boyd-Graber, 2019). This kind of interaction is necessary to properly deploy machine learning systems in the real world.

This dissertation wants to take a step further and understand whether transfer learning can benefit from interactive machine learning (Section 1.1). Here, we define *interactive machine learning* as the setting in which the machine learns through training on data and interacting with other agents. The rest of the section focuses on prior work in interactive machine learning. First, we look at *active learning* where the goal is to sample data that can most effectively train a model (Section 2.3.1). We assume the existence of an oracle that can label any data. Then, we discuss recent work in *human-in-the-loop* NLP (Section 2.3.2). These works motivate later chapters that apply human-AI interaction to transfer learning.

2.3.1 Active Learning

In the typical machine learning framework, the model trains on labeled data given by the user. Here, the model is a *passive learner* because it accepts any kind of training data. An *active learner* is a model that actively queries labels of training examples from a large, unlabeled data pool. We assume that an oracle, like a human annotator, can provide labels for the queried examples. The objective is to reach higher accuracy after training the model with this small sample of labeled data. Thus, active learning strives to reduce annotation cost and time (Figure 2.5).

Active learning is well-studied empirically and theoretically, especially in NLP (Settles, 2009). However, these observations and analyses only apply to linear models. Widely-used methods, like uncertainty sampling (Lewis and Gale, 1994), may not work as well for neural networks (Lowell et al., 2019). Recent work in active learning attempt to adapt active learning for deep learning. The rest of the section looks at traditional active learning algorithms (Section 2.3.1.1) and recent work in deep active learning (Section 2.3.1.2).

2.3.1.1 Traditional Active Learning

One of the earliest works in active learning is *membership query synthesis* (Angluin, 1988). In this framework, each query is any point x in the feature space $\mathcal{X}$ and an oracle can provide label y for example $\boldsymbol{x}$. Lewis and Gale (1994) argue that membership query synthesis is infeasible because AI cannot generate (at the time) comprehensible text for humans to annotate. Instead, they limit data sampling to a large, unlabeled data pool $\mathcal{U}$. Additionally, they assume that there is already a small, labeled dataset $\mathcal{L}$ to start with. They introduce *uncertainty sampling*, a framework that chooses the most uncertain example $\boldsymbol{x}^*$ for the active learner to learn. The paper focuses on

binary text classification with a naïve Bayes classifier. Instance $\boldsymbol{x}^*$ is chosen if $p(y \mid \boldsymbol{x}^*)$ is closest to 0.5 than other examples in the data pool. Later on, this method is generalized for multi-label classification as *maximum entropy sampling* that samples the example with the largest predictive entropy:

$$\boldsymbol{x}^*_{\text{ENT}} = \arg\max_{\boldsymbol{x} \in \mathcal{U}} -\sum_{y \in \mathcal{Y}} p(y \mid \boldsymbol{x}) \log p(y \mid \boldsymbol{x}), \tag{2.14}$$

where y is any label in label space $\mathcal{Y}$.

Maximum entropy sampling is arguably the most popular active learning strategy. As long as the model provides estimate for $p(y \mid \boldsymbol{x})$, then it is straightforward to compute uncertainty for each example. There are many other ways to compute uncertainty of an example. For instance, the most uncertain example can be the one that the model is least confident about. Least confidence sampling (Culotta and McCallum, 2005) chooses the example such that:

$$\boldsymbol{x}^*_{\text{LC}} = \arg\min_{\boldsymbol{x} \in \mathcal{U}} p(\arg\max_{y \in \mathcal{Y}} p(y \mid \boldsymbol{x}) \mid \boldsymbol{x}). \tag{2.15}$$

They apply least confidence sampling to conditional random fields for structured prediction tasks. What if the classifier is not probabilistic? For support vector machines, Tong and Koller (2001) propose measuring uncertainty as the example's distance to the maximum-margin hyperplane. Thus, the concept of uncertainty can be extended to other types of machine learning models.

Active learning also samples the example that can possibly change the model the most if its label is known. *Expected gradient length* uses classification loss gradient to compute expected model change because machine learning models are typically optimized with the loss gradient (Settles et al., 2008b). Suppose that $f_{\mathcal{L}}$ is the model trained on a labeled dataset $\mathcal{L}$. The

algorithm chooses an instance such that

$$\boldsymbol{x}^*_{\text{EGL}} = \arg\max_{\boldsymbol{x}\in\mathcal{U}} \sum_{y\in\mathcal{Y}} p(y \mid \boldsymbol{x}) \left\| \nabla L(f_{\mathcal{L}\cup\langle\boldsymbol{x},y\rangle}(\boldsymbol{x}), y) \right\|, \tag{2.16}$$

where L is the classification loss and $f_{\mathcal{L}\cup\langle\boldsymbol{x},y\rangle}$ is the model trained on the union of $\mathcal{L}$ and training example $\langle\boldsymbol{x}, y\rangle$. Thus, the sampled example should impact the model parameters the most and thereby reduce classification loss through training. Another similar method is *expected error reduction* which samples the instance that can reduce training error the most (Roy and McCallum, 2001). For both expected gradient length and expected error reduction, computation takes a long time. Thus, these algorithms are not as well-known as maximum entropy sampling.

A disadvantage of the mentioned active learning algorithms is that they tend to select outliers. The example that confuses the model the most might be the one that rarely occurs. So, *diversity sampling* is a family of active learning strategies that counteract this issue. The goal is to pick examples that are diverse in feature distribution or representation. Diversity sampling is also known as representative sampling because the sampled examples should represent the unlabeled data pool. Xu et al. (2003) develop a representative sampling method for support vector machines, which are models that optimize the hinge loss function. First, they train the model on already labeled data $\mathcal{L}$. Then, they find a subset of the unlabeled data pool $\mathcal{U}$ that lies in the margin. Finally, they apply k-MEANS clustering to this subset and sample examples that are closest to the k-MEANS centers. Other works in representative sampling also apply clustering techniques to diversify active learning samples. Hu et al. (2010) use non-deterministic clustering methods to improve cluster robustness. Bodó et al. (2011) use spectral clustering to sample data for support vector machines.

These methods are primarily for models like naïve Bayes and support vector machines. Recently, the research in active learning has shifted its focus toward developing sampling strategies for deep learning. In the next section, we discuss novel active learning algorithms for neural models.

2.3.1.2 Deep Active Learning

Neural networks have shown state-of-the-art performance in several fields, including NLP (Section 2.2). However, the models are computationally expensive and require a substantial amount of training data. Active learning should mitigate the expensive costs of training neural models. For deep learning, active learning is challenging for multiple reasons. First, deep models are less interpretable (Feng et al., 2018), which makes it difficult to estimate how different examples influence the model. Second, methods, like uncertainty sampling, depend on confidence scores $p(y \mid \boldsymbol{x})$ but neural networks are poorly calibrated (Guo et al., 2017). Finally, many early frameworks for active learning are *serial* because the active learning strategy selects one example on each iteration for the model to train. Neural networks are trained with mini-batches of data, so active learning should instead select batches of data.

Wang and Shang (2014) adapts prior active learning strategies to deep learning. Zhang et al. (2017) propose the first active learning strategy for neural text classification that uses expected gradient length (Settles et al., 2008b) to select sentences that contain words with the most label-discriminative embeddings. Alongside text classification, active learning has been applied to neural models for semantic parsing (Duong et al., 2018), named entity recognition (Shen et al., 2018), and machine translation (Liu et al., 2018).

Impressive breakthroughs in active learning have been applied on image datasets. Sener and Savarese (2018) develop CORESET, a representative sampling strategy for convolutional neural networks. The method is inspired by coresets (Agarwal et al., 2005), which is the concept in computational geometry that large datasets can be represented by a small subset of points. To sample a coreset, the CORESET algorithm selects examples that can solve the k-center problem (Wolf, 2011). Another innovation in deep active learning is the BADGE algorithm (Ash et al., 2020). The algorithm combines uncertainty and diversity sampling by conducting k-MEANS clustering in a hallucinated gradient embedding space. Across different models, BADGE shows improvement in accuracy against many prior works in active learning.

Bayesian active learning is another family of methods that use Bayesian neural networks to capture *epistemic uncertainty*, the uncertainty over model parameters (Gal et al., 2017). Traditional uncertainty algorithms only capture *aleatoric uncertainty*, which is uncertainty in knowledge or data, because they sample data for deterministic models. Siddhant and Lipton (2018) run a large-scale study that shows potential of Bayesian active learning in NLP. However, with recent transformer models, Bayesian approaches may not be feasible. Applying reinforcement learning to active learning also demonstrates success in cross-lingual NLP (Fang et al., 2017; Vu et al., 2019). Since labeled data is abundant in the source language across different tasks, then reinforcement learning can help learn an active learning policy for the source language than can be applied to the target languages.

Most active learning work focus on sampling data for improving the source model on the source domain. In this dissertation, we will discuss how active learning can help transfer the source model to a target domain or task. Chapter 4 looks at using active learning for inductive transfer learning, while Chapter 6 focuses on active learning for transductive transfer learning.

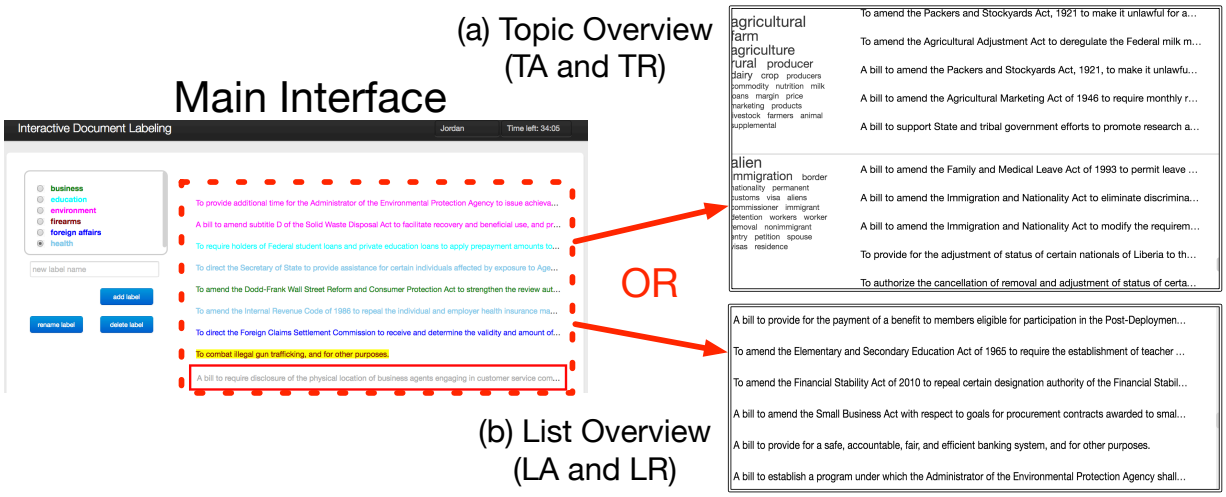


Figure 2.6: The ALTO (Poursabzi-Sangdeh et al., 2016) interface that presents documents in topic groups or sequential list order.

2.3.2 Human-in-the-loop NLP

Active learning is one possible framework for interactive learning. The area is well-studied but limited to instance-level annotation. More information could be needed for more challenging tasks. Aside from active learning, there are other ways for machines to learn from human feedback. Settles (2011) build DUALIST, a NLP interface that enables users to both assign documents and words to the classes. For example, the user label the "baseball" class for baseball-related documents and words like umpire, innings, and pitcher. DUALIST enhances traditional active learning through selecting words most relevant to downstream task. User study experiments show success on word sense disambiguation, sentiment analysis, and information extraction.

Interaction is natural to include in topic modeling because topic models are more interpretable to humans than other NLP models. Choo et al. (2013) innovate one of the first interactive topic modeling systems called UTOPIAN. Hu et al. (2014a) use human-AI interaction to improve LDA topic models with user feedback on word correlation. The ALTO interface integrates ac-

tive learning with topic modeling (Poursabzi-Sangdeh et al., 2016). Topic modeling provides a general overview to help users label documents quickly and accurately (Figure 2.6). Lund et al. (2017) use anchor-based topic models to reduce computing time for fast, interactive updates.

Feedback from non-experts can also improve other NLP tasks. For instance, human users can fix mistakes of a natural language parser (He et al., 2016). Users can write difficult questions to stump question-answering systems (Wallace et al., 2019). These user-generated questions are adversarial examples that can strengthen question-answering models. Gao et al. (2018) propose APRIL, a human-in-the-loop framework that combines active learning and reinforcement learning for document summarization. APRIL learns to summarize from user preferences without needing any reference summaries. For entity linking, humans can disambiguate entity mentions in low-resource settings (Klie et al., 2020).

These interactive learning algorithms and interfaces are innovative ways to bolster NLP models with human knowledge (Wang et al., 2021). However, these works tend to focus on user-centric goals like interpretability, explainability, transparency, and control (Section 2.3). How does the model also benefit from user-AI interaction then? We have discussed some shortcomings in transfer learning where labeled data from the target task or domain is still needed to fully adapt the model (Section 2.2). In this scenario, the user can provide input to help transfer knowledge from source to target settings. In the following chapters, we develop interactive approaches for transfer learning and understand its effects on model adaptation.

Part I

Interactive Feedback for Inductive Transfer Learning

Chapter 3: From Topic Modeling to Text Classification[1]

In NLP, topic modeling is widely used to quickly understand large text collections. The structure of topic models is interpretable and simple: documents are mixtures of topics and topics are distributions over words. Analysts can easily use topic models to preview large text collections. Due to its ease of use, topic models can become interactive for people to improve and find topics (Section 2.3.2).

In this chapter, we apply interactive topic modeling to inductive transfer learning. The setting is inductive because topics act as features for text classification. However, there is also transductive transfer involved as users strive to create a cross-lingual topic model. So, we investigate the effect of interactive topic modeling on both kinds of transfer learning settings.

To handle the transfer across languages, we propose a multilingual anchoring algorithm, which is an extension to anchor-based topic inference for comparable corpora (Chapter 2.2.1). In addition, we introduce MTAnchor, a human-in-the-loop system that uses multilingual anchoring to align topics and enables users to make further adjustments to the model.[2] Through interaction, the model produces *interpretable*, low-dimensional representations of documents. The topic model generates coherent topic alignments for comparable corpora because users them-

[1] Michelle Yuan, Benjamin Van Durme, and Jordan Boyd-Graber. 2018. Multilingual anchoring: Interactive topic modeling and alignment across languages. In *Proceedings of Advances in Neural Information Processing Systems*

[2] `http://github.com/forest-snow/mtanchor_demo`.

selves align topics. Furthermore, these vector representations improve knowledge transfer for text classification.

3.1 Anchor-based Topic Models

In Section 2.2.1, we discuss how topic models like LDA are typically probabilistic by modeling topics as a distribution over words and documents as a mixture of topics, which is sampled from a Dirichlet distribution (Figure 2.1). However, the runtime for LDA is slow and precludes interactive updates. A computationally attractive way to model topics is the anchor word algorithm. The algorithm uses the row-normalized word co-occurrence matrix $\bar{\boldsymbol{Q}}$, where $\bar{\boldsymbol{Q}}_{i,j} = p(w_2 = j \mid w_1 = i)$. The vector $\bar{\boldsymbol{Q}}_i$ is the i^{th} row of $\bar{\boldsymbol{Q}}$ and represents the conditional distribution of words in a document given that word i has occurred. Anchor word a appears with high probability in only one topic, so $\bar{\boldsymbol{Q}}_a$ resembles a topic's word distribution in topic models like LDA. For example, if "concealer" is an anchor word for a cosmetics topic, then its conditional distribution will have high probability for cosmetics-related words and low probability for other words. Still, these are not the distributions that typically define probabilistic topic models: the probability of a word given a topic.

3.1.1 Anchoring

To discover topic distributions, anchor word approaches (Arora et al., 2013) search for coefficients that describe non-anchor words' document contexts with anchor words' conditional distributions. The word "liner" has meanings that are explained by "album" in a music topic, "concealer" in a cosmetics topic, and "carburetor" in an automotive topic. Then, the conditional

distribution of "liner" can be expressed as a convex combination of the conditional distributions of "album", "concealer", and "carburetor". Given anchor words $a_1, \ldots, a_K$, the conditional distribution of word i can be approximated as

$$\bar{\boldsymbol{Q}}_i \approx \sum_{k=1}^{K} \boldsymbol{C}_{i,k} \bar{\boldsymbol{Q}}_{a_k} \quad \text{subject to} \sum_{k=1}^{K} \boldsymbol{C}_{i,k} = 1 \text{ and } \boldsymbol{C}_{i,k} \geq 0. \tag{3.1}$$

The coefficient $\boldsymbol{C}_{i,k}$ represents $p(z = k \mid w = i)$, the probability of topic k given a word i. These coefficients are recovered using the RecoverL2 algorithm (Arora et al., 2013), which minimizes the quadratic loss between $\bar{\boldsymbol{Q}}_i$ and $\sum_{k=1}^{K} \boldsymbol{C}_{i,k} \bar{\boldsymbol{Q}}_{a_k}$. Using Bayes' rule, we can obtain the standard topic matrix $\boldsymbol{A}$,

$$\boldsymbol{A}_{i,k} = p(w = i \mid z = k) \propto p(z = k \mid w = i) p(w = i) = \boldsymbol{C}_{i,k} \sum_{j=1}^{V} \bar{\boldsymbol{Q}}_{i,j}. \tag{3.2}$$

For a large vocabulary size V, finding these anchor words is a challenge, but understanding the geometric intuition behind the anchoring algorithm can help us select the right words. Points inside a convex hull are expressed as the convex combination of their vertices. If we want to approximate $\bar{\boldsymbol{Q}}_i$ as the convex combination of $\bar{\boldsymbol{Q}}_{a_1}, \ldots, \bar{\boldsymbol{Q}}_{a_K}$ (Equation 3.1), then $\bar{\boldsymbol{Q}}_{a_1}, \ldots, \bar{\boldsymbol{Q}}_{a_K}$ should be the vertices of the convex hull of $\bar{\boldsymbol{Q}}$. However, finding the vertices to a V-dimensional convex hull is time-consuming (Arora et al., 2012). Instead, Arora et al. (2013) use FastAnchorWords, a greedy approach similar to Gram-Schmidt orthogonalization, to construct an approximate convex hull of $\bar{\boldsymbol{Q}}$ and expand it as much as possible with each choice of anchor word. Other methods include projecting $\bar{\boldsymbol{Q}}$ to a low-dimensional space and finding the vertices of its exact convex hull (Lee and Mimno, 2014), adding another dimension to capture metadata (Nguyen

et al., 2015), or finding nonparametric anchor words (Yurochkin et al., 2017).

3.1.2 Multiword Anchoring

Finding topics in anchor-based models is fast, so it can be used in an interactive setting where users iteratively choose anchor words for every topic. Nevertheless, users may want to choose multiple anchor words for a topic, such as selecting both "concealer" and "lipstick" for a cosmetics topic. Therefore, Lund et al. (2017) propose multiword anchoring: users select a set $\mathcal{G}_k$ of multiple anchor words for topic k. After users select $\mathcal{G}_1, \ldots, \mathcal{G}_K$, $\bar{\boldsymbol{Q}}$ is augmented so that new rows $\bar{\boldsymbol{Q}}_{V+1}, \ldots, \bar{\boldsymbol{Q}}_{V+K}$ represent these pseudo-anchors in the conditional word co-occurrence space. These vectors $\bar{\boldsymbol{Q}}_{V+k}$ are constructed as,

$$\bar{\boldsymbol{Q}}_{V+k,j} = \left(\frac{\sum_{i \in \mathcal{G}_k} \bar{\boldsymbol{Q}}_{i,j}^{-1}}{|\mathcal{G}_k|} \right)^{-1} . \tag{3.3}$$

The motivation for using the harmonic mean (Equation 3.3) is that the function can centralize input values and ignore large outliers. Finding topics follows the same algorithm as before using single word anchors. Instead of modeling $\bar{\boldsymbol{Q}}_i$ as the convex combination of $\bar{\boldsymbol{Q}}_{a_1}, \ldots, \bar{\boldsymbol{Q}}_{a_K}$, a convex combination of $\bar{\boldsymbol{Q}}_{V+1}, \ldots, \bar{\boldsymbol{Q}}_{V+K}$ models $\bar{\boldsymbol{Q}}_i$ with minimal quadratic loss.

3.2 Bridging Languages: How Do You Say Anchor in Chinese?

Anchor-based topic models are well-defined for individual languages, but a multilingual model requires topics that are thematically connected across languages. Discovering two separate sets of anchor words does not suffice. In this section, we propose multilingual anchoring as an

algorithm to cross-lingually link topics and their corresponding anchor words.

First, we can connect anchor words across languages as *anchor links*. For example, "anchor" may be linked to "錨(máo)" in Chinese under a nautical context. After anchor words are linked, all words in the same topic across languages will be form a coherent multilingual topic. A straightforward way to link words across languages is through a *dictionary*, much as a human would. Just as possessing a Chinese dictionary does not enable someone to speak Chinese, a dictionary does not magically create multilingual topics. To construct an overall coherent model, anchor links should be carefully selected.

We define these links in more detail. A vocabulary $\mathcal{V}$ is a set of word types w. A bilingual dictionary $\mathcal{B}$ is a subset of the Cartesian product $\mathcal{V}_S \times \mathcal{V}_T$, where $\mathcal{V}_S, \mathcal{V}_T$ are vocabularies of two different languages. An element (w_S, w_T) of $\mathcal{B}$ represents a dictionary entry where words $w_S \in \mathcal{V}_S$ and $w_T \in \mathcal{V}_T$ are translations of each other. While $\mathcal{B}$ is a binary relation, it is not necessarily a function. Other multilingual topic models require that the dictionary is a one-to-one correspondence (Jagarlamudi and Daumé, 2010; Boyd-Graber and Blei, 2009; Gutiérrez et al., 2016). We relax this restriction on $\mathcal{B}$ to extract as much information from the dictionary as possible.

We could select anchor words $a_1, \ldots, a_K$ *independently* for each language by considering all words $w_S \in \mathcal{V}_S$ and $w_T \in \mathcal{V}_T$ as possible candidates for anchors (e.g., independent runs of anchor algorithm). Instead, we want to *jointly* choose anchor words for both languages. First, we use dictionary entries to create *links* between words. Then, we choose anchor words a_k^S for language S and a_k^T for language T such that a_k^S and a_k^T are linked. Through this process, we obtain a set of K anchor words for each language and can obtain topics using RecoverL2 (Arora et al., 2013).

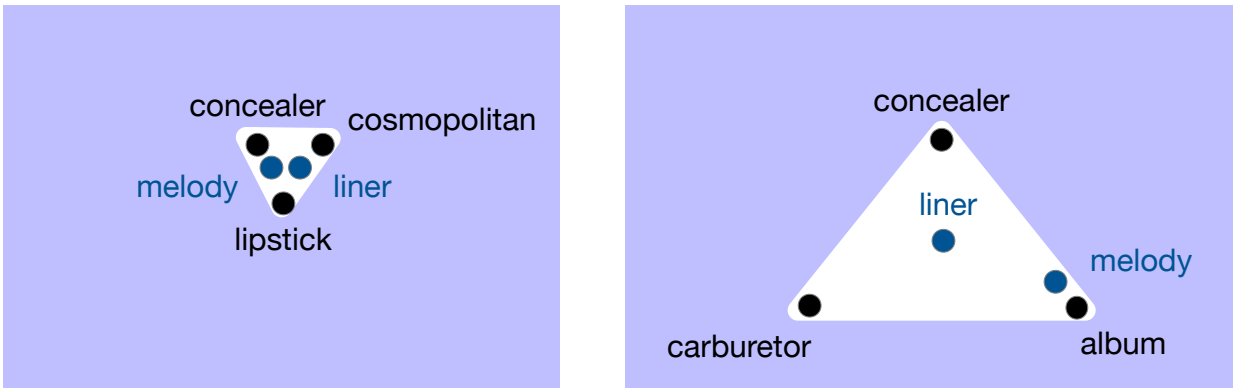


Figure 3.1: Visualizing the importance of choice in anchor words for approximating conditional distributions. The chosen anchor words are the black dots and their span is the white triangle. On the left, the span of anchor words is small, so the words "melody" and "liner" are too close together. On the right, the span of anchor words is large, so the conditional distributions of words "melody" and "liner" are approximated more accurately.

3.2.1 Multilingual Anchoring

If there is only one anchor word for each topic, our goal of building a coherent multilingual topic model would fail. Any imperfection in the dictionary would scupper the topic model. Fortunately, Arora et al. (2013) assert that there exist many anchor word choices for a topic. Even if we reduce the pool for candidate anchors, we can still find suitable anchor words for each topic. Recall that anchor words are the vertices to the convex hull of words in the conditional distribution space (Section 3.1). Finding the actual vertices of the convex hulls is too expensive, so FastAnchorWords searches for a set of anchors with maximal span. This span should approximate the convex hull of $\bar{\boldsymbol{Q}}$. Without a large enough span, we can never find accurate approximations for words in the conditional distribution space. All words w will have indistinguishable conditional distributions (Figure 3.1). As a result, every topic will have indistinct word distributions and the resulting topics will be copies of one another.

To maximize span of anchor words, FastAnchorWords (Arora et al., 2013) chooses anchor

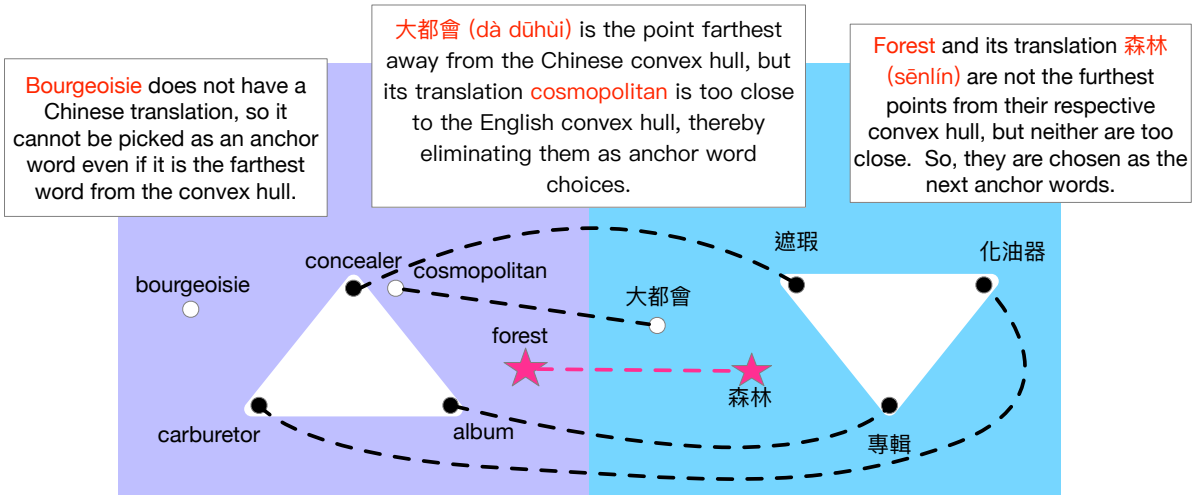


Figure 3.2: Selecting anchor links for multilingual anchoring. The purple (blue) area represents the conditional distribution space of words in the English (Chinese) corpus. The white triangle designates the space spanned by chosen anchor words. Dashed lines depict anchor links across spaces. Black points denote words already chosen as anchors, white points are unchosen words, and pink stars are most optimal anchors for the current iteration. Multilingual anchors should maximize area spanned by white triangles in both spaces.

word a_k such that

$$a_k = \arg\max_w \; d\left(\text{span}\left(\bar{\boldsymbol{Q}}_{a_1}, ..., \bar{\boldsymbol{Q}}_{a_{k-1}}\right), \bar{\boldsymbol{Q}}_w\right), \tag{3.4}$$

where $d(\mathcal{P}, \boldsymbol{x})$ is defined as the Euclidean distance from vector $\boldsymbol{x}$ to subspace $\mathcal{P}$, or the norm of the projection of $\boldsymbol{x}$ onto the orthogonal complement of $\mathcal{P}$.

To extend the greedy approach to multilingual settings, we need anchor words that can guide topic inference in *multiple* languages. This motivates our approach for linking words with a dictionary. By choosing linked anchor words, the algorithm can align topics cross-lingually so that the aligned topics form one multilingual topic. However, randomly choosing translation pairs as anchor links will not produce coherent multilingual topics. We need multilingual anchors that also inherit the geometric properties of monolingual anchors. So, the span of anchor words should be maximized in both languages for optimal topic inference. To clearly state our objective,

we define $\mathcal{P}_j^l$ as the subspace spanned by j chosen anchor words in the conditional distribution space of language l,

$$\mathcal{P}_j^l = \text{span}\left(\bar{\boldsymbol{Q}}_{a_1^l}^l, ..., \bar{\boldsymbol{Q}}_{a_j^l}^l\right). \tag{3.5}$$

Word w is a good choice of a k^{th} anchor if $\bar{\boldsymbol{Q}}_w$ is far enough from $\mathcal{P}_{k-1}^l$ so that having $\bar{\boldsymbol{Q}}_w$ as an additional vertex can greatly expand span of anchors. A word might be a great choice for an anchor in one language, but we cannot select it if its translation is a poor choice for the other language (Figure 3.2). We need to pick linked words $w \in \mathcal{V}_S$ and $v \in \mathcal{V}_T$ such that w is far from $\mathcal{P}_{k-1}^S$ and v is also far away from $\mathcal{P}_{k-1}^T$. Then, adding w and v as anchor words can increase total span of anchor word set in both languages. Using this intuition, we maximize the lower bound on the distance from anchor words to $\mathcal{P}_{k-1}^S$ and $\mathcal{P}_{k-1}^T$. We select anchor words w and v such that

$$a_k^S, a_k^T = \arg\max_{w,v} \min\left\{d\left(\mathcal{P}_{k-1}^S, \bar{\boldsymbol{Q}}_w^S\right), d\left(\mathcal{P}_{k-1}^T, \bar{\boldsymbol{Q}}_v^T\right)\right\} \quad \text{subject to } (w,v) \in \mathcal{B}. \tag{3.6}$$

We greedily select anchors $a_k^S \in \mathcal{V}_S, a_k^T \in \mathcal{V}_T$ such that Equation 3.6 is satisfied on every iteration k. Words with multiple translations are elegantly addressed: if an anchor word w is picked already, then it is not likely to be picked again. The algorithm expands both convex hulls simultaneously with each iteration. Indeed, more translations aid our anchor search because there will be more linked anchors to choose from. Even if the algorithm chooses anchor words similar in meaning within the same language, interactivity can help remove duplicate topics (Section 3.2.2). After picking a set of anchor words for each language, multilingual anchoring follows FastAnchorWords (Section 3.1.1). Topic matrices $\boldsymbol{A}_S$ and $\boldsymbol{A}_T$ are separately recovered (Equa-

tions 3.1, 3.2). These matrices are the output of multilingual anchoring. In the next sections, we show how MTAnchor further updates $\boldsymbol{A}_S$ and $\boldsymbol{A}_T$ based on human feedback.

Lacking dictionary entries If dictionary entries are scarce, then we cannot constrain the anchor words to only be words from the dictionary. So, we independently find anchor words for each language using RecoverL2. This reduction to monolingual settings resembles other cross-lingual models: JointLDA reduces to LDA and PTLDA reduces to TLDA when there are no dictionary entries (Jagarlamudi and Daumé, 2010; Hu et al., 2014b).

Predicting labels from topics Multilingual anchoring is an unsupervised method, but the topic distribution acts as a low-dimensional representation for each document (Bengio et al., 2013; Xiao and Guo, 2013; Rastogi et al., 2015). To infer the topic distribution of documents, we pass in the topic matrices as inputs into variational inference (Blei et al., 2003), where topic variational parameter β is fixed and only document variational parameter α is fitted (Sections 2.2.1). Then, we train a linear SVM on the topic distributions of documents (Fan et al., 2008) to classify document labels.

3.2.2 Interactive Topic Alignment

Multilingual anchoring uses translations to find anchor words that can lead to better topics for both languages. However, we cannot completely rely on dictionary entries to construct the topic model. In reality, translations may not be available, could be a poor fit for the dataset, or might be wrong. In addition to problems with the dictionary, the data may be too noisy, or the anchoring algorithm returns a topic model unsuited for our needs (e.g., if a user needs to separate

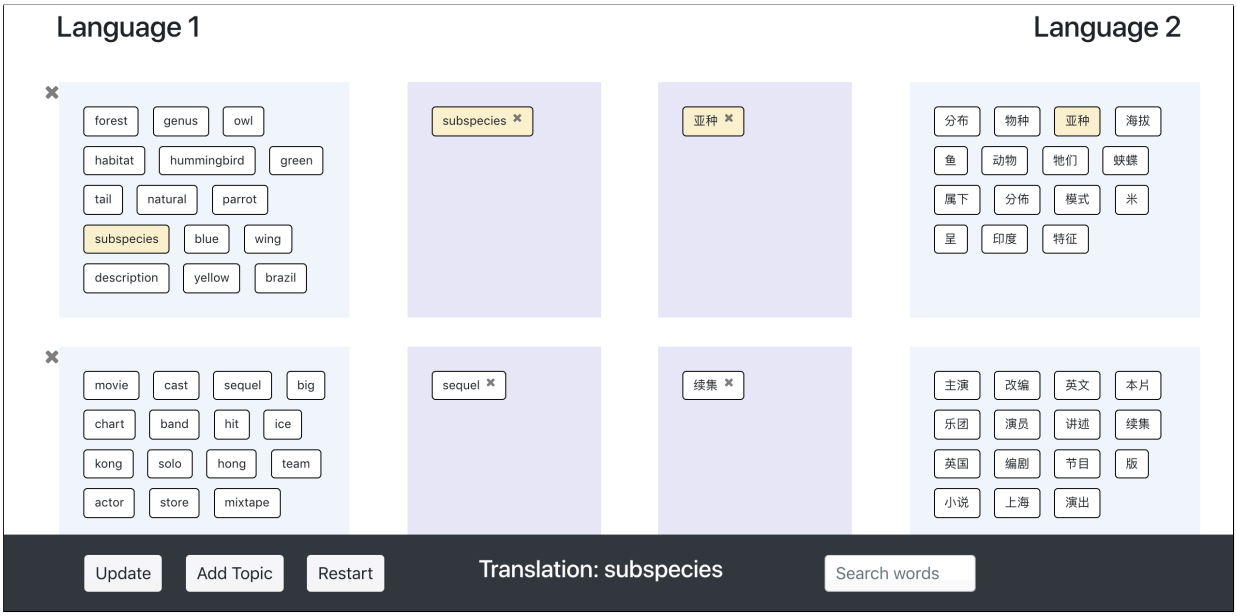


Figure 3.3: The user interface for exploring topics in English and Chinese documents. Anchor words are in the center, while the most likely words for each topic are on the left and right sides of the interface. The user can drag words from the side and add them as anchor words. When the user hovers over "亞種(yàzhǒng)", then its translation, "subspecies", appears at the bottom of the screen. When the user presses on the word, all occurrences of it and its translation are highlighted in yellow. Users can type words in the "Search words" box to find which words are in the vocabulary. These features help the user explore topics in an unfamiliar language.

news from opinion and the topic model puts them together). Thus, we incorporate interactivity into MTAnchor so that we can extract linguistic and cultural knowledge from humans.

First, MTAnchor takes in a pair of comparable corpora and a bilingual dictionary as inputs. Next, it uses multilingual anchoring (Section 3.2.1) to find sets of anchor words for each language. After the algorithm recovers topic matrices, the interface shows information about the topic model. The user can press on the red "X" to delete any incoherent or duplicate topics (Figure 3.3). The user can also add new topics by pressing on "Add Topics". The interface will create a new blank row beneath the existing topics. Then, the user can add words as anchors to the new topic. These features are similar to the ones used for interactively modeling monolingual topics (Section 2.3.2).

Once the user finishes choosing anchor words for each topic, they press "Update Topics". This is a signal for MTAnchor to retrieve new anchor words from the interface and run multiword anchoring (Section 3.1.2). The algorithm approximates $\bar{\boldsymbol{Q}}_w$ for every word w in the vocabulary and then recomputes the topic matrices for each language. When MTAnchor finds new topics, the user can see the updated topics on the interface. At this point, anchors no longer have to be linked by dictionary entries because MTAnchor does not select anchors based on Equation 3.6. After the initial alignment, users define anchors and customize the topic model to their own needs.

3.3 Multilingual Topic Modeling Experiments

The first dataset consists of Wikipedia articles: 11,043 in English and 10,135 in Chinese. We shorten the articles to contain no more than three sections. We lemmatize the English articles using WordNet Lemmatizer (Bird et al., 2009) and segment the Chinese articles using Stanford CoreNLP (Manning et al., 2014). For both languages, the articles fall under one of six categories: film, music, animals, politics, religion, and food.

Another dataset consists of Amazon reviews: 53,558 in English and 53,160 in Chinese (mostly from Taiwan) (Constant et al., 2009). Each review has a rating, ranging from one to five. Since about half of the reviews have a rating of five, we change the classification task to a binary problem by labeling reviews with rating of five as "1" and the rest as "0". For the Wikipedia and Amazon datasets, the training-test split is set to 80:20. For the Chinese-English dictionary, we use entries from MDBG.[3]

To test low-resource languages, we use data from the LORELEI Sinhalese language pack (Strassel

[3] https://www.mdbg.net/chinese/dictionary?page=cc-cedict.

and Tracey, 2016). These language packs are created to develop technologies that can process data in low-resource languages. In the pack, only a small subset of documents is labeled based on need type.[4] So, we treat the classification task as a semi-supervised problem. There are eight possible labels: evacuation, food supply, search/rescue, utilities, infrastructure, medical assistance, shelter, and water supply (Strassel et al., 2017). Out of the 1,100 (4,790) English (Sinhalese) documents, only 77 (49) of them have labels. For each language, half of the labeled documents are in the training set and the other half are in the test set. For the Sinhalese-English dictionary, we use entries from the LORELEI Sinhalese language pack.

We run experiments to evaluate three methods: multilingual anchoring, MTAnchor, and MCTA (Multilingual Cultural-common Topic Analysis) (Shi et al., 2016). We choose MCTA as a baseline because it is a recent work on multilingual topic models with readily available code and aligns topics using a bilingual dictionary. We train models on multilingual anchoring and MCTA with twenty topics. For MTAnchor, we initially show users twenty topics, but the final number of topics is their choice. All methods are implemented in Python on a 2.3 GHz Intel Core i5 processor.

The data for the MTAnchor user study are the English-Chinese Wikipedia articles. We invite twenty participants on Amazon Mechanical Turk (MTurk) to partake in the study. Each user is given thirty minutes to interact with the interface.[5] MTAnchor scales with the number

[4]Documents in LORELEI language pack have multiple need types, but we have simplified the classification task by assigning only the first label to each document.

[5]Synopsis of user instructions: "There are 11,000 English Wikipedia articles and 10,000 Chinese Wikipedia articles, which belong to one of six categories: film, music, animals, politics, religion, food. Your goal is to find topics that can help classify documents within 30 minutes."

of unique word types, rather than number of documents or number of words in the documents, so updates to the system take no longer than seven seconds on average. We only approve HITs from workers who have completed the task for the first time. After worker finishes the task, the interface provides a unique code for them to enter on MTurk. These rules ensure fair assessment of workers' interaction with MTAnchor.

3.3.1 Evaluating multilingual topics

Ideally, topic models should have topics that are *interpretable* and *useful* as classification features. So, we primarily base evaluation on two measures: classification accuracy and topic coherence. Measuring topic coherence considers both intrinsic and extrinsic scores (Lau et al., 2014). The difference between the two is the reference corpus.[6] The intrinsic score uses the trained corpus itself, whereas the extrinsic score uses an external, larger dataset. The Sinhalese extrinsic coherence scores are not available because a large reference corpus cannot be formed for low-resource languages. By measuring both, we can evaluate the model's interpretability within a local and global context.

We evaluate these metrics separately for each language: English (EN), Chinese (ZH), and Sinhalese (SI). To classify labels from topics, we use the same procedure as described in Section 3.2.1. Then, we measure intra-lingual (I) and cross-lingual accuracy (C) with F_1 scores. Intra-lingual accuracy refers to percentage of documents classified correctly using a classifier trained on documents in the *same* language. Cross-lingual accuracy refers to percentage of documents classified correctly using a classifier trained on documents in a *different* language (testing

[6]Measuring topic coherence requires a reference corpus to sample lexical probabilities.

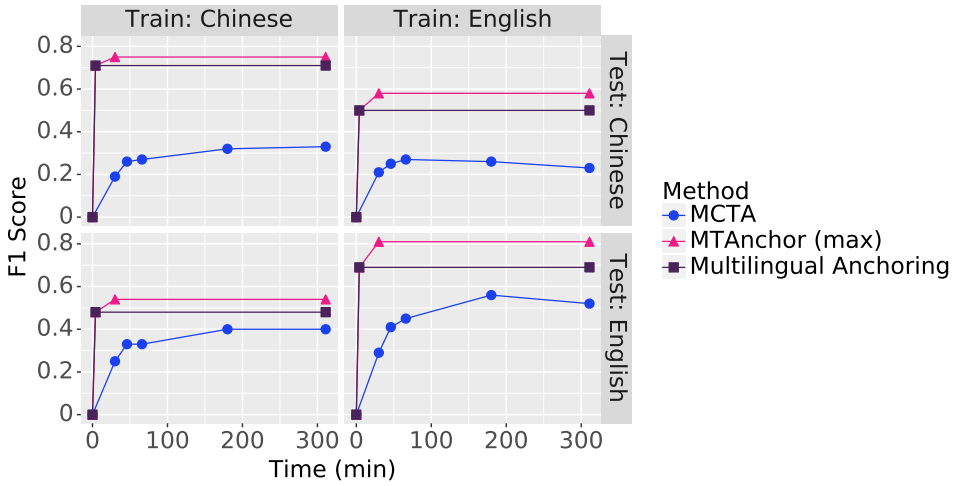


Figure 3.4: Classification accuracy over time until MCTA converges. For the Wikipedia dataset, multilingual anchoring converges within 5 minutes, but MCTA takes 5 hours and 18 minutes to converge. Multilingual anchoring outperforms MCTA in speed and classification accuracy.

the algorithm's ability to generalize). For topic coherence, we use the NPMI (normalized pointwise mutual information) variant of automated topic interpretability scores over the fifteen most probable words in a topic (Lau et al., 2014). For intrinsic scores (I), we use the trained corpus itself as the reference corpus. For extrinsic scores (E), we use 2.2M English Wikipedia articles and 1.1M Chinese Wikipedia articles.

During the user study, we hold out 100 documents as a development set for each corpus. Each time the user updates topics, the interface shows classification accuracy on the development set. When the user finally submits final anchor words, we evaluate their topics on the test set.

3.3.2 Results

In experiments, multilingual anchoring converges much faster than MCTA (Figure 3.4). We compare test accuracy across experiments for multilingual anchoring, MTAnchor, and MCTA, but only report the maximum and median scores from MTAnchor user experiments (Table 3.1). We

Dataset	Method	EN-I	ZH-I SI-I	EN-C	ZH-C SI-C
Wikipedia (EN-ZH)	Multilingual anchoring	69.49%	71.24%	50.37%	47.76%
	MTAnchor (maximum)	**80.71%**	**75.33%**	**57.62%**	**54.54%**
	MTAnchor (median)	69.49%	71.44%	50.27%	47.22%
	MCTA	51.56%	33.35%	23.24%	39.79%
Amazon (EN-ZH)	Multilingual anchoring	**59.79%**	**61.10%**	**51.73%**	**53.20%**
	MCTA	49.53%	50.64%	50.27%	49.49%
LORELEI (EN-SI)	Multilingual anchoring	**20.78%**	**32.65%**	**24.49%**	**24.68%**
	MCTA	12.99%	26.53%	4.08%	15.58%

Table 3.1: Comparison of test accuracy among multilingual topic modeling methods. Multilingual anchoring scores higher in classification accuracy than MCTA. MTAnchor does as well as multilingual anchoring on average with few users showing significant improvement in accuracy.

Dataset	Method	EN-I	ZH-I SI-I	EN-E	ZH-E SI-E
Wikipedia (EN-ZH)	Multilingual anchoring	0.141	0.178	0.084	0.128
	MTAnchor (maximum)	**0.195**	**0.198**	**0.103**	**0.147**
	MTAnchor (median)	0.141	0.178	0.084	0.129
	MCTA	0.126	0.085	0.000	0.037
Amazon (EN-ZH)	Multilingual anchoring	**0.069**	**0.061**	**0.031**	**0.045**
	MCTA	-0.028	0.019	0.017	0.011
LORELEI (EN-SI)	Multilingual anchoring	0.077	0.000	0.025	n/a
	MCTA	**0.132**	0.000	**0.036**	n/a

Table 3.2: Comparison of topic coherence among multilingual topic modeling methods. Multilingual anchoring scores higher in topic coherence than MCTA. MTAnchor does as well as multilingual anchoring on average, but a few users can produce more interpretable topics.

also compare topic coherence across all experiments (Table 3.2). For English-Chinese datasets, multilingual anchoring performs better than MCTA in all metrics. For English-Sinhalese LORELEI dataset, topics from multilingual anchoring are more useful for classification tasks but are less coherent than MCTA topics.

In every metric, the MTAnchor maximum score across all users is higher than scores from other methods. The MTAnchor median score across all users is approximately same as those of multilingual anchoring for all metrics. A few users outperform multilingual anchoring by

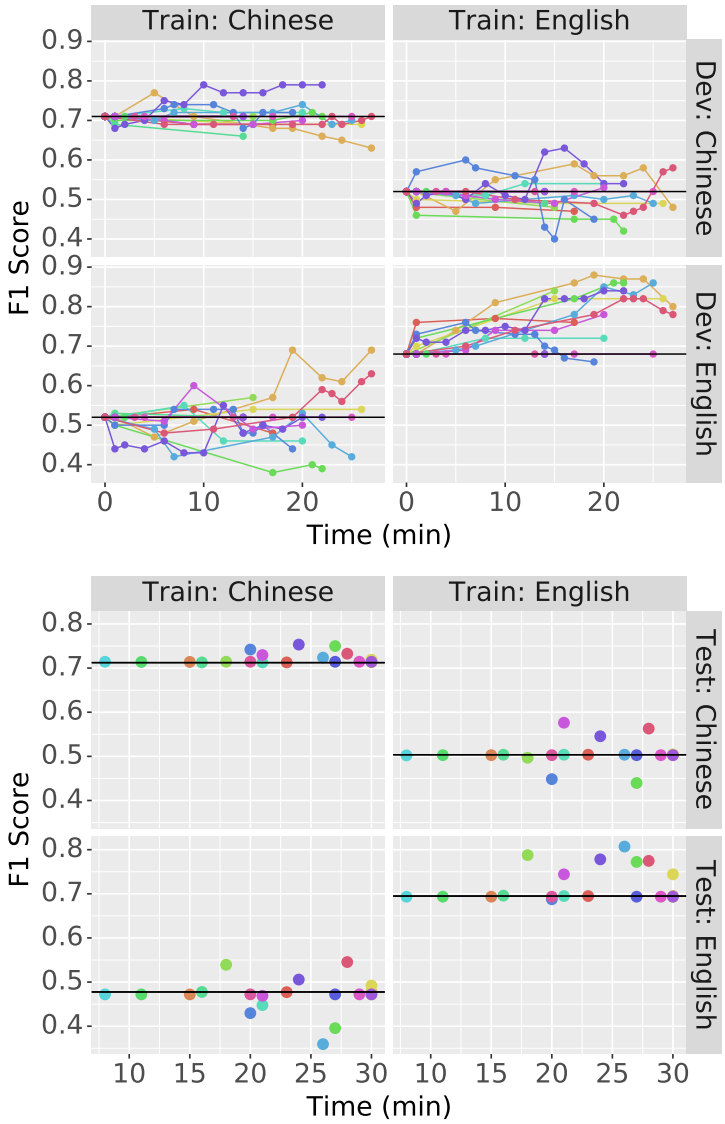


Figure 3.5: Classification accuracy of each participant in the MTAnchor user study over time. Each plot indicates the language of topics that the classifier is trained on and the language of topics that the classifier is tested on. The black horizontal line denotes multilingual anchoring score (no interactive updates). Each colored line represents a different user interaction and shows the fluctuation in scores on development set (top). Each colored point represents the final classification score on the test set; the point's x-coordinate indicates total duration of user's session (bottom).

spending more time interacting with the model (Figure 3.5). Within thirty minutes, a user can improve topic coherence and reach up to a 0.40 increase in any one of the classification scores.

Dataset	Method	Topic
Wikipedia	MCTA	dog san movie mexican fighter novel california 主演 改編 本 小説 拍攝 角色 戰士
	Multilingual anchoring	**adventure** daughter bob kong hong robert movie 主演 改編 本片 飾演 **冒險** 講述 編劇
	MTAnchor	**kong hong movie** office martial box reception 主演 改編 飾演 本片 **演員 編劇** 講述
Amazon	MCTA	woman food eat person baby god chapter 來貨 頂頂 水 耳機 貨物 張傑 傑 同樣
	Multilingual anchoring	eat diet food recipe **healthy** lose weight **健康** 幫 吃 身體 全面 同事 中醫
LORELEI	MCTA	help need floodrelief please families needed victim
	Multilingual anchoring	aranayake warning landslide site missing nbro areas

Table 3.3: Top seven words of sample English and Chinese topics are shown with anchors bolded. Topics from multilingual anchoring and MTAnchor are more relevant to document labels, thereby making them more useful as features for classification.

3.4 Analyzing Multilingual Topics

Multilingual anchoring is a spectral approach to modeling multilingual topics. The algorithm converges much faster than generative methods (Figure 3.4) and resulting topics form better vector representations for documents (Table 3.1). An advantage of anchoring over generative models is its robustness and practicality (Arora et al., 2013). Generative methods need long documents to correctly estimate topic-word distributions, but anchoring handles documents of any size (Arora et al., 2012). This is evident in models built on the Amazon dataset, which contains reviews with only one to three sentences. The health topic for multilingual anchoring is more interpretable than that of MCTA (Table 3.3).

Arora et al. (2013) observe that more specific words appear in the top words of anchor-based topics. This is clearly shown in the LORELEI experiments; a topic from MCTA has general words like "help" and "need", while a topic from multilingual anchoring has specific words like "aranayanke", a town in Sri Lanka, and "nbro", which stands for the National Building Research Organization (Table 3.3). Both topics are about the 2016 Sri Lankan floods, but the topic from MCTA cannot specify the "need" type of documents. So, accuracy is higher when using topics from multilingual anchoring to classify documents. However, LORELEI experiments show that multilingual anchoring topics are less interpretable than MCTA topics. This might be caused by the obscure top topic words. Arayanake is a Sri Lankan town and "nbro" stands for National Building Research Organization. These words may have lowered coherence because they do not co-occur frequently with other top topic words. In this case, using MTAnchor can possibly increase topic coherence.

In the user study, a few participants create topics that are more applicable for specific tasks. In one experiment, a user finds the topic with anchor words "adventure" and "冒險(màoxiǎn)" too vague. The user knows that the task is to classify Wikipedia articles into one of six categories, so they add movie-related terms as anchors, like "movie", "演員(yǎnyuán)", and "編劇(biānjù)". Afterward, their topics significantly improves in classification accuracy and coherence. Other participants do not significantly change the topic model through interactive updates. More work can look into improving MTAnchor so that updates change topic distributions more drastically.

Interestingly, the classification and topic coherence scores for English topics increase considerably after user interaction compared to Chinese topics. The participants are anonymous MTurk workers, so we are not aware of their language skills. We believe that workers are most likely fluent in English because the MTurk website is only available in English. If this fact holds

true, then it can explain why the English topics have much higher scores than the Chinese ones. It also shows that people can improve topic models with prior knowledge, which supports the need for human-in-the-loop algorithms. In the future, it would be interesting to observe how language fluency affects quality of multilingual topics.

3.5 Conclusion

We propose an interactive, multilingual topic model called MTAnchor. Users align and customize topics across languages. By aligning topics, the users create cross-lingual representations of the documents. These representations can be used for tasks like text classification. While the interaction mainly helps with transductive transfer learning, it also involves inductive transfer learning because the topic distributions serve as features for other applications. MTAnchor exemplifies the importance of user feedback for both types of transfer learning.

In this chapter, users interact with topic models to create representations for text classification. In Chapter 4, we will look at adapting language models for text classification. Specifically, we focus our efforts on BERT, a commonly used transformer model. The model is more accurate but also requires more labeled data. So, we try to reduce the amount of labeled data through active learning.

Chapter 4: From Language Modeling to Text Classification[1]

The transformer models can generalize across several NLP tasks due to innovations like the pre-training task, contextualization, and self-attention mechanism (Section 2.2.1). Yet the price of adopting transformer-based models is to collect and label more training data. If these models are not fine-tuned on enough examples, their accuracy drastically varies across different hyperparameter configurations (Dodge et al., 2020). Moreover, computational resources are a major drawback as training one model can cost thousands of dollars in cloud computing and hundreds of pounds in carbon emissions (Strubell et al., 2019).

To overcome these issues with modern NLP models, active learning is a fitting solution. However, traditional active learning depends on warm-starting the model with information about the task (Ash and Adams, 2019). Also, the model may experience larger gains in accuracy if the right data is labeled from the start (Felt et al., 2015). In this chapter, we focus on the *cold-start* setting where the model has fine-tuned on few to no examples. Given the knowledge already encoded in pre-trained models, the annotation for a new task should focus on the information missing from pre-training. This chapter develops ALPS (Active Learning by Processing Surprisal), an active learning strategy for BERT-based models.[2] While many methods randomly choose an

[1]Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020a. Cold-start active learning through self-supervised language modeling. In *Proceedings of Empirical Methods in Natural Language Processing*

[2]`https://github.com/forest-snow/alps`

initial sample, ALPS selects the first batch of data using the masked language modeling loss. We evaluate our approach on four text classification datasets spanning across three different domains. ALPS outperforms active learning baselines in accuracy and algorithmic efficiency. The success of ALPS highlights the importance of cold-start active learning for inductive transfer learning.

4.1 Preliminaries

We formally introduce the setup, notation, and terminology that will be used throughout this chapter.

Pre-trained Encoder Pre-training uses the language modeling loss to train encoder parameters for generalized representations We call the model input $\boldsymbol{x} = (w_i)_{i=1}^{l}$ a "sentence", which is a sequence of l tokens w from a vocabulary $\mathcal{V}$. Given weights $\boldsymbol{W}$, the encoder h maps $\boldsymbol{x}$ to a d-dimensonal hidden representation $h(\boldsymbol{x}; \boldsymbol{W})$. We use BERT (Devlin et al., 2019) as our data encoder, so h is pre-trained with two tasks: masked language modeling (MLM) and next sentence prediction. The embedding $h(\boldsymbol{x}; \boldsymbol{W})$ is computed as the final hidden state of the `[CLS]` token in $\boldsymbol{x}$. We also refer to $h(\boldsymbol{x}; \boldsymbol{W})$ as the BERT embedding.

Fine-tuned Model We fine-tune BERT on the downstream task by training the pre-trained model and the attached sequence classification head (Section 2.2.1). Suppose that f represents the model with the classification head, has parameters $\theta = (\boldsymbol{W}, \boldsymbol{V})$, and maps input $\boldsymbol{x}$ to a C-dimensional vector with confidence scores for each label. Specifically, $f(\boldsymbol{x}; \theta) = \sigma(\boldsymbol{V} \cdot h(\boldsymbol{x}; \boldsymbol{W}))$ where σ is a softmax function.

Let D be the labeled data for our classification task where the labels belong to set $\mathcal{Y} =$

Algorithm 1 Active learning for Sentence Classification

Require: Initial model $f(\boldsymbol{x}; \theta_0)$ with pre-trained encoder $h(\boldsymbol{x}; \boldsymbol{W}_0)$, unlabeled data pool $\mathcal{U}$, number of queries per iteration k, number of iterations T, sampling algorithm $\mathcal{A}$

$\mathcal{D} = \{\}$
for iterations $t = 1, \ldots, T$ **do**
  if $\mathcal{A}$ is cold-start for iteration t **then**
    $M_t(\boldsymbol{x}) = f(\boldsymbol{x}; \theta_0)$
  else
    $M_t(\boldsymbol{x}) = f(\boldsymbol{x}; \theta_{t-1})$
  end if
  $\mathcal{Q}_t \leftarrow$ Apply $\mathcal{A}$ on model $M_t(\boldsymbol{x})$ and data $\mathcal{U}$ to select k sentences to query
  $\mathcal{D}_t \leftarrow$ Label queries $\mathcal{Q}_t$
  $\mathcal{D} = \mathcal{D} \cup \mathcal{D}_t$
  $\mathcal{U} = \mathcal{U} \setminus \mathcal{D}_t$
  $\theta_t \leftarrow$ Fine-tune $f(\boldsymbol{x}; \theta_0)$ on $\mathcal{D}$
end for
return $f(\boldsymbol{x}; \theta_T)$

$\{1, ..., C\}$. During fine-tuning, we take a base classifier f with weights $\boldsymbol{W}_0$ from a pre-trained encoder h and fine-tune f on D for new parameters θ_t. Then, the predicted classification label is $\hat{y} = \arg\max_{y \in \mathcal{Y}} f(\boldsymbol{x}; \theta_t)_y$.

Active Learning for Sentence Classification Assume that there is a large unlabeled dataset $U = \{(\boldsymbol{x}_i)\}_{i=1}^n$ of n sentences. The goal of active learning is to sample a subset $D \subset U$ efficiently so that fine-tuning the classifier f on subset D improves test accuracy. On each iteration t, the learner uses strategy $\mathcal{A}$ to acquire k sentences from dataset U and queries for their labels (Algorithm 3). Strategy $\mathcal{A}$ usually depends on an acquisition model M_t (Lowell et al., 2019). If the strategy depends on model warm-starting, then the acquisition model M_t is f with parameters θ_{t-1} from the previous iteration. Otherwise, we assume that M_t is the pre-trained model with parameters θ_0. After T rounds, we acquire labels for Tk sentences. We provide more concrete details about active learning simulation in Section 4.4.

4.2 The Uncertainty–Diversity Dichotomy

In Section 2.3.1, we talk about two general frameworks for active learning: uncertainty sampling and diversity sampling. Dasgupta (2011) describes uncertainty and diversity as the "two faces of active learning". While uncertainty sampling efficiently searches the hypothesis space by finding difficult examples to label, diversity sampling exploits heterogeneity in the feature space. Uncertainty sampling requires model warm-starting because it depends on model predictions, whereas diversity sampling can be a cold-start approach. A successful active learning strategy should integrate both aspects, but its exact implementation is an open research question. For example, a naïve idea is to use a fixed combination of strategies to sample points. Nevertheless, Hsu and Lin (2015) experimentally show that this approach hampers accuracy. BADGE (Ash et al., 2020) optimizes for both uncertainty and diversity by using confidence scores and clustering. This strategy beats previous work in uncertainty sampling and diversity sampling.

4.2.1 BADGE

The goal of BADGE is to sample a diverse and uncertain batch of points for training neural networks. The algorithm transforms data into representations that encode model confidence and then clusters these transformed points. First, an unlabeled point $\boldsymbol{x}$ passes through the trained model to obtain its predicted label $\hat{y}$. Next, a *gradient embedding* $\boldsymbol{g}_x$ is computed for $\boldsymbol{x}$ such that it embodies the gradient of the cross-entropy loss on $(f(\boldsymbol{x};\theta), \hat{y})$ with respect to the parameters of the model's last layer. The gradient embedding is

$$(\boldsymbol{g}_x)_i = (f(\boldsymbol{x};\theta)_i - \mathbb{1}(\hat{y} = i))h(\boldsymbol{x};\boldsymbol{W}). \tag{4.1}$$

The i-th block of $\boldsymbol{g}_x$ is the hidden representation $h(\boldsymbol{x}; \boldsymbol{W})$ scaled by the difference between model confidence score $f(\boldsymbol{x}; \theta)_i$ and an indicator function $\mathbb{1}$ that indicates whether the predictive label $\hat{y}$ is label i. Finally, BADGE chooses a batch to sample by applying k-MEANS++ (Arthur and Vassilvitskii, 2006) on the gradient embeddings. These embeddings consist of model confidence scores and hidden representations, so they encode information about both uncertainty and the data distribution. By applying k-MEANS++ on the gradient embeddings, the chosen examples differ in feature representation and predictive uncertainty.

4.2.2 Limitations

BADGE combines uncertainty and diversity sampling to profit from advantages of both methods but also brings the downsides of both: reliance on warm-starting and computational inefficiency.

Model Uncertainty and Inference Dodge et al. (2020) observe that training is highly unstable when fine-tuning pre-trained language models on small datasets. Accuracy significantly varies across different random initializations. The model has not fine-tuned on enough examples, so model confidence is an unreliable measure for uncertainty. While BADGE improves over uncertainty-based methods, it still relies on confidence scores $f(\boldsymbol{x}; \theta)_i$ when computing the gradient embeddings (Equation 4.1). Also, it uses labels inferred by the model to compensate for lack of supervision in active learning, but this inference is inaccurate for ill-trained models. Thus, warm-start methods may suffer from problems with model uncertainty or inference.

Algorithmic Efficiency Many diversity-based methods involve distance comparison between embedding representations, but this computation can be expensive, especially in high-dimensional space. For instance, CORESET is a farthest-first traversal in the embedding space where it chooses the farthest point from the set of points already chosen on each iteration (Sener and Savarese, 2018). The embeddings may appropriately represent the data, but issues, like the "curse of dimensionality" (Beyer et al., 1999) and the "hubness problem" (Tomasev et al., 2013), persist. As the dimensionality increases, the distance between any two points converges to the same value. Moreover, the gradient embeddings in BADGE have dimensionality of Cd for a C-way classification task with data dimensionality of d (Equation 4.1). These issues make distance comparison between gradient embeddings less meaningful and raises costs to compute those distances.

4.3 A Self-supervised Active Learner

Cold-start active learning is challenging because of the shortage in labeled data. Prior works, like BADGE, often depend on model uncertainty or inference, but these measures can be unreliable if the model has not trained on enough data (Section 4.2.2). To overcome the lack of supervision, what if we apply self-supervision to active learning? For NLP, the language modeling task is self-supervised because the label for each token is the token itself. If the task has immensely improved transfer learning, then it may reduce generalization error in active learning too.

For our approach, we adopt the uncertainty-diversity BADGE framework for clustering embeddings that encode information about uncertainty. However, rather than relying on the classification loss gradient, we use the MLM loss to bootstrap uncertainty estimates. Thus, we combine

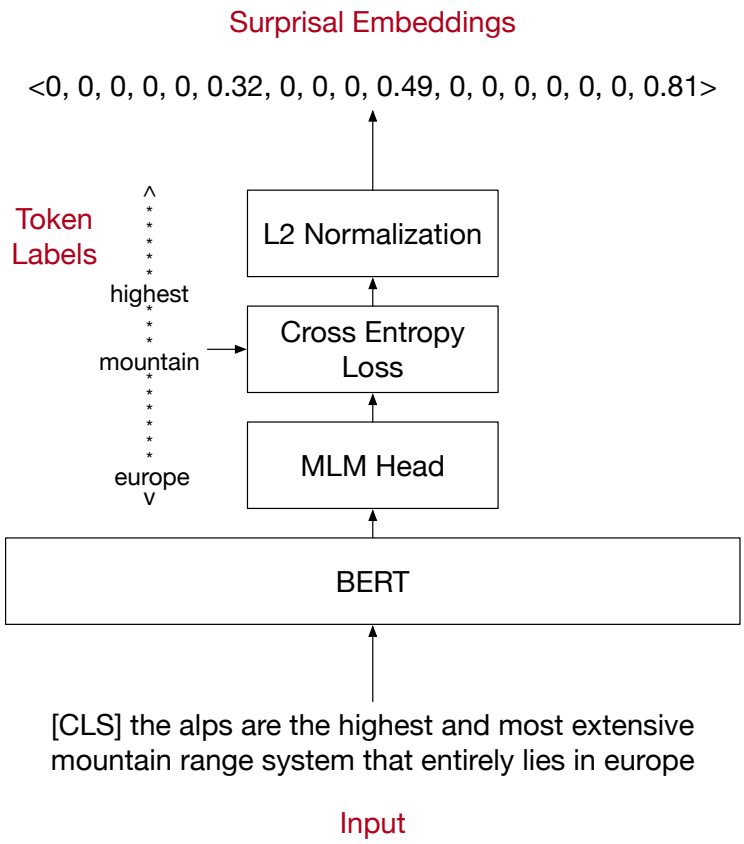


Figure 4.1: To form surprisal embedding $\boldsymbol{s}_x$, we pass in unmasked $\boldsymbol{x}$ through the BERT MLM head and compute cross-entropy loss for a random 15% subsample of tokens against the target labels. The unsampled tokens have entries of zero in $\boldsymbol{s}_x$.

uncertainty and diversity sampling for cold-start active learning.

4.3.1 ALPS

Surprisal Embeddings Inspired by how BADGE forms gradient embeddings from the classification loss, we create *surprisal embeddings* from language modeling. Surprisal theory measures the cost of processing a word by its difficulty to predict from its content (Levy, 2008). Therefore, we use the language modeling loss to compute surprisal of a token. For sentence $\boldsymbol{x}$, we compute surprisal embedding $\boldsymbol{s}_x$ by evaluating $\boldsymbol{x}$ with the MLM objective. To evaluate MLM loss, BERT randomly masks 15% of the tokens in $\boldsymbol{x}$ and computes cross-entropy loss for the masked

tokens against their true token labels. When computing surprisal embeddings, we make one crucial change: *none of the tokens are masked when the input is passed into* BERT. However, we still randomly choose 15% of the tokens in the input to evaluate with cross-entropy against their target token labels. The unchosen tokens are assigned a loss of zero as they are not evaluated (Figure 4.1).

These decisions for not masking input and evaluating only 15% of tokens are made because of experiments on the validation set (Section 4.6). Proposition 1 provides insight on the information encoded in surprisal embeddings. Finally, the surprisal embedding is l_2-normalized as normalization improves clustering (Aytekin et al., 2018). If the input sentences have a fixed length of l, then the surprisal embeddings have dimensionality of l. The length l is usually less than the hidden size of BERT embeddings.

Proposition 1. *For an unnormalized surprisal embedding $\boldsymbol{s}_x$, each nonzero entry $(\boldsymbol{s}_x)_i$ estimates $I(w_i)$, the surprisal of its corresponding token within the context of sentence $\boldsymbol{x}$.*

Proof. Extending notation from Section 4.1, assume that m is the MLM head, with parameters $\phi = (\boldsymbol{W}, \boldsymbol{Z})$, which maps input x to a $l \times |\mathcal{V}|$ matrix $m(\boldsymbol{x}; \phi)$. The ith row $m(\boldsymbol{x}; \phi)_i$ contains prediction scores for w_i, the ith token in $\boldsymbol{x}$. Suppose that w_i is the jth token in vocabulary $\mathcal{V}$. Then, $m(\boldsymbol{x}; \phi)_{i,j}$ is the likelihood of predicting w_i correctly.

Now, assume that context is the entire input $\boldsymbol{x}$ and define the language model probability p_m as,

$$p_m(w_i \mid \boldsymbol{x}) = m(\boldsymbol{x}; \phi)_{i,j}. \tag{4.2}$$

Salazar et al. (2020) have a similar definition as Equation 4.2 but instead have defined it in terms of the masked input. We argue that their definition can be extended to the unmasked input $\boldsymbol{x}$.

Algorithm 2 Single iteration of ALPS

Require: Pre-trained encoder $h(\boldsymbol{x}; \boldsymbol{W}_0)$, unlabeled data pool $\mathcal{U}$, number of queries k

for sentences $\boldsymbol{x} \in \mathcal{U}$ **do**
    Compute $\boldsymbol{s}_x$ with MLM head of $h(\boldsymbol{x}; \boldsymbol{W}_0)$
end for
$\mathcal{M} = \{\boldsymbol{s}_x \mid \boldsymbol{x} \in \mathcal{U}\}$
$\mathcal{C} \leftarrow k$-MEANS cluster centers of $\mathcal{M}$
$\mathcal{Q} = \{\arg\min_{\boldsymbol{x} \in \mathcal{U}} \|\boldsymbol{c} - \boldsymbol{s}_x\| \,|\boldsymbol{c} \in \mathcal{C}\}$
return $\mathcal{Q}$

During BERT pre-training, the MLM objective is evaluated on the `[MASK]` token for 80% of the time, random token for 10% of the time, and the original token for 10% of the time. This helps maintain consistency across pre-training and fine-tuning because `[MASK]` never appears in fine-tuning (Devlin et al., 2019). Thus, we assume that m estimates occurrence of tokens within a maskless context as well.

Next, the information-theoretic surprisal (Shannon, 1948) is defined as $I(w) = -\log p(w \mid \boldsymbol{c})$, the negative log likelihood of word w given context $\boldsymbol{c}$. If w_i is sampled and evaluated, then the ith entry of the unnormalized surprisal embedding is,

$$
\begin{aligned}
(\boldsymbol{s}_x)_i &= -\log m(\boldsymbol{x}; \phi)_{i,j} = -\log p_m(w_i \mid \boldsymbol{x}) \\
&= I(w_i).
\end{aligned}
$$

□

Proposition 1 shows that the surprisal embeddings consist of estimates for token-context surprisal. Intuitively, these values can help with active learning because they highlight the information missing from the pre-trained model. For instance, consider the sentences: "this is my favorite television show" and "they feel ambivalent about catholic psychedelic synth folk

Dataset	Domain	Train	Dev	Test	# Labels
AG NEWS	News articles	110,000	10,000	7,600	4
IMDB	Sentiment reviews	17,500	7,500	25,000	2
PUBMED 20k RCT	Medical abstracts	180,040	30,212	30,135	5
SST-2	Sentiment reviews	60,615	6,736	873	2

Table 4.1: Sentence classification datasets used in experiments.

music". Tokens from the latter have higher surprisal than those from the former. If this is a sentiment classification task, the second sentence is more confusing for the classifier to learn. The surprisal embeddings indicate sentences challenging for the pre-trained model to understand and difficult for the fine-tuned model to label.

The most surprising sentences contain many rare tokens. If we only train our model on the most surprising sentences, then it may not generalize well across different examples. Plus, we may sample several, atypical sentences that are similar to each other, which is often an issue for uncertainty-based methods (Kirsch et al., 2019). Therefore, we incorporate clustering in ALPS to maintain diversity.

k-MEANS Clustering After computing surprisal embeddings for each sentence in the unlabeled pool, we use k-MEANS to cluster the surprisal embeddings. Then, for each cluster center, we select the sentence that has the nearest surprisal embedding to it. The final set of sentences are the queries to be labeled by an oracle (Algorithm 2). Although BADGE uses k-MEANS++ to cluster, experiments on the validation set show that k-MEANS works better for surprisal embeddings (Section 4.6).

4.4 Active Sentence Classification

We evaluate ALPS on sentence classification for three different domains: sentiment reviews, news articles, and medical abstracts (Table 4.1). To simulate active learning, we sample a batch of 100 sentences from the training dataset, query labels for this batch, and then move the batch from the unlabeled pool to the labeled dataset (Algorithm 3). The initial encoder $h(\boldsymbol{x}; \theta_0)$, is an already pre-trained, BERT-based model (Section 4.4.2). In a given iteration, we fine-tune the base classifier $f(\boldsymbol{x}; \theta_0)$ on the labeled dataset and evaluate the fine-tuned model with classification micro-F_1 score on the test set. We do not fine-tune the model $f(\boldsymbol{x}; \theta_{t-1})$ from the previous iteration to avoid issues with warm-starting (Ash and Adams, 2019). We repeat for ten iterations, collecting a total of 1,000 sentences.

4.4.1 Baselines

We compare ALPS against warm-start methods (Entropy, BADGE, FT-BERT-KM) and cold-start methods (Random, BERT-KM). For FT-BERT-KM, we use BERT-KM to sample data in the first iteration. For other warm-start methods, data is randomly sampled in the first iteration.

Entropy Sample k sentences with highest predictive entropy (Lewis and Gale, 1994; Wang and Shang, 2014).

BADGE Sample k sentences based on diversity in loss gradient (Section 4.2.1).

BERT-KM Cluster pre-trained, l_2-normalized BERT embeddings with k-MEANS and sample the nearest neighbors of the k cluster centers. The algorithm is the same as ALPS except that

	AG NEWS	PUBMED
Random	< 1	< 1
Entropy	7	10
ALPS	14	24
BADGE	23	70
BERT-KM	28	58
FT-BERT-KM	33	79

Table 4.2: Average runtime (minutes) per sampling iteration during active learning simulation for large datasets. BADGE, FT-BERT-KM, and BERT-KM take much longer to run.

BERT embeddings are used.

FT-BERT-KM This is the same algorithm as BERT-KM except the BERT embeddings $h(\boldsymbol{x}; \boldsymbol{W}_{t-1})$ from the previously fine-tuned model are used.

4.4.2 Setup

For each sampling algorithm and dataset, we run the active learning simulation five times with different random seeds. We set the maximum sequence length to 128. We fine-tune on a batch size of thirty-two for three epochs. We use AdamW (Loshchilov and Hutter, 2019) with learning rate of 2e-5, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a linear decay of learning rate.

For IMDB (Maas et al., 2011), SST-2 (Socher et al., 2013), and AG NEWS (Zhang et al., 2015), the data encoder is the uncased BERT-Base model with 110M parameters. For PUBMED (Dernoncourt and Lee, 2017), the data encoder is SCIBERT, a BERT model pre-trained on scientific texts (Beltagy et al., 2019). All experiments are run on GeForce GTX 1080 GPU and 2.6 GHz AMD Opteron 4180 CPU processor; runtimes in Table 4.2.

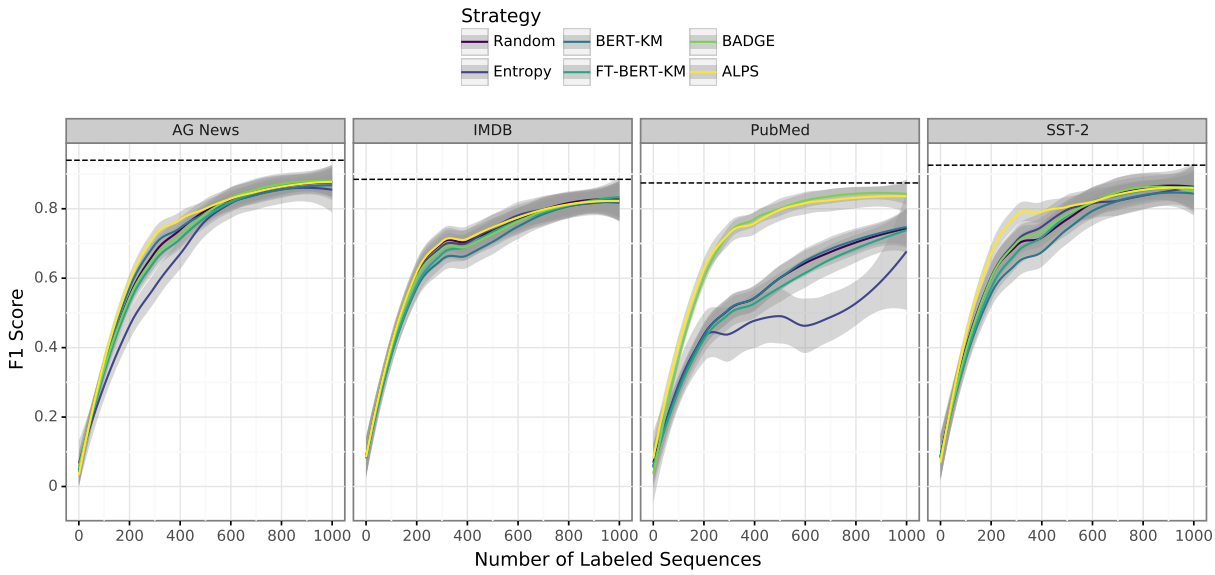


Figure 4.2: Test accuracy of simulated active learning over ten iterations with 100 sentences queried per iteration. The dashed line is the test accuracy when the model is fine-tuned on the entire training dataset. Table 4.1 provides the dataset sizes. Overall, models trained with data sampled from ALPS have the highest test accuracy, especially for the earlier iterations.

4.4.3 Results

The model fine-tuned with data sampled by ALPS has higher test accuracy than the baselines (Figure 4.2). For AG NEWS, IMDB, and SST-2, this is true in earlier iterations. We often see the most gains in the beginning for crowdsourcing (Felt et al., 2015). Interestingly, clustering the fine-tuned BERT embeddings is not always better than clustering the pre-trained BERT embeddings for active learning. The fine-tuned BERT embeddings may require training on more data for more informative representations.

For PUBMED, test accuracy greatly varies between the strategies. The dataset belongs to a specialized domain and is class-imbalanced, so naïve methods show poor accuracy. Entropy sampling has the lowest accuracy because the classification entropy is uninformative in early iterations. The models fine-tuned on data sampled by ALPS and BADGE have about the same ac-

curacy. Both methods strive to optimize for uncertainty and diversity, which alleviates problems with class imbalance.

Our experiments cover the first ten iterations because we focus on the cold-start setting. As sampling iterations increase, test accuracy across the different methods converges. Both ALPS and BADGE already approach the model trained on the full training dataset across all tasks (Figure 4.2). Once the cold-start issue subsides, uncertainty-based methods can be employed to further query the most confusing examples for the model to learn.

4.5 Analysis

Sampling Efficiency Given that the gradient embeddings are computed, BADGE has a time complexity of $\mathcal{O}(Cknd)$ for a C-way classification task, k queries, n points in the unlabeled pool, and d-dimensional BERT embeddings. Given that the surprisal embeddings are computed, ALPS has a time complexity of $\mathcal{O}(tknl)$ where t is the fixed number of iterations for k-MEANS and l is the maximum sequence length. In our experiments, $k = 100$, $d = 768$, $t = 10$, and $l = 128$. In practice, t will not change much, but n and C could be much higher. For large dataset PUBMED, the average runtime per iteration is 24 minutes for ALPS and 70 minutes for BADGE (Table 4.2). So, ALPS can match BADGE's accuracy more quickly.

Diversity and Uncertainty We estimate diversity and uncertainty for data sampled across different strategies. For diversity, we look at the overlap between tokens in the sampled sentences and tokens from the rest of the data pool. A diverse batch of sentences should share many of the same tokens with the data pool. In other words, the sampled sentences can represent the data pool because of the substantial overlap between their tokens. In our simulations, the entire data

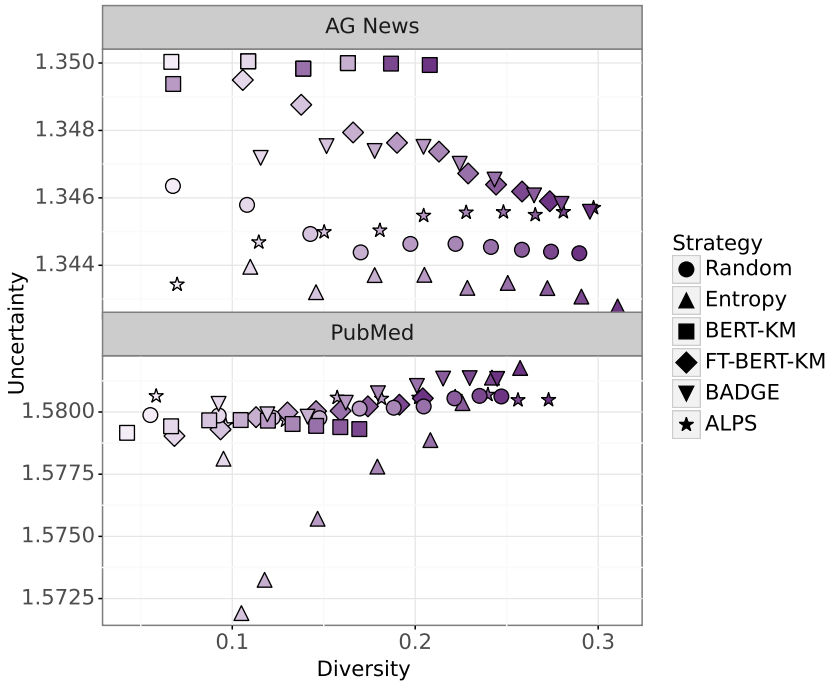


Figure 4.3: Plot of diversity against uncertainty estimates from active learning simulations. Each point represents a sampled batch of sentences. The shape indicates the active learning strategy and the color represents the sample iteration. The lightest color corresponds to the first iteration and the darkest color represents the tenth iteration. While uncertainty estimates are similar across different batches, ALPS shows a consistent increase in diversity without drops in uncertainty.

pool is the training dataset (Section 4.4). So, we compute the Jaccard similarity between $\mathcal{V}_D$, set of tokens from the sampled sentences $\mathcal{D}$, and $\mathcal{V}_{D'}$, set of tokens from the unsampled sentences $\mathcal{U} \setminus \mathcal{D}$,

$$G_d(\mathcal{D}) = J(\mathcal{V}_D, \mathcal{V}_{D'}) = \frac{|\mathcal{V}_D \cap \mathcal{V}_{D'}|}{|\mathcal{V}_D \cup \mathcal{V}_{D'}|}. \tag{4.3}$$

If G_d is high, this indicates high diversity because the sampled and unsampled sentences have many tokens in common. If G_d is low, this indicates poor diversity and representation.

To measure uncertainty, we use $f(x, \theta_*)$, the classifier trained on the full training dataset. In our experiments, classifier $f(x, \theta_*)$ has high accuracy (Figure 4.2) and inference is stable after training on many examples. Thus, we can use the logits from the classifier to understand

its uncertainty toward a particular sentence. First, we compute predictive entropy of sentence x when evaluated by model $f(x, \theta_*)$. Then, we take the average of predictive entropy over all sentences in a sampled batch $\mathcal{D}$. We use the average predictive entropy to estimate uncertainty of the sampled sentences,

$$G_u(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{i=1}^{C} (f(x; \theta_*)_i) \ln(f(x; \theta_*)_i)^{-1}. \tag{4.4}$$

We compute G_d and G_u for batches sampled in the active learning experiments of AG NEWS and PUBMED. Diversity is plotted against uncertainty for batches sampled across different iterations and active learning strategies (Figure 4.3). For AG NEWS, G_d and G_u are relatively low for ALPS in the first iteration. As iterations increase, samples from ALPS increase in diversity and decrease minimally in uncertainty. Samples from other methods have a larger drop in uncertainty as iterations increase. For PUBMED, ALPS again increases in sample diversity without drops in uncertainty. In the last iteration, ALPS has the highest diversity among all the algorithms.

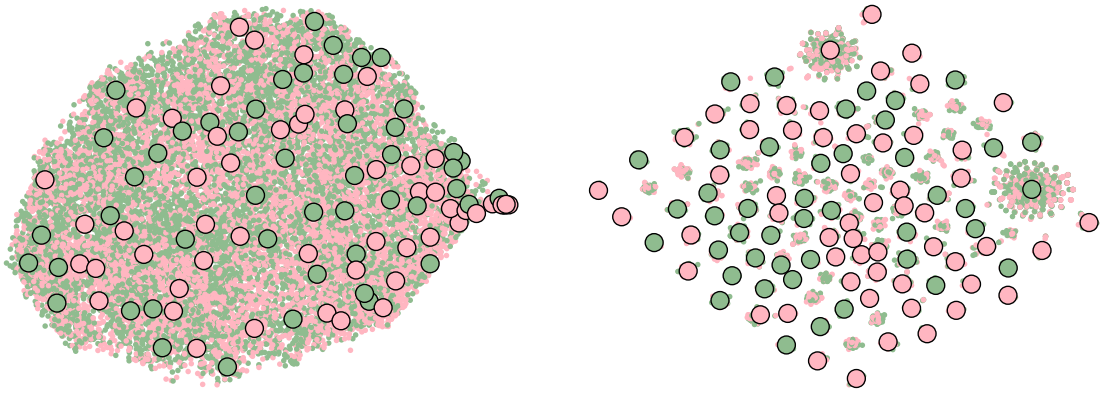
Sample Sentences We take a closer look at the kind of sentences that are sampled by ALPS. Table 4.3 compares sentences that are chosen by ALPS and random sampling in the first active learning iteration. The tokens highlighted are the ones evaluated with surprisal loss. Random sampling can fall prey to data idiosyncrasies. For example, AG News has sixty-two articles about the German golfer Bernhard Langer, and random sampling picks multiple articles about him on one of five runs. For PubMed, many sentences labeled as "methods" are simple sentences with a short, independent clause. While random sampling chooses many sentences of this form, ALPS seems to avoid this problem. Since the surprisal embedding encodes the fluctuation in

	AG NEWS	PUBMED
ALPS	Jason Thomas matches a career-high with 26 points and American wins its fifth straight by beating visiting Ohio, 64-55, Saturday at Bender Arena (Sports)	The results showed that physical activity and exercise capacity in the intervention group was significantly higher than the control group after the intervention . (results)
	Sainsbury says it will take a 550 million pound hit to profits this year as it invests to boost sales and reverse falling market share (Business)	Flumazenil was administered after the completion of endoscopy under sedation to reduce recovery time and increase patient safety . (objective)
Random	Bernhard Langer and Hal Sutton stressed the importance of playing this year's 135th Ryder Cup . . . (Sports)	The study population consisted of 20 interns and medical students (methods)
	BLOOMFIELD TOWNSHIP, Mich. – When yesterday's Ryder Cup pairings were announced, Bernhard Langer knew his team had been given an opportunity. (Sports)	The subject , health care provider , and research staff were blinded to the treatment . (methods)

Table 4.3: Sample sentences from AG News and PubMed while using ALPS and Random in the first iteration. For ALPS, highlighted tokens are the ones that have a nonzero entry in the surprisal embedding. Compared to random sampling, ALPS samples sentences with more diverse content.

information content across the sentence, ALPS is less likely to repeatedly choose sentences with similar patterns in surprisal. This may diversify syntactic structure in a sampled batch.

Surprisal Clusters Prior work uses k-MEANS to cluster feature representations as a cold-start active learning approach (Zhu et al., 2008; Bodó et al., 2011). Rather than clustering BERT embeddings, ALPS clusters surprisal embeddings. We compare the clusters between surprisal embeddings and BERT embeddings to understand the structure of the surprisal clusters. First, we use t-SNE (Maaten and Hinton, 2008) to plot the embeddings for each sentence in the IMDB training set (Figure 4.4). The labels are not well-separated for both embedding sets, but the surprisal embeddings seem easier to cluster. To quantitively measure cluster quality, we use the



(a) BERT embeddings with k-MEANS centers (b) Surprisal embeddings with k-MEANS centers

Figure 4.4: T-SNE plots of BERT embeddings and surprisal embeddings for each sequence in the IMDB training dataset. The enlarged points are the centers determined by k-MEANS (left) and k-MEANS++ (right). The points are colored according to their classification labels. In both sets of embeddings, we cannot clearly separate the points from their labels, but the distinction between clusters in surprisal embeddings seems more obvious.

Silhouette Coefficient for which larger values indicate desirable clustering (Rousseeuw, 1987). The surprisal clusters have a coefficient of 0.38, whereas the BERT clusters have a coefficient of only 0.04.

These results, along with the classification experiments, show that naïvely clustering BERT embeddings is not suited for active learning. Possibly, more complicated clustering algorithms can capture the intrinsic structure of the BERT embeddings. However, this would increase the algorithmic complexity and runtime. Alternatively, one can map the feature representations to a space where simple clustering algorithms work well. During this transformation, important information for active learning must be preserved and extracted. Our approach uses the MLM head, which has already been trained on extensive corpora, to map the BERT embeddings into the surprisal embedding space. As a result, simple k-MEANS can efficiently choose representative sentences.

Dataset	k	Iterative	Single
IMDB	200	0.63 ± 0.04	0.61 ± 0.03
	500	0.74 ± 0.05	0.76 ± 0.04
	1000	0.82 ± 0.01	0.82 ± 0.01
PUBMED	200	0.63 ± 0.03	0.64 ± 0.03
	500	0.80 ± 0.02	0.82 ± 0.01
	1000	0.84 ± 0.00	0.84 ± 0.00

Table 4.4: Test accuracy on IMDB and PubMed between different uses of ALPS for various k, the number of sentences to query. We compare using ALPS iteratively (Iterative) as done in Chapter 4.4 with using ALPS to query all k sentences in one iteration (Single). The test accuracy does not change much, showing that ALPS is flexible to apply in different settings.

Single-iteration Sampling In Section 4.4, we sample data iteratively (Algorithm 3) to fairly compare the different active learning algorithms. However, ALPS does not require updating the classifier because it only depends on the pre-trained encoder. Rather than sampling data in small batches and re-training the model, ALPS can sample a batch of k sentences in one iteration (Algorithm 2). Between using ALPS iteratively and deploying the algorithm for a single iteration, the difference is insignificant (Table 4.4). Plus, sampling 1,000 sentences only takes about 97 minutes for PUBMED and 7 minutes for IMDB.

With this flexibility in sampling, ALPS can accommodate different budget constraints. For example, re-training the classifier may be costly, so users want a sampling algorithm that can query k sentences all at once. In other cases, annotators are not always available, so the number of obtainable annotations is unpredictable. Then, users would prefer an active learning strategy that can query a variable number of sentences for any iteration. These cases illustrate practical needs for a cold-start algorithm like ALPS.

Using BERT-Base for PUBMED The results for PUBMED in Figure 4.2 use a SCIBERT model. Beltagy et al. (2019) show the advantages of using SCIBERT on tasks for scientific domains. So,

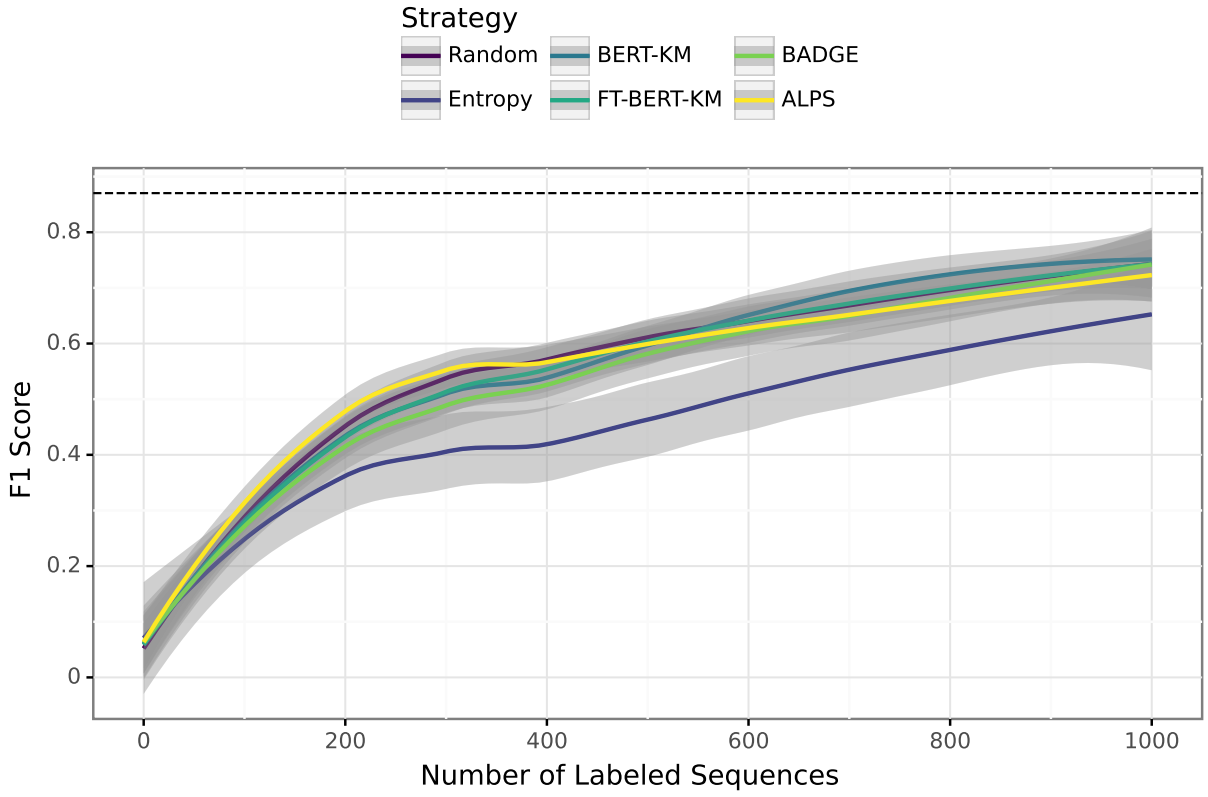


Figure 4.5: The results for active learning simulation of the BERT-Base encoder for PUBMED. Entropy sampling still has lowest accuracy, but there is no optimal strategy among the rest of them. The accuracy of ALPS and BADGE is lower than the experiments in Figure 4.2 that use SCIBERT.

for the main experiments, we are inclined to use the SCIBERT encoder rather than BERT-Base. Figure 4.5 repeats the active learning simulation on PUBMED with a BERT-Base encoder. The accuracy for ALPS and BADGE are no longer as high as they were in the SCIBERT experiments. One possible explanation for this is that BERT-Base does not recognize many words in PUBMED. Similar to entropy sampling, the words are equally surprising to the model. On the other hand, SCIBERT has already pre-trained on scientific texts and has a general understanding of when certain terms should appear. The SCIBERT model can then better detect which sentences are actually confusing in the scientific domain. Future work may look at how various pre-training

	IMDB		SST-2	
	$k = 100$	$k = 200$	$k = 100$	$k = 200$
ALPS	0.60 ± 0.03	$\mathbf{0.69 \pm 0.04}$	$\mathbf{0.57 \pm 0.06}$	$\mathbf{0.64 \pm 0.04}$
ALPS-tokens-0.1	$\mathbf{0.61 \pm 0.05}$	0.63 ± 0.11	0.56 ± 0.07	0.63 ± 0.04
ALPS-tokens-0.2	0.55 ± 0.07	0.65 ± 0.05	$\mathbf{0.57 \pm 0.05}$	0.63 ± 0.05
ALPS-tokens-1.0	0.59 ± 0.05	0.65 ± 0.07	0.56 ± 0.05	0.62 ± 0.05
ALPS-masked	0.59 ± 0.03	0.63 ± 0.09	0.56 ± 0.03	0.60 ± 0.02

Table 4.5: Comparison of validation accuracy between the variants of ALPS to sample data for IMDB and SST-2 in the first two iterations. ALPS-tokens-p varies the percentage p of tokens evaluated with MLM loss when computing surprisal embeddings. ALPS-masked passes in the input with masks as originally done in pre-training. Overall, ALPS has higher mean and smaller variance in accuracy.

strategies (Gururangan et al., 2020) could affect the accuracy of ALPS for active learning.

4.6 Ablation

We provide results from ablation experiments. Here, we look at the effects of token masking, token sampling, and clustering technique on ALPS,

Token masking In our preliminary experiments on the validation set, accuracy improves after passing in the original input with no masks (Table 4.5). The purpose of the `[MASK]` token during pre-training is to train the token embeddings to learn context so that it can predict the token labels. Since we are not training the token embeddings to learn context, masking the tokens does not help much for active learning. We use active learning for fine-tuning, so the input should be in the same format for active learning and fine-tuning. Otherwise, there is a mismatch between the two stages.

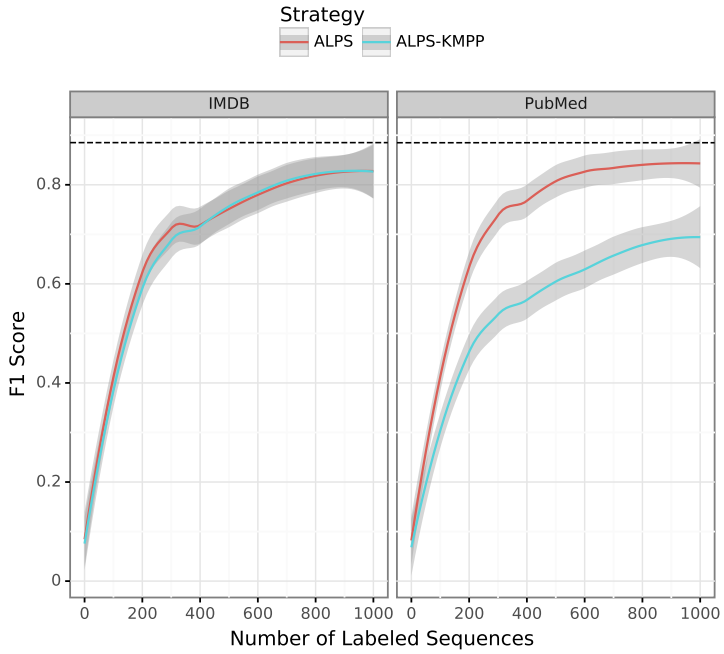
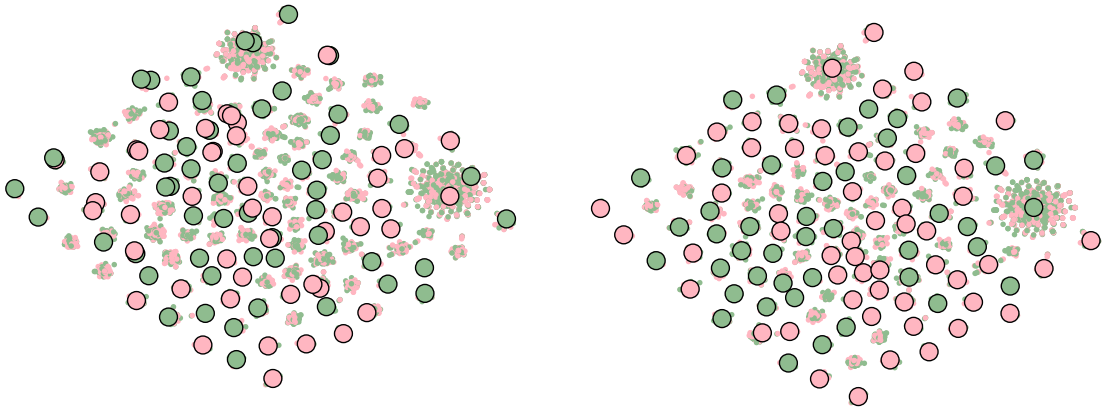


Figure 4.6: Comparing validation accuracy between using k-MEANS and k-MEANS++ to select centroids in the surprisal embeddings. Using k-MEANS reaches higher accuracy.

Token sampling percentage When BERT evaluates MLM loss, it only focuses on the masked tokens, which are from a 15% random subsample of tokens in the sentence. We experiment with varying this subsample percentage on the validation set (Table 4.5). We try sampling 10%, 15%, 20%, and 100%. Overall, we notice that mean accuracy are roughly the same, but variance in accuracy across different runs is slightly higher for percentages other than 15%.

After the second active learning iteration, we notice that accuracy mean and variance between the different token sampling percentages converge. So, the token sampling percentage makes more of a difference in early stages of active learning. Devlin et al. (2019) show that the difference in accuracy between various mask strategies is minimal for fine-tuning BERT. We believe this can also be applied to what we have observed for ALPS.



(a) Surprisal embeddings with k-MEANS++ centers

(b) Surprisal embeddings with k-MEANS centers

Figure 4.7: T-SNE plots of surprisal embeddings for IMDB training data. The centers are either picked by k-MEANS++ (left) or k-MEANS (right). There is less overlap between the centers with k-MEANS compared to k-MEANS++. So, using k-MEANS is better for exploiting diversity in the surprisal embedding space.

Clustering technique BADGE applies k-MEANS++ on gradient embeddings to select points to query. Initially, we also use k-MEANS++ on the surprisal embeddings but validation accuracy is only slightly higher than random sampling. Since k-MEANS++ is originally an algorithm for robust initialization of k-MEANS, we instead apply k-MEANS on the surprisal embeddings. As a result, we see more significant increase in accuracy over baselines, especially for PubMed (Figure 4.6). Additionally, the t-SNE plots show that k-MEANS selects centers that are further apart compared to the ones chosen by k-MEANS++ (Figure 4.7). This shows that k-MEANS can help sample a more diverse batch of data.

4.7 Conclusion

While transformers are powerful models for inductive transfer learning, their accuracy and stability require fine-tuning on large amounts of data. Active learning can help direct limited an-

notations most effectively so that labels complement, rather than duplicate, unlabeled data. Since the pre-training loss is useful for inductive transfer learning, ALPS uses the language modeling loss to find examples that surprise the model. Thus, ALPS is a strategy that samples data for improved model adaptation to downstream tasks.

Next, we transition to transductive transfer learning. In Chapter 3, we have already seen how user interaction can help align topics across languages. We extend that work and solely focus on improving cross-lingual text classification. We develop CLIME, a system for updating word embeddings based on user feedback about keywords. The bilingual speakers annotate nearest neighbors of words that are most salient for the classification task. Compared to MTAnchor, the feedback is more tailored toward text classification in low-resource situations.

Part II

Interactive Feedback for Transductive Transfer Learning

Chapter 5: Refining Word Embeddings across Languages[1]

The last chapter focuses on user interaction for cross-lingual topic modeling. This chapter considers a more difficult problem. We still look at cross-lingual transfer but primarily focus on low-resource languages. Several languages around the world do not have enough data to train models with simply supervised learning. Here, we need to make use of innnovations like cross-lingual word embeddings (Section 2.2.2.2). Using CLWE features, models trained in a resource-rich language (e.g., English) can predict labels for other languages.

CLWE depend on quality of dictionary entries (Vulić and Korhonen, 2016) and training data (Søgaard et al., 2018). If CLWE do not transfer enough information for low-resource languages, how else can we improve these embeddings? One way is to employ a bilingual speaker in the loop to provide linguistic and cultural knowledge. We develop **CL**assifying **I**nteractively with **M**ultilingual **E**mbeddings (CLIME), that efficiently specializes CLWE with *human interaction*.[2] [3] CLIME builds upon active learning to obtain word-level feedback, rather than instance-level label-

[1]Michelle Yuan, Mozhi Zhang, Benjamin Van Durme, Leah Findlater, and Jordan Boyd-Graber. 2020b. Interactive refinement of cross-lingual word embeddings. In *Proceedings of Empirical Methods in Natural Language Processing*

[2]`https://github.com/forest-snow/clime-ui`.

[3]My contributions to this work involve building the human-in-the-loop framework for retrofitting embeddings, designing the CLIME interface, and running the user study experiments.

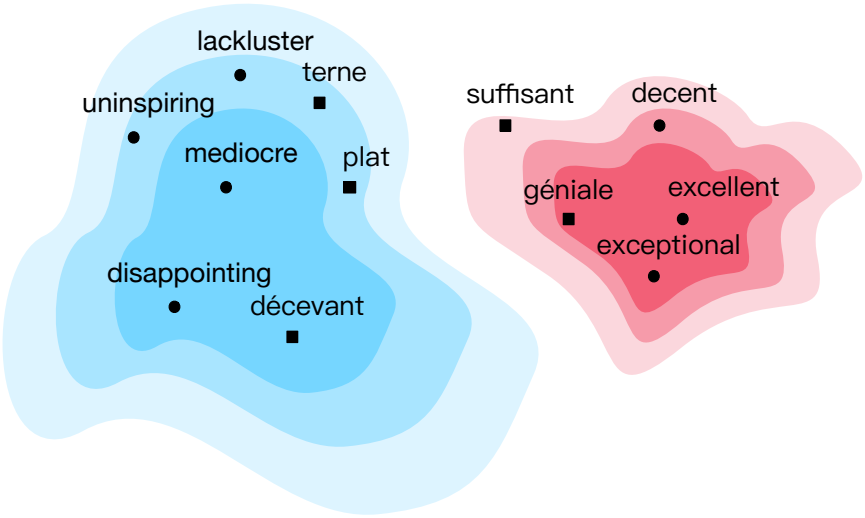


Figure 5.1: A hypothetical topographic map of an English–French embedding space tailored for sentiment analysis. Dots are English words, and squares are French words. Positive sentiment words are grouped together (red), while negative sentiment words are placed together (blue).

ing, that can modify the word embedding space. First, CLIME uses loss gradients in downstream tasks to find keywords with high salience (Section 5.1.1). Focusing on these keywords allows the bilingual user to most efficiently refine CLWE by marking their similarity or dissimilarity (Section 5.1.2). After collecting annotations, CLIME pulls similar words closer and pushes dissimilar words apart (Section 5.2). Feedback from bilingual users establishes desired geometry in the embedding space (Figure 5.1).

In our user study, we compare CLIME with an active learning baseline that asks a user to label target language documents. Under the same annotation time constraint, CLIME often has higher accuracy. Furthermore, the two methods are complementary. Combining active learning with CLIME increases accuracy even more, and the user-adapted model is competitive with a large, resource-hungry multilingual transformer (Conneau et al., 2020). The results show that human-AI interaction stimulates effective cross-lingual transfer in low-resource settings.

5.1 Interactive Neighborhood Reshaping

This section introduces the interface designed to solicit human feedback on neighborhoods of CLWE and our keyword selection criterion. Suppose that we have two languages with vocabulary $\mathcal{V}_1$ and $\mathcal{V}_2$. Let $\boldsymbol{E}$ be a pre-trained CLWE matrix, where $\boldsymbol{E}_w$ is the vector representation of word type w in the joint vocabulary $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$. Our goal is to help a bilingual novice (i.e., not a machine learning expert) improve the CLWE $\boldsymbol{E}$ for a downstream task through inspection of neighboring words.

5.1.1 Keyword Selection

With limited annotation time, users cannot vet the entire vocabulary. Instead, we need to find a small salient subset of *keywords* $\mathcal{K} \subseteq \mathcal{V}$ whose embeddings, if vetted, would most improve a downstream task. For example, if the downstream task is sentiment analysis, our keywords set should include sentiment words such as "good" and "bad". Prior work in active learning solicits keywords using information gain (Settles, 2011), but this cannot be applied to continuous embeddings. Li et al. (2016) suggest that the contribution of one dimension of a word embedding to the loss function can be approximated by the absolute value of its partial derivative, and therefore they use partial derivatives to visualize the behavior of neural models. However, rather than understanding the importance of individual dimensions, we want to compute the salience of an *entire word vector*. Therefore, we extend their idea by defining the salience of a word embedding as the *magnitude* of the loss function's gradient. This score summarizes salience of all dimensions from a word embedding. Formally, let $\boldsymbol{x} = \langle w_1, w_2, \cdots, w_n \rangle$ be a document of n words with label y; let L be the training loss function of the downstream classifier f. We measure the

example-level salience of word w_i in document $\boldsymbol{x}$ as

$$S_{\boldsymbol{x}}(w_i) = \left\| \nabla_{\boldsymbol{E}_{w_i}} L(f(\boldsymbol{x}), y) \right\|_2 . \tag{5.1}$$

Equation 5.1 measures the local contribution of a token in one document, but we are interested in the global importance of a word type across many documents. To compute the global salience score of a word type w, we add example-level salience scores of all token occurrences of a word type w in a large labeled dataset $\boldsymbol{X}$ and multiply by the inverse document frequency (IDF) of w:

$$S(w) = \text{IDF}(w, \boldsymbol{X}) \cdot \sum_{\boldsymbol{x} \in \boldsymbol{X} : w \in \boldsymbol{x}} S_{\boldsymbol{x}}(w). \tag{5.2}$$

The IDF term is necessary because it discounts *stop words* with high document frequency (e.g., "the" and "of"). These words are often irrelevant to the downstream task and thus have low example-level salience, but they have high total salience because they appear in many examples.

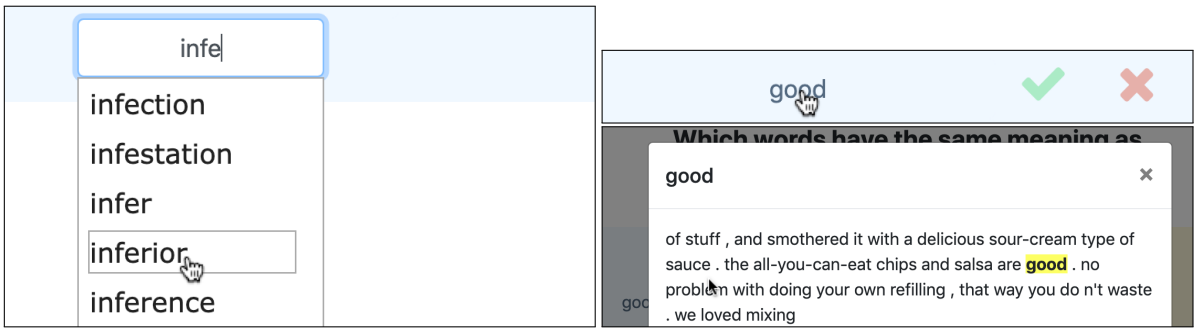
Based on Equation 5.2, we choose the top-s most salient words as the keyword set $\mathcal{K}$. The hyperparameter s is the number of keywords displayed to the user, which controls the length of a CLIME session. We limit s to fifty in experiments.

5.1.2 User Interaction

For each keyword k, we want to collect a positive set $\mathcal{P}_k$ with semantically similar words, and a negative set $\mathcal{N}_k$ with unrelated words. To specialize embeddings for a classification task, we ask the user to consider semantic similarity as *inducing a similar label*. For example, if the task is English–French sentiment analysis, then "good" should be considered similar to "excellent" and

Figure 5.2: The CLIME interface displays a keyword on top while its nearest neighbors in the two languages appear in the two columns below. A user can accept or reject each neighbor, and add new neighbors by typing them in the "add word" textboxes. They may also click on any word to read its context in the training set.

"génial" but dissimilar to "bad" and "décevant". On the interface, the keyword k is displayed on the top, and its nearest neighbors in the two languages are arranged in two columns (Figure 6.8). The neighbors are the words w with embeddings $\boldsymbol{E}_w$ closest to $\boldsymbol{E}_k$ in cosine similarity. The number of displayed nearest neighbors can be adjusted as a hyperparameter, which also controls the session length. For each nearest neighbor, the user can either: (1) press on the green checkmark to add a positive neighbor to $\mathcal{P}_k$, (2) press on the red "X" mark to add a negative neighbor to $\mathcal{N}_k$, or (3) leave an uncertain neighbor alone. The "add word" textbox lets the user add words that are not in the current neighbor list (Figure 5.3a). The added word can then be marked as positive or negative. Section 5.2 explains how CLIME refines the embeddings with the feedback sets $\mathcal{P}$ and $\mathcal{N}$. The interface also provides a word concordance—a brief overview of the contexts where a



(a) Adding new words (b) Getting concordance

Figure 5.3: Features on the interface that help users find similar words.

word appears—to disambiguate and clarify words. Users can click on any word to find example sentences (Figure 5.3b).

5.2 Fitting Word Embeddings to Feedback

After receiving user annotations, CLIME updates the embeddings to reflect their feedback. The algorithm reshapes the neighborhood so that words near a keyword share similar semantic attributes. Together, these embeddings form desired task-specific connections between words across languages. Our update equations are inspired by ATTRACT-REPEL (Mrkšić et al., 2017), which fine-tunes word embeddings with synonym and antonym constraints. The objective in ATTRACT-REPEL pulls synonyms closer to and pushes antonyms further away from their nearest neighbors. This objective is useful for large lexical resources like BabelNet with hundreds of thousands linguistic constraints, but our pilot experiment suggests that the method is not suitable for smaller constraint sets. Since CLIME is designed for low-resource languages, we optimize an objective that reshapes the neighborhood more drastically than ATTRACT-REPEL.

5.2.1 Feedback Cost

For each keyword $k \in \mathcal{K}$, we collect a positive set $\mathcal{P}_k$ and a negative set $\mathcal{N}_k$ (Section 5.1.2). To refine embeddings $\boldsymbol{E}$ with human feedback, we increase the similarity between k and each positive word $p \in \mathcal{P}_k$, and decrease the similarity between k and each negative word $n \in \mathcal{N}_k$. Formally, we update the embeddings $\boldsymbol{E}$ to minimize the following:

$$C_f(\boldsymbol{E}) = \sum_{k \in \mathcal{K}} \left(\sum_{n \in \mathcal{N}_k} \boldsymbol{E}_k^\top \boldsymbol{E}_n - \sum_{p \in \mathcal{P}_k} \boldsymbol{E}_k^\top \boldsymbol{E}_p \right), \tag{5.3}$$

where $\boldsymbol{E}_k^\top \boldsymbol{E}_n$ measures the similarity between the keyword k and a negative word n, and $\boldsymbol{E}_k^\top \boldsymbol{E}_p$ measures the similarity between the keyword k and a positive word p. Minimizing C_f is equivalent to maximizing similarities of positive pairs while minimizing similarities of negative pairs.

5.2.2 Topology-Preserving Regularization

Prior retrofitting methods for word embeddings emphasize regularization to maintain the topology—or properties that should be preserved under transformations—of the embedding space (Section 2.2.2.2). If the original CLWE brings certain translations together, those translated words should remain close after updating the embeddings. The topology also encodes important semantic information that should not be discarded. Therefore, we also include the following regularization term:

$$R(\boldsymbol{E}) = \sum_{w \in \mathcal{V}} \left\| \hat{\boldsymbol{E}}_w - \boldsymbol{E}_w \right\|_2^2. \tag{5.4}$$

Minimizing $R(\boldsymbol{E})$ prevents $\boldsymbol{E}$ from drifting too far away from the original embeddings $\hat{\boldsymbol{E}}$.

The final cost function combines the feedback cost (Equation 5.3) and the regularizer

(Equation 5.4):

$$C(\boldsymbol{E}) = C_f(\boldsymbol{E}) + \lambda R(\boldsymbol{E}), \tag{5.5}$$

where the hyperparameter λ controls the strength of the regularizer. The updated embeddings enforce constraints from user feedback while preserving other structures from the original embeddings. After tuning in a pilot user study, we set λ to one. We use the Adam optimizer (Kingma and Ba, 2015) with default hyperparameters.

5.3 Cross-Lingual Classification Experiments

We evaluate CLIME on cross-lingual document-classification (Klementiev et al., 2012), where we build a text classifier for a low-resource target language using labeled data in a high-resource source language through CLWE. Our task identifies whether a document describes a medical emergency, useful for planning disaster relief (Strassel and Tracey, 2016). The source language is English and the four low-resource target languages are Ilocano, Sinhalese, Tigrinya, and Uyghur.

Our experiments confirm that a bilingual user can quickly improve the test accuracy of cross-lingual models through CLIME. Alternatively, we can use active learning to have an annotator label more training documents in the target language (Section 2.3.1). Therefore, we compare CLIME to an active learning baseline that queries the user for document labels; CLIME often improves accuracy faster. Then, we combine CLIME and active learning to show an even faster improvement of test accuracy.

Comparing active learning to CLIME may seem unfair at first glance. In theory, document labeling only requires target language knowledge, while CLIME learns from a bilingual user. In

Ilocano	... Nagtalinaed dagiti pito a balod ti Bureau of Jail Management and Penology (BJMP) ditoy ciudad ti Laoag iti isolation room gapo iti tuko ...
English	... Seven inmates from the Bureau of Jail Management and Penology (BJMP), Laoag City, have been transferred to the isolation room due to chicken pox ...

Table 5.1: Excerpt of a positive Ilocano test example (top) and its English translation (bottom) that describes a medical emergency.

practice, researchers who speak a high-resource language provide instructions to the annotator and answer their questions, so bilingual knowledge is usually required in document labeling for low-resource languages. Moreover, CLIME is complementary to active learning, as combining them gives the highest accuracy across languages.

We also experiment with refining the same set of keywords with multiple rounds of user interaction. The repeated sessions slightly improve test accuracy on average. Finally, we compare with XLM-R (Conneau et al., 2020), a state-of-the-art multilingual transformer. Despite using fewer resources, CLIME has competitive results.

5.3.1 Experiment Setup

Labeled Data We train models on 572 English documents and test on 48 Ilocano documents, 58 Sinhalese documents, 158 Tigrinya documents, and 94 Uyghur documents. The documents are extracted from LORELEI language packs (Strassel and Tracey, 2016), a multilingual collection of documents of emergencies with a public health component.[4] To simplify the task, we consider a binary classification problem of detecting whether the documents are associated with medical needs. Table 5.1 shows an example document. To balance the label distribution, we sample an

[4]Download from `https://www.ldc.upenn.edu`

equal number of negative examples.

Word Embeddings To transfer knowledge between languages, we build CLWE between English and each target language. We experiment with two methods to pre-train CLWE: (1) train monolingual embeddings with word2vec (Mikolov et al., 2013c) and align with CCA (Faruqui et al., 2015; Ammar et al., 2016), (2) train monolingual embeddings with fastText (Bojanowski et al., 2017) and align with RCSLS (Joulin et al., 2018). The English embeddings are trained on Wikipedia and the target language embeddings are trained on unlabeled documents from the LORELEI language packs. For alignment, we use the small English dictionary in each pack. Low-resource language speakers are hard to find, so we do not try all combinations of languages and CLWE: we use CCA embeddings for Tigrinya and Uyghur, RCSLS embeddings for Ilocano. Since Sinhalese speakers are easier to find, we experiment with both CLWE for Sinhalese.

Text Classifier Our classifier is a convolutional neural network (Kim, 2014). Each document is represented as the concatenation of word embeddings and passed through a convolutional layer, followed by max-pooling and a final softmax layer. To preserve cross-lingual alignments, we freeze embeddings during training. This simple model is effective in low-resource cross-lingual settings (Chen et al., 2018b). We minimize cross-entropy on the training set by running Adam (Kingma and Ba, 2015) with default hyperparameters for thirty epochs. All experiments use GeForce GTX 1080 GPU and 2.6 GHz AMD Opteron 4180 processor.

User Study We use Upwork to hire participants who are fluent in both English and the target language.[5] Low-resource language speakers are hard to find, so we have a different number of

[5] https://upwork.com/

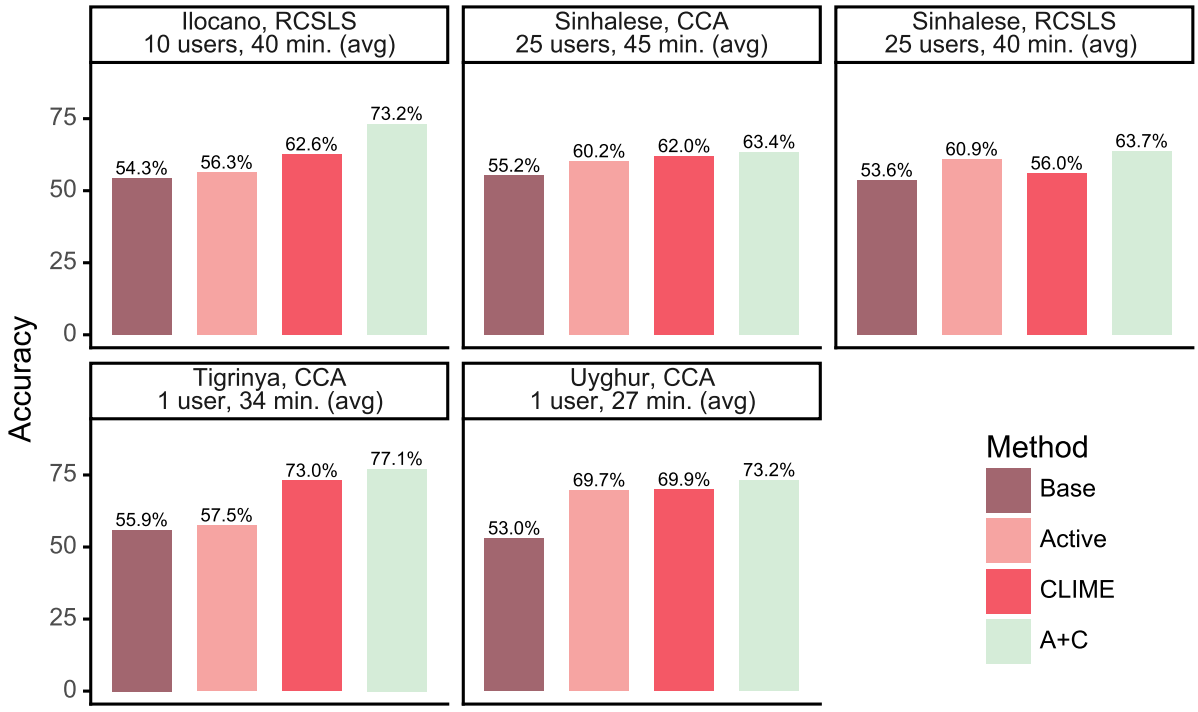


Figure 5.4: Test accuracy on four target languages and two CLWE methods. The Sinhalese and Ilocano results are averaged over multiple users, while we only have one user for other languages. Each subcaption indicates the target language, embedding alignment, number of users, and average time per user. **CLIME** has higher accuracy than **Active** on four of the five embeddings, and the combined **A+C** model has the highest.

users for each language. We hire ten Ilocano users and twenty-five Sinhalese users. For additional case studies, we hire one Tigrinya user and one Uyghur user. Each user annotates the fifty most salient keywords, which takes less than an hour (Figure 5.4). For each keyword, we show five nearest neighbors for each language. Each user provides about nine constraints for each keyword.

5.3.2 Comparisons

After receiving feedback, we update the embeddings (Section 5.2). We evaluate the new embeddings by retraining a classifier. For each set of embeddings, we train ten models with different random seeds and report average test accuracy.

We compare a classifier trained on the updated embeddings (**CLIME** in Figure 5.4) against two baselines. The first baseline is a classifier trained on original embeddings (**Base** in Figure 5.4). If we have access to a bilingual speaker, an alternative to using CLIME is to annotate more training documents in the target language. Therefore, we also compare CLIME to uncertainty sampling (Lewis and Gale, 1994), an active learning method that asks a user to label documents (**Active** in Figure 5.4). We choose a set of fifty documents where model outputs have the highest entropy from a set of unlabeled *target language* documents and ask an annotator to label them as additional training documents. We then retrain a model on both the English training set and the fifty target language documents, using the original embeddings. For each model, a human annotator labels fifty documents within forty to fifty minutes. This can either be slower or take approximately the same time as an average CLIME session (Figure 5.4). Thus, any improvements in accuracy using CLIME are even more impressive given that **Active** is no faster than CLIME.

Finally, we explore combining active learning and CLIME (**A+C** in Figure 5.4). Document-level and word-level interactions are complementary, so using both may lead to higher accuracy. To keep the results comparable, we allocate half of the user interaction time to active learning, and the other half to CLIME. Specifically, we use active learning to expand the training set with twenty-five target language documents and refine the embeddings by running CLIME on only twenty-five keywords. Then, we retrain a model using both the augmented training set and the refined embeddings.

Comparison	Model	p	t	df
CLIME vs. **Base**	SI(CCA)	< 0.01	7.64	24
	SI(RCSLS)	< 0.01	3.62	24
	IL(RCSLS)	< 0.01	5.16	9
CLIME vs. **Active**	SI(CCA)	0.07	2.00	24
	SI(RCSLS)	< 0.01	-7.09	24
	IL(RCSLS)	< 0.01	3.96	9
A+C vs. **Active**	SI(CCA)	< 0.01	4.297	24
	SI(RCSLS)	< 0.01	3.40	24
	IL(RCSLS)	< 0.01	13.97	9

Table 5.2: Results of single-sample t-tests between CLIME and **Base**, CLIME and **Active**, and **A+C** and **Active**, showing the p-value, the t statistic, and the degree of freedoms df. CLIME is significantly better than **Base**, and **A+C** is significantly better than **Active** across different languages and embedding models. The only combination with results that are not significantly different is CLIME and **Active** for Sinhalese (CCA).

5.3.3 Results and Analysis

Effectiveness of CLIME Figure 5.4 compares the four methods described in the previous section. CLIME is effective in this low-resource setting. On all four target languages, the classifier trained on embeddings refined by CLIME has higher accuracy than the classifier that trains on the original embeddings: CLIME reshapes embeddings in a way that helps classification. CLIME also has higher accuracy than active learning for most users. The combined method has the highest accuracy: active learning and CLIME are complementary. Single-sample t-tests confirm that **CLIME** is significantly better than **Base** and **A+C** is significantly better than **Active** (Table 5.2).

Keyword Detection We inspect the list of the fifty most salient keywords (Section 5.1.1). Most keywords have obvious connections to our classification task of detecting medical emergencies, such as “ambulance”, “hospitals”, and “disease”. However, the list also contains some words that are unrelated to a medical emergency, including “over” and “given”. These words may be biases

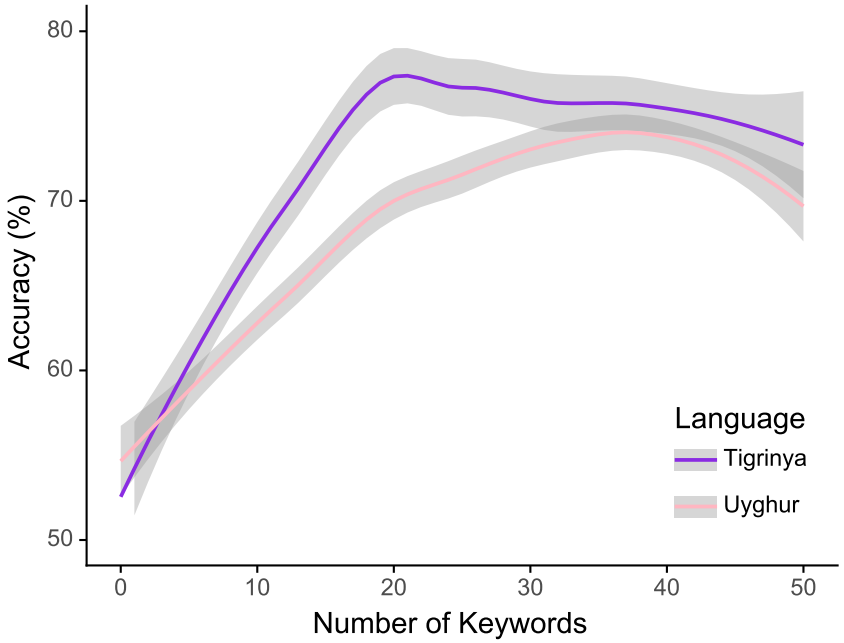


Figure 5.5: For Uyghur (pink) and Tigrinya (purple), we compare test accuracy between sets of CLWE that differ in the number of keywords used to refine them. The leftmost point corresponds to the **Base** model in Figure 5.4, while the rightmost point corresponds to the **CLIME** model. Test accuracy generally improves with more feedback at the beginning but slightly drops after reaching an optimal number of keywords.

or artifacts from training data (Feng et al., 2018).

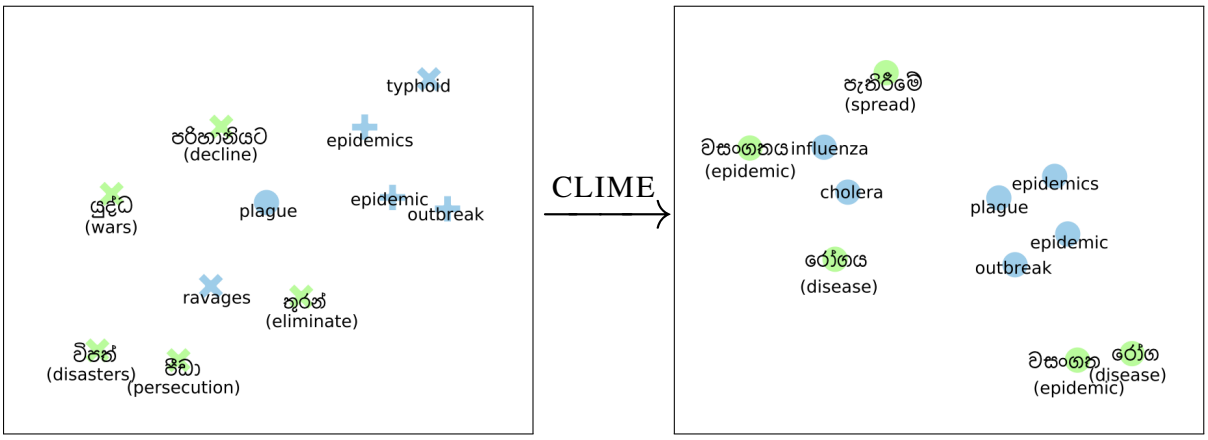
Number of Keywords To evaluate how feedback quantity changes accuracy, we vary the number of keywords and compare test accuracy on Tigrinya and Uyghur datasets (Figure 5.5). For each keyword s from one to fifty, we update the original embeddings using only the feedback on the top-s keywords and evaluate each set of embeddings with test accuracy. For both languages, test accuracy generally increases with more annotation at the beginning of the session. Interestingly, test accuracy plateaus and slightly drops after reaching an optimal number of keywords, which is around twenty for Tigrinya and about forty for Uyghur. One explanation is that the later keywords are less salient, which causes the feedback to become less relevant. These redundant constraints hamper optimization and slightly hurt test accuracy.

Comparing with Contextual Embeddings Contextualized embeddings based on multilingual transformers reach state-of-the-art in many tasks, so we compare CLIME with these models. Most existing models (Lample and Conneau, 2019) do not cover our low-resource languages. The only exception is XLM-R (Conneau et al., 2020), which covers Uyghur and Sinhalese. To compare with CLIME, we fine-tune XLM-R for three epochs with AdamW (Loshchilov and Hutter, 2019), batch size of sixteen, and learning rate of 2e-5. We compute average accuracy over ten runs with different random seeds.

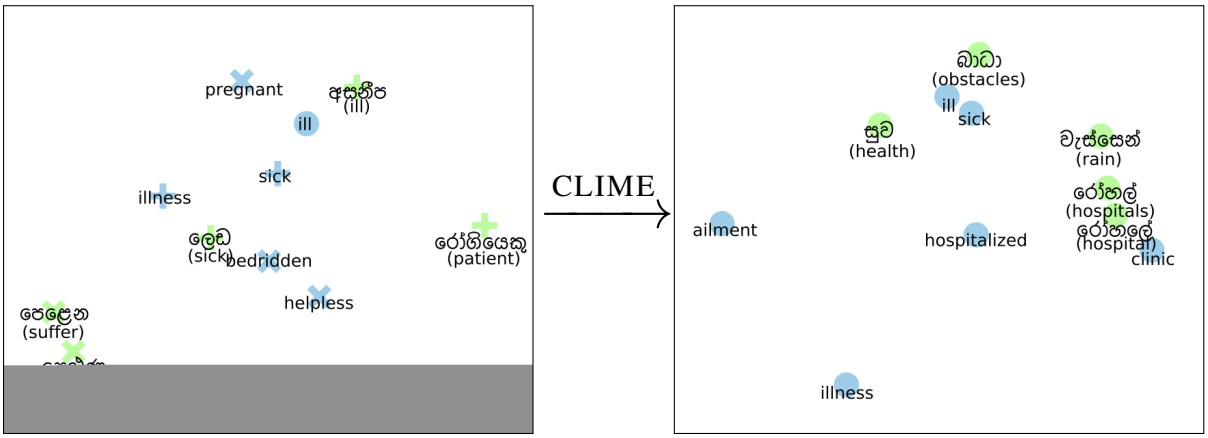
For Uyghur, XLM-R has lower accuracy than our **A+C** approach (71.7% vs. 73.2%). This is impressive given that XLM-R uses much more resources: 270 million parameters, 2.5TB of multilingual Common Crawl data, and 500 GPUs. In contrast, the **A+C** model has 120K parameters and is built in less than two hours with a single GPU (including human interaction and model training).

For Sinhalese, XLM-R has higher accuracy than our **A+C** approach (69.3% vs. 63.7%). Common Crawl has much more Sinhalese words than Uyghur words. This aligns with our intuition: CLIME is more useful in low-resource settings, whereas multilingual transformers are more appropriate for languages with more data. Future work can extend the interactive component of CLIME to multilingual transformers.

Qualitative Analysis To understand how CLIME updates the embeddings, we visualize changes in the neighborhoods of keywords with t-SNE (Maaten and Hinton, 2008). All embeddings from before and after the user updates are projected into the same space for fair distance comparison. We inspect the user updates to the Sinhalese CCA embeddings (Figure 5.6). We confirm that positive neighbors are pulled closer and negative neighbors are pushed further away. The user marks



(a) Neighborhood of "plague"



(b) Neighborhood of "ill"

Figure 5.6: T-SNE visualization of embeddings before (left) and after (right) CLIME updates. From one Sinhalese user study, we inspect two keywords, "ill" and "plague", and their five closest neighbors in English (blue) and Sinhalese (green). The Sinhalese words are labeled with English translations. Shape denotes the type of feedback: "+" for positive neighbors and "x" for negative neighbors.

"epidemic" and "outbreak" as similar to the keyword "plague", and these words are closer after updates (Figure 5.6a). For the keyword "ill", the user marks "helpless" as a negative neighbor, because "helpless" can signal other types of situations and is more ambiguous for detecting a medical emergency. After the update, "helpless" is pushed away and disappears from the nearest neighbors of "ill" (Figure 5.6b). However, a few positive neighbors have inadvertently moved away, such as the Sinhalese translation for "ill". The update algorithm tries to satisfy constraints

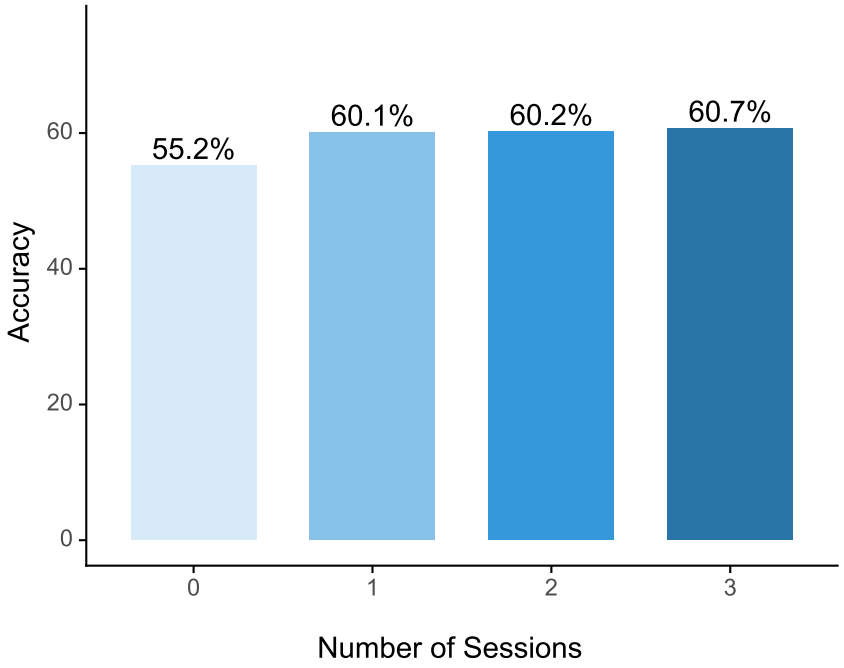


Figure 5.7: Progress of five Sinhalese users over three CLIME sessions. Largest increase in test accuracy occurs after first session. The leftmost point is the **Base** model from Figure 5.4. Average accuracy for first session is not the same as Figure 5.4 because only a subset of users are asked to complete three sessions.

for multiple keywords, so soft constraints may be overlooked. This motivates repeated CLIME sessions where the user can continue fixing errors.

5.3.4 Repeating User Sessions

We investigate the effects of having a user complete multiple CLIME sessions. After the user finishes a session, we fit the embeddings to their feedback, produce a new vocabulary ranking, and update the interface for the next session. We experiment on the Sinhalese dataset with CCA embeddings and ask five users to complete three sessions of fifty keywords. Average test accuracy increases with more sessions, but the improvement is marginal after the first session (Figure 5.7). By the end of the three sessions, one user reaches 65.2% accuracy, an improvement from the

55.2% baseline.

5.4 Conclusion

Cross-lingual learning is challenging in the low-resource setting where no labeled data exists for the target language. In our experiments, we use an emergency relief dataset in low-resource languages to simulate an urgent situation for transductive transfer learning. Through CLIME, we facilitate cross-lingual model transfer with feedback from bilingual user. We compare CLIME to standard active learning. Our results reveal that annotators can achieve higher accuracy on the target data with CLIME. Furthermore, CLIME is complementary with active learning. Obtaining both word-level and instance-level feedback refines models that are significantly better than retrieving only instance-level feedback. Therefore, future work should focus on the type of user interaction needed to improve transfer learning in NLP.

So far, the past chapters have applied interactive feedback to text classification. However, many NLP tasks are more complex. For instance, coreference resolution requires predicting antecedents of individual entity mentions in sentences. To solve this problem, the model needs to detect text spans that are entity mentions and link them to the correct entity clusters. In the next chapter, we use active learning to adapt coreference resolution models across domains.

Chapter 6: Transferring Coreference Resolution across Domains[1]

Chapter 3 discusses interaction in topic models and Chapter 5 talks about user feedback for word embeddings. This chapter considers domain shift for a more complex task, coreference resolution. This task involves discovering spans of text mentions that refer to the same entity (CR). Neural models (Lee et al., 2018; Joshi et al., 2020) are state-of-the-art (SOTA) on ONTONOTES 5.0 (Pradhan et al., 2013) but cannot immediately generalize to other datasets. Generalization is difficult because domains differ in content, writing style, and annotation guidelines. To overcome these challenges, models need copiously labeled, in-domain data (Bamman et al., 2020).

Despite expensive labeling costs, adapting CR is crucial for downstream applications like question answering and information extraction. In Figure 6.1, knowing that "the bonds" refers to municipal bonds could inform the model about what was gained by Goldman Sachs. Ideally, we would like to quickly and cheaply adapt the model without repeatedly relying on an excessive amount of annotations to retrain the model. To reduce labeling cost, we investigate active learning approaches. For example, a word like "the bonds" refers to municipal bonds in ONTONOTES but links to "chemical bonds" in another domain (Figure 6.1). If users annotate the antecedents of

[1]Michelle Yuan, Patrick Xia, Chandler May, Benjamin Van Durme, and Jordan Boyd-Graber. 2022. Adapting coreference resolution models through active learning. In *Proceedings of the Association for Computational Linguistics*

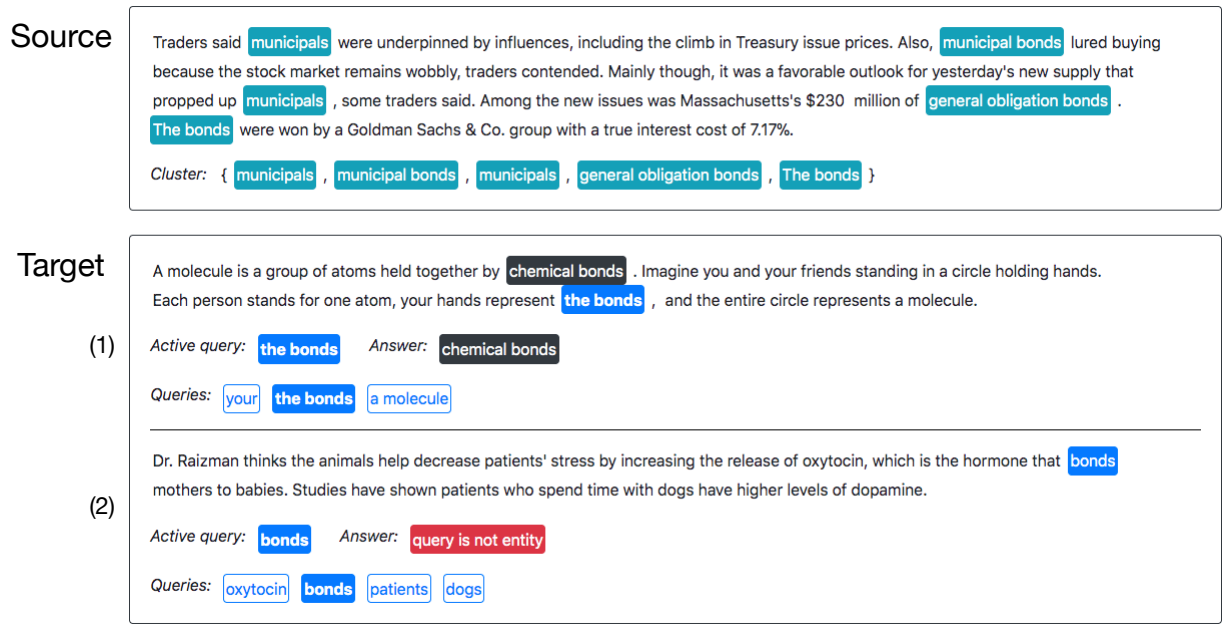


Figure 6.1: CR models are trained on **source** domain ONTONOTES, which contains data like news articles. The **source** document links "the bonds" to "municipal bonds". In a **target** domain like PRECO (Chen et al., 2018a), "the bonds" may no longer have the same meaning. It can refer to "chemical bonds" (Document 1) or not be considered an entity (Document 2). A solution is to continue training the **source** model on more spans from the **target** domain. Active learning helps select ambiguous spans, like "the bonds", for the user to label on this interface (Section 6.3.2).

"the bonds" and other ambiguous entity mentions, then these labels help adapt a model trained on ONTONOTES to new domains.

6.1 Problem: Adapting Coreference

C2F-COREF Lee et al. (2018) introduce C2F-COREF, a neural model that outperforms prior rule-based systems. It assigns an antecedent y to mention span x. The set $\mathcal{Y}(x)$ of possible antecedent spans include a dummy antecedent ϵ and all spans preceding x. If span x has no antecedent, then x should be assigned to ϵ. Given entity mention x, the model learns a distribution

over its candidate antecedents in $\mathcal{Y}(x)$,

$$P(Y = y) = \frac{\exp\{s(x, y)\}}{\sum_{y' \in \mathcal{Y}(x)} \exp\{s(x, y')\}}. \tag{6.1}$$

The scores $s(x, y)$ are computed by the model's pairwise scorer. The model's pairwise scorer judges whether span x and span y are coreferent based on their antecedent score s_a and individual mention scores s_m,

$$s(x, y) = \begin{cases} 0 & y = \epsilon \\ s_m(x) + s_m(y) + s_a(x, y) & y \neq \epsilon \end{cases}, \tag{6.2}$$

Suppose $\boldsymbol{g_x}$ and $\boldsymbol{g_y}$ are the span representations of x and y, respectively. Mention scores and antecedent scores are then computed with feedforward networks $FFNN_m$ and $FFNN_c$,

$$s_m(x) = FFNN_m(\boldsymbol{g_x}) \tag{6.3}$$

$$s_a(x, y) = FFNN_a(\boldsymbol{g_x}, \boldsymbol{g_y}, \phi(x, y)). \tag{6.4}$$

The input $\phi(x, y)$ includes features like the distance between spans. The unary mention score s_m can be viewed as the likelihood that the span is an entity mention. For computational purposes, the C2F-COREF model only retains top-k spans with the highest unary mention scores. Lee et al. (2018) provide more details about the pairwise scorer and span pruning.

Transfer of Coreference Resolution CR models like C2F-COREF are typically trained on ONTONOTES. Recent work in CR improves upon C2F-COREF and has SOTA results on ONTONOTES (Wu et al., 2020; Joshi et al., 2020). However, annotation guidelines and the underlying text differ across

domains. As a result, these CR models cannot immediately transfer to other datasets. For different domains, spans could hold different meanings or link to different entities. Xia and Van Durme (2021) show the benefits of *continued training* where a model trained on ONTONOTES is further trained on the target dataset. For several target domains, continued training from ONTONOTES is stronger than training the model from scratch, especially when the training dataset is small. Their experiments use an incremental variant of C2F-COREF called ICOREF (Xia et al., 2020).

ICOREF While C2F-COREF requires $\Theta(n)$ memory to simultaneously access all spans in the document and infer a span's antecedent, ICOREF only needs constant memory to predict a span's entity cluster. Despite using less space, ICOREF retains the same accuracy as C2F-COREF. Rather than assigning x to antecedent y, ICOREF assigns x to cluster c where c is from a set of observed entity clusters $\mathcal{C}$,

$$P(C = c) = \frac{\exp\{s(x, c)\}}{\sum_{c' \in \mathcal{C}} \exp\{s(x, c')\}}. \tag{6.5}$$

As the algorithm processes spans in the document, each span is either placed in a cluster from $\mathcal{C}$ or added to a new cluster. To learn the distribution over clusters (Equation 6.5), the algorithm first creates a cluster representation $\boldsymbol{g_c}$ that is an aggregate of span representation that is an aggregate of span representations over spans that currently exist in the cluster. (Equation 6.6). With cluster and span representations, individual spans and entity clusters are mapped into a shared space. Then, we can compute $s(x, c)$ using the same pairwise scorer as Lee et al. (2018). In the example from Figure 6.1, the cluster representation of "municipal bonds" should be close to the span representation of "the bonds", so the model would predict "municipal bonds" as the most likely cluster of "the bonds".

Suppose that model predicts c^* as most likely cluster: $c^* = \arg\max_{c \in \mathcal{C}} s(x, c)$. Now, the

algorithm makes one of two decisions:

1. If $s(x, c^*) > 0$, then x is assigned to c^* and update $\boldsymbol{g_{c^*}}$ such that

$$\boldsymbol{g_{c^*}} = s_e(c^*, x)\boldsymbol{g_{c^*}} + (1 - s_e(c^*, x))\boldsymbol{g_x}, \tag{6.6}$$

   where s_e is a learned weight.

2. If $s(x, c^*) \leq 0$, then a new entity cluster $c_x = \{x\}$ is added to $\mathcal{C}$.

The algorithm repeats for each span in the document. Using the same example from Figure 6.1, the model would place "the bonds" in the "municipal bonds" cluster and then update the cluster representation for "municipal bonds".

Like C2F-COREF, the ICOREF model only retains top-k spans with the highest unary mention score. All of our active learning baselines (Section 6.3), except **random**, sample spans from this top-k pool of spans.

Xia and Van Durme (2021) show that continued training is useful for domain adaptation but assume that labeled data already exist in the target domain. However, model transfer is more critical when annotations are scarce. Thus, the question becomes: how can we adapt CR models without requiring a large, labeled dataset? This chapter investigates active learning as a potential solution. Through active learning, we reduce labeling costs by sampling and annotating a small subset of ambiguous spans.

6.2 Method: Active Learning

Neural models achieve high accuracy for ONTONOTES but cannot quickly adapt to new datasets because of shifts in domain or annotation standards (Poot and van Cranenburgh, 2020). To transfer to new domains, models need substantial in-domain, labeled data. In low-resource situations, CR is infeasible for real-time applications. To reduce the labeling burden, active learning may target spans that most confuse the model. Active learning for domain adaptation (Rai et al., 2010) typically proceeds as follows: begin with a model trained on source data, sample and label k spans from documents in the target domain based on a strategy, and train the model on labeled data.

This labeling setup may appear straightforward to apply to CR, but there are some tricky details. The first complication is that—unlike text classification—CR is a *clustering* task. Early approaches in active learning for CR use *pairwise annotations* (Miller et al., 2012; Sachan et al., 2015). Pairs of spans are sampled, and the annotator labels whether each pair is coreferent. The downside to pairwise annotations is that it requires many labels. To label the antecedent of entity mention x, x must be compared to every candidate span in the document. For example, if we want to verify that each highlighted span belongs to the "municipal bonds" cluster, we would have to provide ten pairwise annotations (Figure 6.1).

Li et al. (2020) propose a new scheme called *discrete annotations*. Instead of sampling pairs of spans, the active learning strategy samples individual spans. Then, the annotator only has to find and label first antecedent of x in the document, which bypasses the multiple pairwise comparisons. In Figure 6.1, we would only need four discrete annotations to discover the "municipal bonds" cluster. Thus, we use discrete annotations to minimize labeling.

To further improve active learning for CR, we consider the following issues. First, the CR model has different scores for mention detection and linking, but prior active learning methods only considers linking. Second, labeling CR requires time to read the document context. Therefore, we explore important aspects of active learning for adapting CR: model uncertainty (Section 6.2.1), and the balance between reading and labeling (Section 6.2.2).

6.2.1 Uncertainty Sampling

A well-known active learning strategy is uncertainty sampling. A common measure of uncertainty is the entropy in the distribution of the model's predictions for a given example (Lewis and Gale, 1994). Labeling uncertain examples improves accuracy for tasks like text classification (Settles, 2009). For CR, models have multiple components, and computing uncertainty is not as straightforward. Is uncertainty over where mentions are located more important than linking spans? Or the other way around? Thus, we investigate different sources of CR model uncertainty.

Clustered Entropy To sample spans for learning CR, Li et al. (2020) propose a strategy called *clustered entropy*. This metric scores the uncertainty in the entity cluster assignment of a mention span x. If x has *high* clustered entropy, then it should be labeled to help the model learn its antecedents. Computing clustered entropy requires the probability that x is assigned to an entity cluster. Li et al. (2020) use C2F-COREF, which only gives probability of x being assigned to antecedent y. So, they define $P(C = c)$ as the sum of antecedent probabilities $P(Y = y)$,

$$P(C = c) = \sum_{y \in C \cap \mathcal{Y}(x)} P(Y = y). \tag{6.7}$$

Then, they define clustered entropy as,

$$\mathrm{H}(x) = -\sum_{c \in \mathcal{C}} P(C = c) \log P(C = c). \tag{6.8}$$

The computation of clustered entropy in Equation 6.8 poses two issues. First, summing the probabilities may not accurately represent the model's probability of linking x to c. There are other ways to aggregate the probabilities (e.g. taking the maximum). C2F-COREF never computes cluster probabilities to make predictions, so it is not obvious how $P(C = c)$ should be computed for clustered entropy. Second, Equation 6.8 does not consider mention detection. For ONTONOTES, this is not an issue because singletons (clusters of size 1) are not annotated and mention detection score is implicitly included in $P(Y = y)$. For other datasets containing singletons, the model should disambiguate singleton clusters from non-mention spans.

To resolve these issues, we make the following changes. First, we use ICOREF to obtain cluster probabilities. ICOREF is a mention clustering model so it already has probabilities over entity clusters (Equation 6.5). Second, we explore other forms of maximum entropy sampling. Neural CR models have scorers for mention detection and clustering. Both scores should be considered to sample spans that confuse the model. Thus, we propose more strategies to target uncertainty in mention detection.

Generalizing Entropy in Coreference To generalize entropy sampling, we first formalize mention detection and clustering. Given span x, assume X is the random variable encoding whether x is an entity mention (1) or not (0). In Section 6.1, we assume that the cluster distri-

bution $P(C)$ is independent of X: $P(C) = P(C \mid X)$.[2] In other words, Equation 6.5 is actually computing $P(C = c \mid X = 1)$. We sample top-k spans with the following strategies.

ment-ent Highest mention detection entropy:

$$\begin{aligned} \mathrm{H}_{\text{MENT}}(x) &= \mathrm{H}(X) \\ &= -\sum_{i=0}^{1} P(X = i) \log P(X = i). \end{aligned} \tag{6.9}$$

The probability $P(X)$ is computed from normalized mention scores s_m (Equation 6.3). **Ment-ent** may sample spans that challenge mention detection (e.g. class-ambiguous words like "park"). The annotator can clarify whether spans are entity mentions to improve mention detection. The strategy does not consider whether entity mentions are difficult to cluster.

clust-ent Highest mention clustering entropy:

$$\begin{aligned} \mathrm{H}_{\text{CLUST}}(x) &= \mathrm{H}(C \mid X = 1) \\ &= -\sum_{c \in \mathcal{C}} P(C = c \mid X = 1) \log \\ &\quad P(C = c \mid X = 1). \end{aligned} \tag{6.10}$$

Clust-ent looks at clustering scores without explicitly addressing mention detection. Unlike **ment-ent**, **clust-ent** solely focuses on how difficult it is to cluster the entity mention. For example, "flamboyance" in "The flamingos turn their heads but the flamboyance still drinks the

[2] A side effect of ONTONOTES models lacking singletons.

water" actually refers to "the flamingos" and would be hard to cluster. Like in ONTONOTES, all spans are assumed to be entity mentions. The likelihood $P(C = c \mid X = 1)$ is given by ICOREF (Equation 6.5).

cond-ent Highest conditional entropy (conditioned on mention detection):

$$\begin{aligned} \mathrm{H}_{\mathrm{COND}}(x) &= \mathrm{H}(C \mid X) \\ &= \sum_{i=0}^{1} P(X = i)\mathrm{H}(C \mid X = i) \\ &= P(X = 1)\mathrm{H}(C \mid X = 1) \\ &= P(X = 1)\mathrm{H}_{\mathrm{CLUST}}(x). \end{aligned} \tag{6.11}$$

We reach the last equation because there is no uncertainty in clustering x if x is not an entity mention and $\mathrm{H}(C \mid X = 0) = 0$. **Cond-ent** takes the uncertainty of mention detection into account. So, we may sample more pronouns because they are obviously mentions but difficult to cluster.

joint-ent Highest joint entropy:

$$\begin{aligned} \mathrm{H}_{\mathrm{JOINT}}(x) &= \mathrm{H}(X, C) = \mathrm{H}(X) + \mathrm{H}(C \mid X) \\ &= \mathrm{H}_{\mathrm{MENT}}(x) + \mathrm{H}_{\mathrm{COND}}(x). \end{aligned} \tag{6.12}$$

Joint-ent may sample spans that are difficult to detect as entity mentions *and* too confusing to cluster. This sampling strategy most closely aligns with the uncertainty of the training objective. It may also fix any imbalance between mention detection and linking (Wu and Gardner, 2021).

An example would be long phrases like "the dynamically-typed programming language with garbage collection and named after a snake" in an article about programming languages. The phrase's length makes it challenging to detect as an entity mention and link it to "Python".

6.2.2 Trade-off between Reading and Labeling

For CR, the annotator reads the document context to label the antecedent of a mention span. Annotating and reading spans from different documents may slow down labeling, but restricting sampling to the same document may cause redundant labeling (Miller et al., 2012). To better understand this trade-off, we explore different configurations with k, the number of annotated spans, and m, the maximum number of documents being read. Given source model h_0 already fine-tuned on ONTONOTES, we adapt h_0 to a target domain through active learning (Algorithm 3):

Scoring To sample k spans from unlabeled data $\mathcal{U}$ of the target domain, we score spans with an active learning strategy S. Assume S scores each span through an *acquisition model* (Lowell et al., 2019). For the acquisition model, we use h_{t-1}, the model fine-tuned from the last cycle. The acquisition score quantifies the span's importance given S and the acquisition model.

Reading Typically, active learning samples k spans with the highest acquisition scores. To constrain m, the number of documents read, we find the documents of the m spans with the highest acquisition scores and only sample spans from those documents. Then, the k sampled spans will belong to at most m documents. If m is set to "unconstrained", then we simply sample the k highest-scoring spans, irrespective of the document boundaries.

Our approach resembles Miller et al. (2012) where they sample spans based on highest un-

Algorithm 3 Active Learning for Coreference

Require: Source model h_0, Unlabeled data $\mathcal{U}$, Active learning strategy S, No. of cycles T, No. of labeled spans k, Max. no. of read docs m

Labeled data $\mathcal{L} = \{\}$
for cycles $t = 1, \ldots, T$ **do**
  $a_x \leftarrow$ Score span $x \in \mathcal{U}$ by $S(h_{t-1}, x)$
  $\mathcal{Q} \leftarrow$ Sort ($\downarrow$) $x \in \mathcal{U}$ by scores a_x
  $\mathcal{Q}_m \leftarrow$ Top-m spans in $\mathcal{Q}$
  $\mathcal{D} \leftarrow \{\boldsymbol{d}_x \mid x \in \mathcal{Q}_m\}$ where $\boldsymbol{d}_x$ is doc of x
  $\widetilde{\mathcal{Q}} \leftarrow$ Filter $\mathcal{Q}$ s.t. spans belong to $\boldsymbol{d} \in \mathcal{D}$
  $\widetilde{\mathcal{Q}}_k \leftarrow$ Top-k spans in $\widetilde{\mathcal{Q}}$
  $\mathcal{L}_k \leftarrow$ Label antecedents for $\widetilde{\mathcal{Q}}_k$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{L}_k$
  $h_t \leftarrow$ Continue train h_0 on $\mathcal{L}$
end for
return h_T

certainty and continue sampling from the same document until uncertainty falls below a threshold. Then, they sample the most uncertain span from a new document. We modify their method because the uncertainty threshold will vary for different datasets and models. Instead, we use the number of documents read to control context switching.

Labeling An oracle (e.g., human annotator or gold data) labels the antecedents of sampled spans with discrete annotations (Section 6.2).

Continued Training We combine data labeled from current and past cycles. We train the source model h_0 (which is already trained on ONTONOTES) on the labeled target data. We do not continue training a model from a past active learning cycle because it may be biased from only training on scarce target data (Ash et al., 2020).

6.3 Active Learning for CR through Simulations and Humans

We run experiments to understand two important factors of active learning for CR: sources of model uncertainty (Section 6.2.1) and balancing reading against labeling (Section 6.2.2). First, we simulate active learning on PRECO to compare sampling strategies based on various forms of uncertainty (Section 6.3.1). Then, we set up a user study to investigate how humans perform when labeling spans from fewer or more documents from PRECO (Section 6.3.2). Specifically, we analyze their annotation time and throughput. Finally, we run large-scale simulations on PRECO and QBCOREF (Section 6.3.3). We explore different combinations of sampling strategies and labeling configurations.

Models In all experiments, the source model is the best checkpoint of ICOREF model trained on ONTONOTES (Xia et al., 2020) with SPANBERT-LARGE-CASED (Joshi et al., 2020) encoder. For continued training on the target dataset, we optimize with a fixed parameter configuration. The SPANBERT-LARGE-CASED encoder has 334M parameters and ICOREF has 373M parameters in total. For model fine-tuning, we train for a maximum of fifty epochs and implement early stopping with a patience of ten epochs. We set top span pruning to 0.4, dropout to 0.4, gradient clipping to 10.0, and learning rate to 1e-4 for Adam optimizer. The hyperparameter configuration is based on results from prior work (Lee et al., 2017; Xia et al., 2020). All experiments in the paper are ran on NVIDIA Tesla V100 GPU and 2.2 GHz Intel Xeon Silver 4114 CPU processor.

We evaluate models on AVG F_1, the averaged F_1 scores of MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and $CEAF_{\phi 4}$ (Luo, 2005). For all synthetic experiments, we simulate active learning with gold data substituting as an annotator. However, gold mention

boundaries are not used when sampling data. The model scores spans that are likely to be entity mentions for inference, so we limit the active learning candidates to this pool of high-scoring spans. For each active learning simulation, we repeat five runs with different random seed initializations.

Baselines We compare the proposed sampling strategies (Section 6.2.1) along with **li-clust-ent**, which is clustered entropy from Li et al. (2020) (Equation 6.8). Active learning is frustratingly less effective than random sampling in many settings (Lowell et al., 2019), so we include two random baselines in our simulation. **Random** samples from all spans in the documents. **Random-ment**, as well as other strategies, samples only from the pool of likely (high-scoring) spans. Thus, **random-ment** should be a stronger baseline than **random**.

Datasets ONTONOTES 5.0 is the most common dataset for training and evaluating CR (Pradhan et al., 2013). The dataset contains news articles and telephone conversations. Only non-singletons are annotated. Our experiments transfer a model trained on ONTONOTES to two target datasets: PRECO and QBCOREF. PRECO is a large corpus of grade-school reading comprehension texts (Chen et al., 2018a). Unlike ONTONOTES, PRECO has annotated singletons. There are 37K training, 500 validation, and 500 test documents. Because the training set is so large, Chen et al. (2018a) only analyze subsets of 2.5K documents. Likewise, we reduce the training set to a subset of 2.5K documents, comparable to the size of ONTONOTES.

The QBCOREF dataset (Guha et al., 2015) contains trivia questions from Quizbowl tournaments that are densely packed with entities from academic topics. Like PRECO, singletons are annotated. Unlike other datasets, the syntax is idiosyncratic and world knowledge is needed to

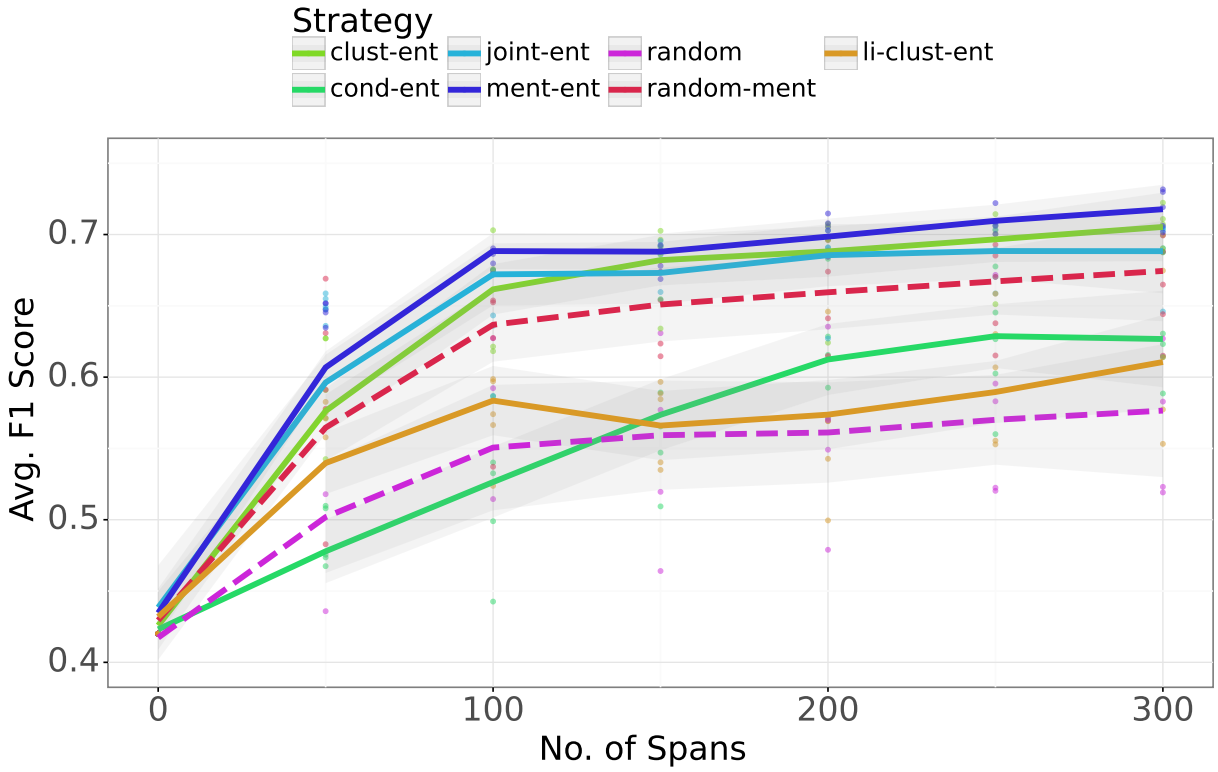


Figure 6.2: Test AVG F_1 on PRECO for each strategy. On each cycle, fifty spans from one document are sampled and labeled. We repeat each simulation five times. **Ment-ent**, **clust-ent**, and **joint-ent** are most effective while **random** hurts the model the most.

solve coreference. Examples are pronouns before the first mention of named entities and oblique references like "this polity" for "the Hanseatic League". These complicated structures rarely occur in everyday text but serve as challenging examples for CR. There are 240 training, 80 validation, and 80 test documents.

6.3.1 Simulation: Uncertainty Sampling

To compare different sampling strategies, we first run experiments on PRECO. We sample fifty spans from one document for each cycle. By the end of a simulation run, 300 spans are sampled from six documents. For this configuration, uncertainty sampling strategies generally

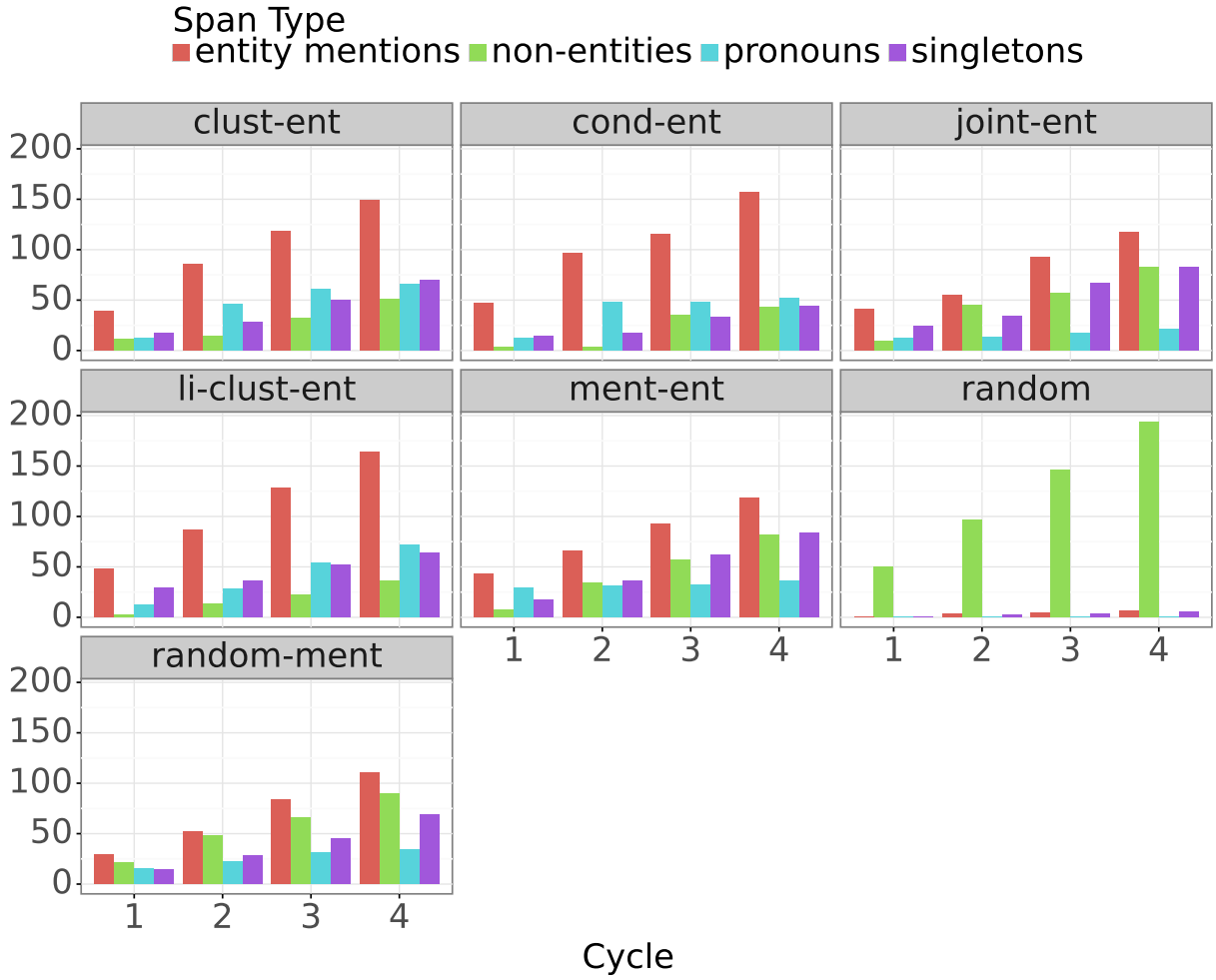


Figure 6.3: Cumulative counts of entities, non-entities, pronouns, and singletons sampled for each strategy over first four cycles of the PRECO simulation. **Random** mostly samples non-entities. **Li-clust-ent** and **cond-ent** sample many entity mentions but avoid singletons.

reach higher accuracy than the random baselines (Figure 6.2), but **cond-ent** and **li-clust-ent** are worse than **random-ment**.

Distribution of Sampled Span Types To understand the type of spans being sampled, we count entity mentions, non-entities, pronouns, and singletons that are sampled by each strategy (Figure 6.3). **Random** samples very few entities, while other strategies sample more entity mentions. **Clust-ent** and **cond-ent** sample more entity mentions and pronouns because the sampling ob-

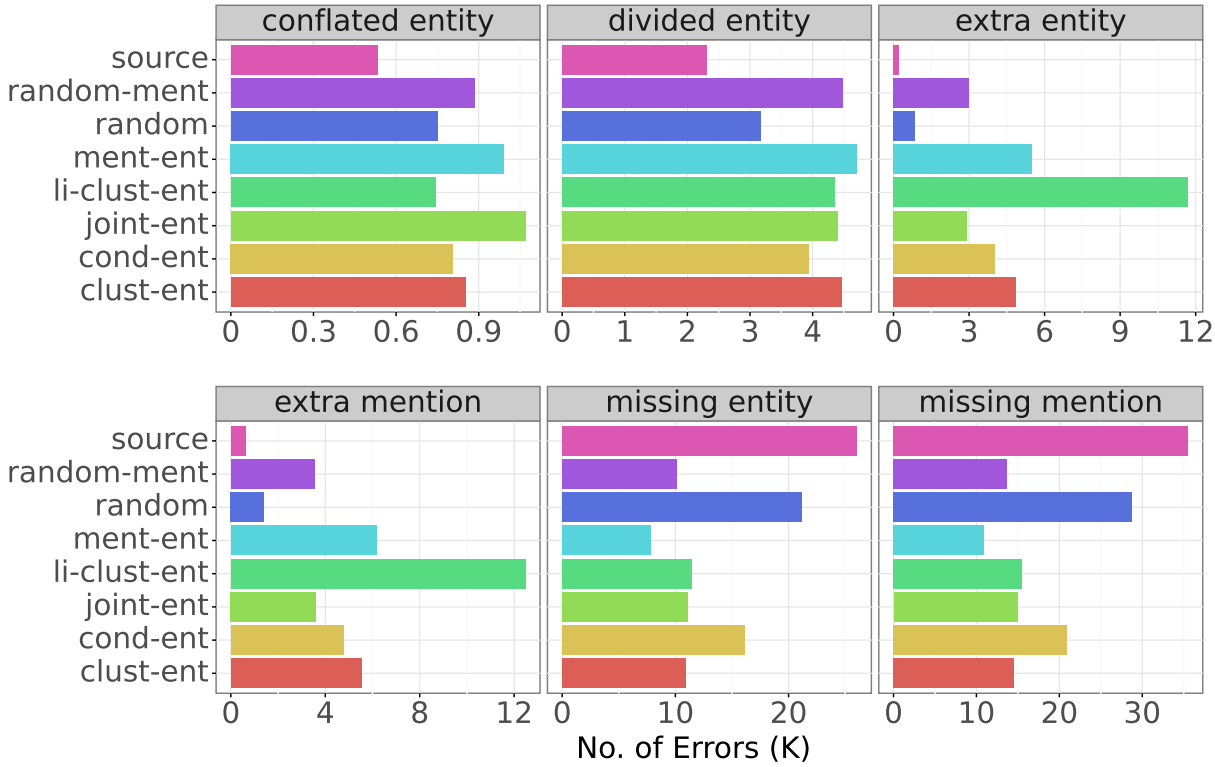


Figure 6.4: For each sampling strategy, we analyze the model from the last cycle of its PRECO simulation. We compare the number of errors across common error types in CR. The **source** ONTONOTES model severely suffers from *missing entities* and *missing mentions*. **Ment-ent** helps most with reducing these errors.

jective prioritizes mentions that are difficult to link. **Clust-ent**, **joint-ent**, and **ment-ent** sample more singleton mentions. These strategies also show higher AVG F_1 (Figure 6.2). For transferring from ONTONOTES to PRECO, annotating singletons is useful because only non-singleton mentions are labeled in ONTONOTES. We notice **ment-ent** sampling pronouns, which should obviously be entity mentions, only in the first cycle. Many pronouns in ONTONOTES are singletons, so the mention detector has trouble distinguishing them initially in PRECO.

Error Analysis Kummerfeld and Klein (2013) enumerate the ways CR models can go wrong: *missing entity*, *extra entity*, *missing mention*, *extra mention*, *divided entity*, and *conflated entity*. *Missing entity* means a gold entity cluster is missing. *Missing mention* means a mention span for

a gold entity cluster is missing. The same definitions apply for *extra entity* and *extra mention*. *Divided entity* occurs when the model splits a gold entity cluster into multiple ones. *Conflated entity* happens when the model merges gold entity clusters. For each strategy, we analyze the errors of its final model from the simulation's last cycle (Figure 6.4). We compare against the **source** model that is only trained on ONTONOTES.

The **source** model makes many *missing entity* and *missing mention* errors. It does not detect several entity spans in PRECO, like locations ("Long Island") or ones spanning multiple words ("his kind acts of providing everything that I needed"). These spans are detected by uncertainty sampling strategies and **rand-ment**. **Ment-ent** is most effective at reducing "missing" errors. It detects gold entity clusters like "constant communication" and "the best educated guess about the storm". By training on spans that confuse the mention detector, the model adapts to the new domain by understanding what constitutes as an entity mention.

Surprisingly, **li-clust-ent** makes at least twice as many *extra entity* and *extra mention* errors than any other strategy. For the sentence, "Living in a large building with only 10 bedrooms", the gold data identifies two entities: "a large building with only 10 bedrooms" and "10 bedrooms". In both ONTONOTES and PRECO, the guidelines only allow the longest noun phrase to be annotated. Yet, the **li-clust-ent** model predicts additional mentions, "a large building" and "only 10 bedrooms". We find that **li-clust-ent** tends to sample nested spans (Table 6.4). Due to the summed entropy computation, nested spans share similar values for clustered entropy as they share similar antecedent-linking probabilities. This causes the *extra entity* and *extra mention* errors because the model predicts there are additional entity mentions within a mention span.

Finally, we see a stark difference between **random-ment** and **random**. Out of all the sampling strategies, **random** is least effective at preventing *missing entity* and *missing mention*

errors. We are more likely to sample non-entities if we randomly sample from all spans in the document (Section 6.4.5). By limiting the sampling pool to only spans that are likely to be entity mentions, we sample more spans that are useful to label for CR. Thus, the mention detector from neural models should be deployed during active learning.

6.3.2 User Study: Reading and Labeling

We hold a user study to observe the trade-off between reading and labeling. Three annotators, with minimal NLP knowledge, label spans sampled from PRECO. We use **ment-ent** to sample spans because the strategy shows highest AVG F_1 (Figure 6.2). First, the users read instructions (Section 6.4.4) and practice labeling for ten minutes. Then, they complete two sessions: **FewDocs** and **ManyDocs**. In each session, they label as much as possible for at least twenty-five minutes. In **FewDocs**, they read fewer documents and label roughly seven spans per document. In **ManyDocs**, they read more documents and label about one span per document.

For labeling coreference, we develop a user interface that is open-sourced (Figure 6.8). To label the antecedent of the highlighted span, the user clicks on a contiguous span of tokens. The interface suggests overlapping candidates based on the spans that are retained by the CR model.

In the user study, participants label at least twice as much in **FewDocs** compared to **ManyDocs** (Figure 6.5). By labeling more spans in **FewDocs**, the mean AVG F_1 score is also slightly higher. Our findings show that the number of read documents should be constrained to increase labeling throughput. Difference in number of labeled spans between **FewDocs** and **ManyDocs** is more pronounced when two annotators volunteer to continue labeling after required duration (Section 6.4.4).

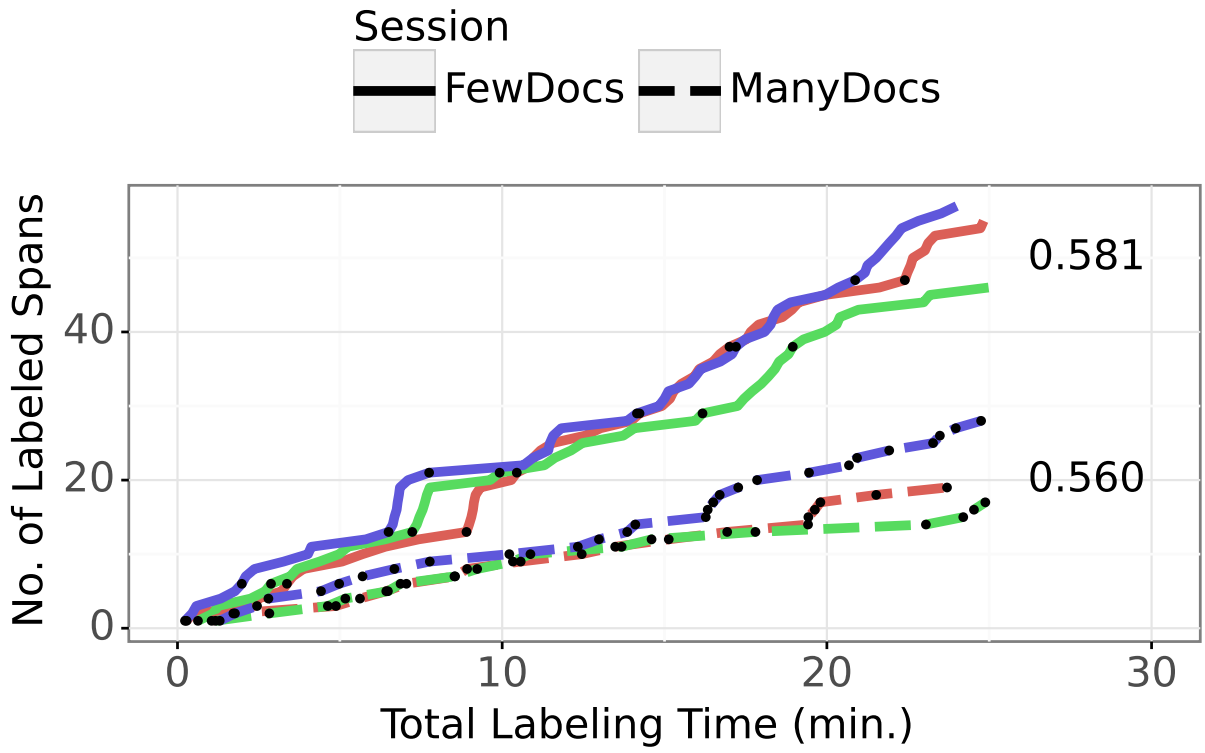
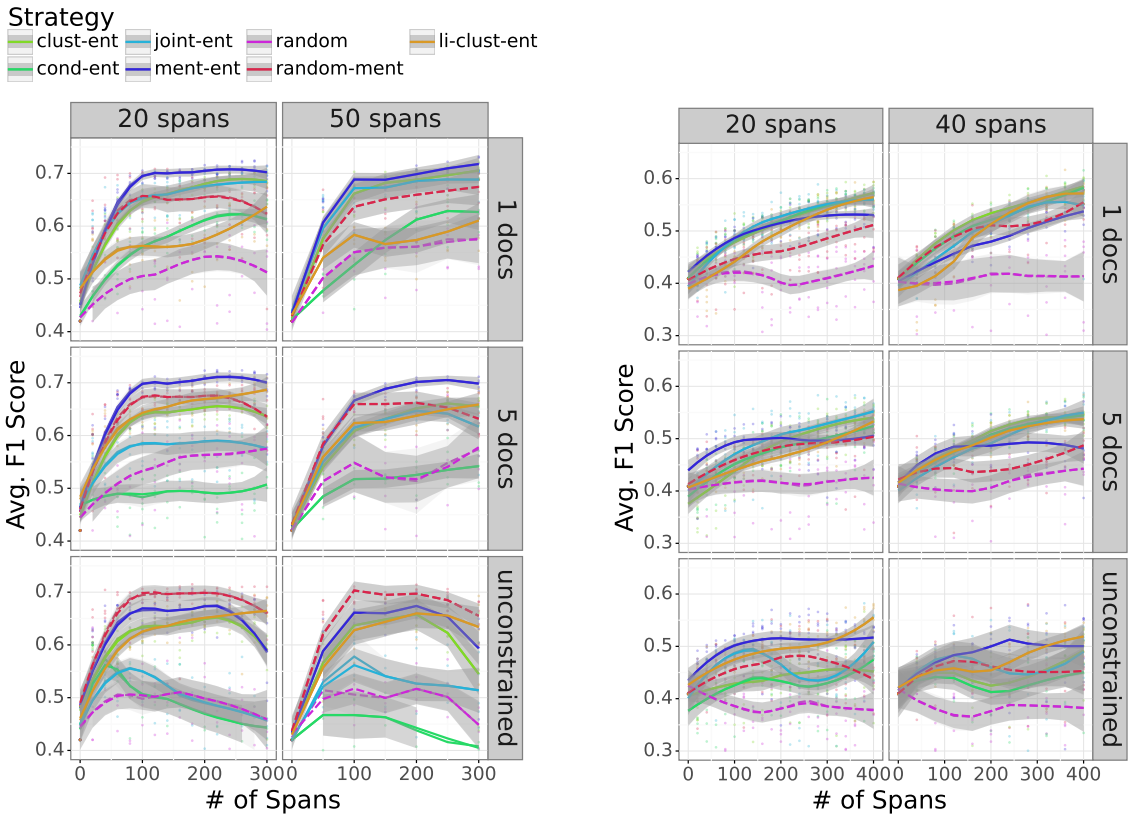


Figure 6.5: The number of spans labeled within twenty-five minutes. Each color indicates one of three users and the linetype designates the session. Black dots mark the first span labeled in a different document. The mean AVG F_1 across users for each session is on the right. By restricting the number of read documents in **FewDocs**, users label at least twice as many spans and the model slightly improves in AVG F_1.

6.3.3 Simulation: Uncertainty Sampling and Reading-Labeling Trade-off

We finally run simulations to explore *both* sources of model uncertainty and the trade-off between reading and labeling. The earlier experiments have individually looked at each aspect. Now, we analyze the interaction between both factors to understand which combination works best for adapting CR to new domains. We run simulations on PRECO and QBCOREF that trade-off the number of documents read m with the number of annotated spans k (Figure 6.6). We vary m between one, five, and an unconstrained number of documents. For PRECO, we set k to twenty and fifty. For QBCOREF, we set k to twenty and forty. These results are also presented in



(a) PRECO (b) QBCOREF

Figure 6.6: Test AVG F_1 on PRECO and QBCOREF of each strategy throughout simulations. Each row varies in m, the maximum number of documents read per cycle. Each column varies in k, the number of annotated spans per cycle. For m of one or five, **ment-ent** shows highest AVG F_1 for PRECO and other uncertainty sampling strategies are best for QBCOREF. When m is unconstrained, many strategies show unstable training.

numerical form (Section 6.4.3).

PRECO For PRECO, the test AVG F_1 of ICOREF trained on the full training dataset is 0.860. When m is constrained to one or five, AVG F_1 can reach around 0.707 from training the model on only 300 spans sampled by **ment-ent**. As m increases, fewer spans are sampled per document and all sampling strategies deteriorate. After training on sparsely annotated documents, the model tends to predict singletons rather than cluster coreferent spans. Like in the user study, we see benefits when labeling more spans within a document. Interestingly, **li-clust-ent** performs better

when document reading is not constrained to one document. The issue with **li-clust-ent** is that it samples nested mention spans (Section 6.3.1). Duplicate sampling is less severe if spans can be sampled across more documents. Another strategy that suffers from duplicate sampling is **cond-ent** because it mainly samples pronouns. For some documents, the pronouns all link to the same entity cluster. As a result, the model trains on a less diverse set of entity mentions and **cond-ent** drops in AVG F_1 as the simulation continues.

QBCOREF For QBCOREF, the test AVG F_1 of ICOREF trained on the full training dataset is 0.795. When we constrain m to one or five, **li-clust-ent**, **clust-ent**, **cond-ent**, and **joint-ent** have high AVG F_1. Clustering entity mentions in QBCOREF questions is difficult, so these strategies help target ambiguous mentions (Table 6.5). **Ment-ent** is less useful because demonstratives are abundant in QBCOREF and make mention detection easier. **Li-clust-ent** still samples nested entity mentions, but annotations for these spans help clarify interwoven entities in Quizbowl questions. Unlike PRECO, **li-clust-ent** does not sample duplicate entities because nested entity mentions belong to different clusters and need to be distinguished.

Overall, the most helpful strategy depends on the domain. For domains like PRECO that contain long documents with many singletons, **ment-ent** is useful. For domains like QBCOREF where resolving coreference is difficult, we need to target linking uncertainty. Regardless of the dataset, **random** performs worst. **Random-ment** has much higher AVG F_1, which shows the importance of the mention detector in active learning. Future work should determine the appropriate strategy for a given domain and annotation setup.

Strategy	PRECO	QBCOREF
random	2	< 1
random-ment	4	< 1
ment-ent	5	< 1
li-clust-ent	12	< 1
clust-ent	12	1
cond-ent	14	1
joint-ent	16	1

Table 6.1: The time (minutes) to sample a batch of fifty spans from five documents from either PRECO or QBCOREF for a given active learning strategy. On large datasets like PRECO, we see that **li-clust-ent**, **clust-ent**, **cond-ent**, and **joint-ent** are slower because the strategy needs to incrementally cluster each span and then compute clustering entropy.

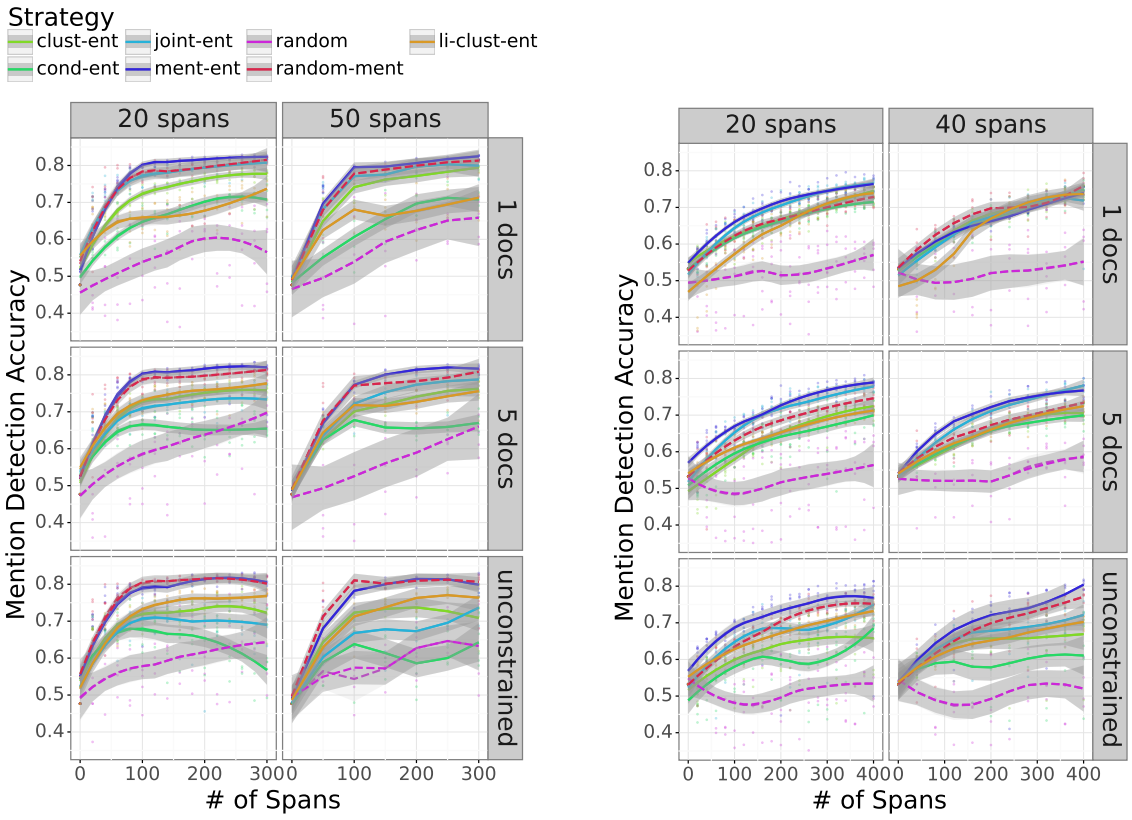
6.4 Additional Experimental Details

We provide additional details about experiments from Section 6.3. These details include simulation time, mention detection accuracy, numerical results from simulations, extended results from the user study, and examples of spans sampled from simulations.

6.4.1 Simulation Time

We compare the time to sample fifty spans between different active learning strategies for PRECO and QBCOREF (Table 6.1). For PRECO, **clust-ent**, **cond-ent**, and **joint-ent** are slower because they need to run documents through ICOREF and get span-cluster likelihood. On the other hand, **ment-ent** only needs unary scores s_m, which is much faster to compute. Thus, for both datasets, running **ment-ent** takes about the same time as **random-ment**.

For QBCOREF, fine-tuning ICOREF on fifty spans takes three minutes and fine-tuning on full training set takes thirty-four minutes. For PRECO, fine-tuning ICOREF on fifty spans takes nine minutes and fine-tuning on full training set takes five hours and 22 minutes.



(a) PRECO (b) QBCOREF

Figure 6.7: Comparing mention detection accuracy on test set for different active learning strategies across reading/labeling configurations. The plots are formatted in the same way as Figure 6.6. Generally, mention detection improves most from **ment-ent** sampling.

6.4.2 Mention Detection Accuracy

For the annotation simulation in Section 6.3, we also record mention detection accuracy. As **ment-ent** targets ambiguity in mention detection, it is the most effective strategy for improving mention detection (Figure 6.7). The strategy is unaffected by labeling setup parameters, like the number of spans labeled per cycle or the number of documents read per cycle. For strategies like **cond-ent** and **joint-ent**, mention detection accuracy is stagnant or decreases as more spans are sampled (Figure 6.7a). Due to deteriorating mention detection, the AVG F_1 of models also drop.

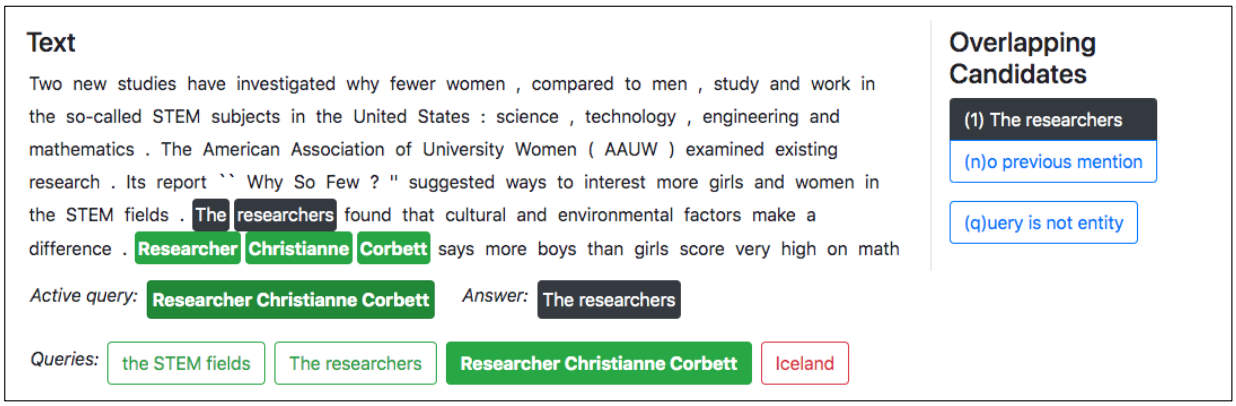


Figure 6.8: On the user interface, the sampled span is highlighted and the user must select an antecedent. If no antecedents exist or the span is not an entity mention, then the user will click the corresponding buttons.

6.4.3 Numerical Results

The results for AVG F_1 and mention detection accuracy are presented as graphs throughout this chapter. To concretely understand the differences between the methods, we provide results in numerical form (Tables 6.2,6.3). We show results from the PRECO and QBCOREF simulations where twenty spans are labeled each cycle and the number of documents read is either one or an unconstrained amount. The values in the tables show the mean and variance of AVG F_1 and mention detection accuracy over five different runs.

6.4.4 User Study

Instructions to Participants We give the following instructions to user study participants:

> You will be shown several sentences from a document. We have highlighted a mention (a word or phrase) of an entity (a person, place, or thing). This entity mention may be a pronoun (such as “she” or “their”) or something else.
>
> We need your help to find an earlier mention of the same entity, whether in the same

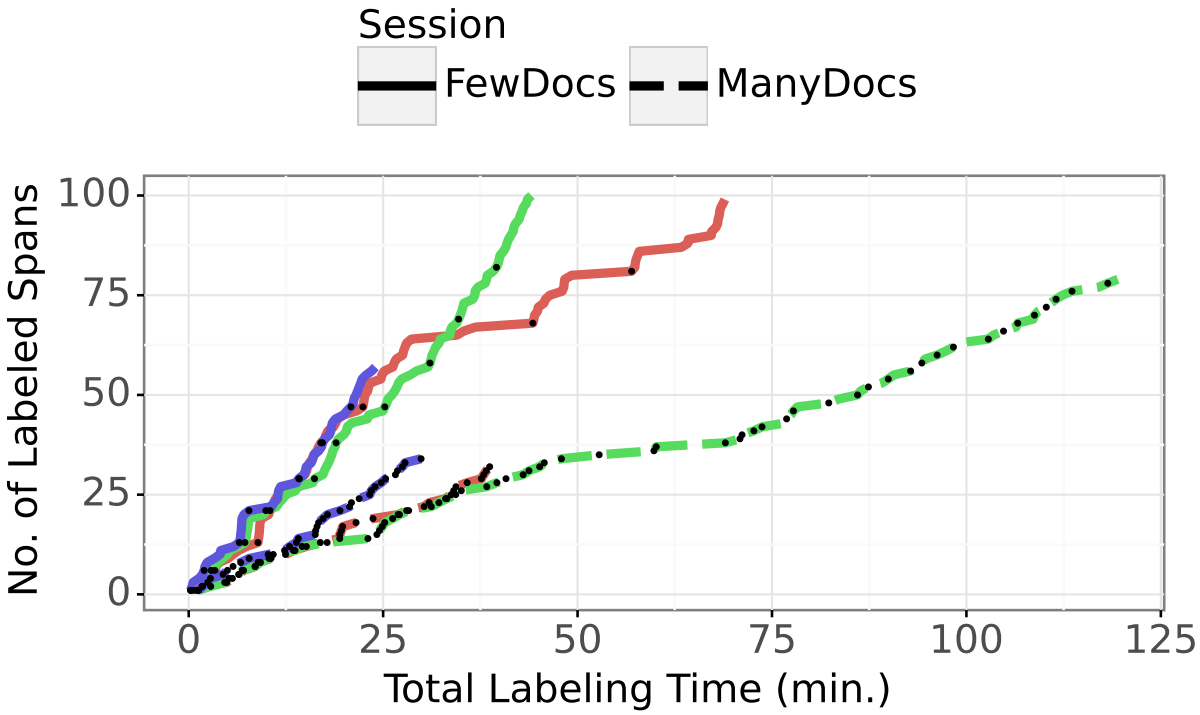


Figure 6.9: Full annotation times of participants (distinguished by color) during the user study. Over a longer period of time, the difference in number of labeled spans between the two sessions is much more pronounced. Within fourty-five minutes, the red user can label a hundred spans in the **FewDocs** session but only labels about thirty spans in the **ManyDocs** session.

sentence or in an earlier sentence. The mention does not have to be the immediately previous one.

If the span is not an entity mention or does not have an antecedent, please make note of it on the interface.

User Interface We design a user interface for annotators to label coreference (Figure 6.8). The user interface takes the sampled spans from active learning as input. Afterward, it will present the document and highlight the sampled spans in the document. The user the proceeds to go through the list of "Queries". For the "Active query", they need to either: find its antecedent, mark there is "no previous mention", or indicate that "query is not an entity". The interface will

suggest some overlapping candidates to help narrow down the user's search. The candidates are spans that the CR model scores as likely entity mentions. Users may use keyboard shortcuts to minimize labeling time. The code for the user interface is released along with the code for the simulations.

Extending Annotation Time User study participants are asked to annotate at least twenty-five minutes (Section 6.3.2). During the study, two participants continue to label after the minimum duration. Figure 6.9 shows full results from the user study. Over a longer duration, the differences between the **FewDocs** and **ManyDocs** sessions are clearer.

6.4.5 Examples of Sampled Spans

We provide examples of spans that are sampled from the experiments. For these examples, we look at the simulation where document reading is constrained to one document and twenty spans are sampled per cycle. We compare the spans sampled by each strategy for both PRECO (Table 6.4) and QBCOREF (Table 6.5). Across domains, the strategies behave similarly, but we notice some differences in **ment-ent** and **joint-ent**. In PRECO, those strategies tend to sample a mix of spans that are and are not entity mentions (Section 6.3.1). In QBCOREF, they sample more entity mentions. This could be due to more entity mentions present in a Quizbowl question, which makes it more likely to sample something that should belong to an entity cluster.

For other strategies, we notice some issues. As mentioned in Section 6.3.1, **li-clust-ent** tends to sample nested entity mentions, which may become redundant for annotators to label. In fact, AVG F_1 for **li-clust-ent** tends to be lower if document reading is constrained to one document. **Cond-ent** suffers from redundant labeling because pronouns are repeatedly sampled

and they tend to link to the same entity cluster.

6.5 Conclusion

In this chapter, we use interaction to adapt domains for a more complicated task, coreference resolution. Neural CR models desperately depend on large, labeled data. We use active learning to transfer a model trained on ONTONOTES, the "de facto" dataset, to new domains. Active learning for CR is difficult because the problem does not only concern sampling examples. We must consider different aspects, like sources of model uncertainty and cost of reading documents. In both simulations and the user study, CR improves from continued training on spans sampled from the same document rather than different contexts.

The next chapter will summarize the findings about interactive transfer learning. From inductive to transductive transfer learning, we have proposed various approaches to transfer different models. We will discuss future directions in areas such as user-centric active learning and proactive learning.

Total No. of Labeled Spans	m	Strategy	AVG F_1	Mention Accuracy
100	1	clust-ent	0.64 ± 0.02	0.71 ± 0.03
		cond-ent	0.57 ± 0.02	0.66 ± 0.02
		joint-ent	0.64 ± 0.03	0.76 ± 0.02
		ment-ent	$\mathbf{0.70 \pm 0.01}$	$\mathbf{0.80 \pm 0.00}$
		random	0.43 ± 0.09	0.49 ± 0.11
		random-ment	0.65 ± 0.04	0.78 ± 0.02
		li-clust-ent	0.56 ± 0.02	0.65 ± 0.03
	unconstrained	clust-ent	0.62 ± 0.03	0.70 ± 0.03
		cond-ent	0.43 ± 0.09	0.67 ± 0.04
		joint-ent	0.55 ± 0.06	0.71 ± 0.05
		ment-ent	0.65 ± 0.03	0.76 ± 0.03
		random	0.48 ± 0.07	0.54 ± 0.07
		random-ment	$\mathbf{0.69 \pm 0.01}$	$\mathbf{0.80 \pm 0.01}$
		li-clust-ent	0.62 ± 0.01	0.73 ± 0.01
200	1	clust-ent	0.68 ± 0.01	0.77 ± 0.01
		cond-ent	0.62 ± 0.02	0.70 ± 0.03
		joint-ent	0.68 ± 0.03	0.80 ± 0.02
		ment-ent	$\mathbf{0.71 \pm 0.01}$	$\mathbf{0.82 \pm 0.00}$
		random	0.48 ± 0.18	0.55 ± 0.21
		random-ment	0.65 ± 0.05	0.77 ± 0.07
		li-clust-ent	0.57 ± 0.05	0.67 ± 0.04
	unconstrained	clust-ent	0.65 ± 0.02	0.73 ± 0.03
		cond-ent	0.36 ± 0.08	0.63 ± 0.07
		joint-ent	0.40 ± 0.12	0.67 ± 0.12
		ment-ent	0.67 ± 0.03	$\mathbf{0.81 \pm 0.01}$
		random	0.49 ± 0.08	0.61 ± 0.07
		random-ment	$\mathbf{0.69 \pm 0.01}$	$\mathbf{0.81 \pm 0.00}$
		li-clust-ent	0.65 ± 0.03	0.75 ± 0.03
300	1	clust-ent	0.68 ± 0.02	0.78 ± 0.01
		cond-ent	0.61 ± 0.03	0.70 ± 0.04
		joint-ent	$\mathbf{0.69 \pm 0.02}$	0.81 ± 0.01
		ment-ent	$\mathbf{0.69 \pm 0.02}$	$\mathbf{0.82 \pm 0.00}$
		random	0.50 ± 0.09	0.58 ± 0.10
		random-ment	0.61 ± 0.10	0.81 ± 0.01
		li-clust-ent	0.63 ± 0.05	0.73 ± 0.05
	unconstrained	clust-ent	0.51 ± 0.12	0.70 ± 0.04
		cond-ent	0.33 ± 0.07	0.57 ± 0.04
		joint-ent	0.41 ± 0.05	0.69 ± 0.04
		ment-ent	0.54 ± 0.07	0.80 ± 0.02
		random	0.40 ± 0.04	0.60 ± 0.13
		random-ment	0.65 ± 0.05	$\mathbf{0.80 \pm 0.04}$
		li-clust-ent	$\mathbf{0.67 \pm 0.02}$	0.78 ± 0.01

Table 6.2: Results of PRECO simulation in numerical form, accompanying the graphs in Figures 6.6a and 6.7a. The table shows AVG F_1 and mention detection accuracy of experiments where twenty spans are sampled and labeled each cycle. Results are shown for m, the maximum number of documents read, equal to one and also unconstrained.

Total No. of Labeled Spans	m	Strategy	AVG F_1	Mention Accuracy
100	1	clust-ent	0.47 ± 0.06	0.62 ± 0.06
		cond-ent	0.47 ± 0.03	0.61 ± 0.03
		joint-ent	$\mathbf{0.50 \pm 0.03}$	$\mathbf{0.65 \pm 0.02}$
		ment-ent	$\mathbf{0.50 \pm 0.01}$	0.66 ± 0.03
		random	0.40 ± 0.07	0.53 ± 0.07
		random-ment	0.44 ± 0.06	0.63 ± 0.04
		li-clust-ent	0.45 ± 0.02	0.59 ± 0.03
	unconstrained	clust-ent	0.41 ± 0.05	0.59 ± 0.07
		cond-ent	0.39 ± 0.10	0.57 ± 0.05
		joint-ent	0.50 ± 0.01	0.66 ± 0.02
		ment-ent	$\mathbf{0.51 \pm 0.02}$	$\mathbf{0.69 \pm 0.01}$
		random	0.36 ± 0.08	0.48 ± 0.10
		random-ment	0.48 ± 0.02	0.65 ± 0.01
		li-clust-ent	0.47 ± 0.01	0.62 ± 0.02
200	1	clust-ent	0.52 ± 0.01	0.67 ± 0.01
		cond-ent	0.52 ± 0.02	0.66 ± 0.02
		joint-ent	$\mathbf{0.53 \pm 0.03}$	0.70 ± 0.03
		ment-ent	0.51 ± 0.02	$\mathbf{0.71 \pm 0.02}$
		random	0.40 ± 0.06	0.53 ± 0.08
		random-ment	0.48 ± 0.05	0.68 ± 0.01
		li-clust-ent	0.49 ± 0.01	0.64 ± 0.02
	unconstrained	clust-ent	0.45 ± 0.04	0.64 ± 0.06
		cond-ent	0.39 ± 0.06	0.55 ± 0.06
		joint-ent	0.48 ± 0.05	$\mathbf{0.70 \pm 0.03}$
		ment-ent	0.49 ± 0.08	0.68 ± 0.13
		random	0.34 ± 0.08	0.50 ± 0.11
		random-ment	0.49 ± 0.04	$\mathbf{0.70 \pm 0.01}$
		li-clust-ent	$\mathbf{0.50 \pm 0.03}$	0.68 ± 0.02
300	1	clust-ent	0.54 ± 0.02	0.70 ± 0.02
		cond-ent	$\mathbf{0.55 \pm 0.02}$	0.70 ± 0.02
		joint-ent	$\mathbf{0.55 \pm 0.02}$	0.74 ± 0.01
		ment-ent	0.53 ± 0.02	$\mathbf{0.75 \pm 0.02}$
		random	0.42 ± 0.05	0.55 ± 0.06
		random-ment	0.49 ± 0.03	0.69 ± 0.03
		li-clust-ent	0.53 ± 0.04	0.71 ± 0.02
	unconstrained	clust-ent	0.46 ± 0.04	0.67 ± 0.06
		cond-ent	0.42 ± 0.07	0.58 ± 0.12
		joint-ent	0.43 ± 0.11	0.68 ± 0.08
		ment-ent	0.50 ± 0.06	0.74 ± 0.04
		random	0.34 ± 0.18	0.45 ± 0.23
		random-ment	0.47 ± 0.08	$\mathbf{0.75 \pm 0.02}$
		li-clust-ent	$\mathbf{0.52 \pm 0.03}$	0.71 ± 0.01

Table 6.3: Results of QBCOREF simulation in numerical form, accompanying the graphs in Figures 6.6b and 6.7b. The table shows AVG F_1 and mention detection accuracy of experiments where twenty spans are sampled and labeled each cycle. Results are shown for m, the maximum number of documents read, equal to one and also unconstrained.

Strategy	Sampled Spans	Comments
random	Later, I got out of the back door secretly and gave the food to the old man, whose [name I had discovered]$_1$ was Taff. I had never seen anything else as lovely as the smile of satisfaction [on]$_2$ Taff's face when he ate the food. From then on, my visits to [the old house had]$_3$ a purpose, and I enjoyed every minute of the rest of my stay.	*Sampled spans are typically not entity mentions.*
random-ment	When opening the door, his face was full of smiles and he hugged [his two children and gave [his wife]$_2$ a kiss]$_1$. Afterwards, he walked with me to the car. We passed the tree. I was so curious that I asked [him]$_3$ about what I had [seen]$_4$ earlier.	*Diverse set of span types is sampled, including spans that are not entity mentions and ones that do link to entities.*
li-clust-ent	Although [he and [his young men]$_2$]$_1$ had taken no part in the killings, he knew that [the white men]$_3$ would blame [all of [the Indians]$_5$]$_4$.	*Many sampled spans are nested entity mentions.*
ment-ent	This summer, Republicans have been [meeting]$_1$ "behind closed doors" on a Medicare proposal scheduled to be released [later this month, only a few weeks before Congress votes]$_2$ on it, thereby avoiding independent analysis of the costs, mobilization by opponents and other inconvenient aspects of a long national debate. Two years ago, the Republicans rang alarms about the [Clinton]$_3$ plan's emphasis on [managed care]$_4$	*Sampled spans are both entity mentions and non-entities. The spans are difficult for mention detection like "meeting" but may also be hard for clustering like "Clinton".*
clust-ent	After that, [Mary]$_1$ buys some school things, too. Here [mother]$_2$ buys a lot of food, like bread, cakes, meat and fish. [They]$_3$ get home very late.	*Different types of entity mentions are sampled.*
cond-ent	It is a chance to thank everyone who has contributed to shaping [you]$_1$ during the high school years; it is a chance to appreciate all those who have been instrumental in [your]$_2$ education. Take a moment to express gratitude to all those who have shared the experiences of [your]$_3$ high school years.	*More pronouns are sampled because they are obviously entity mentions and hard to cluster. However, repeated sampling of the same entity occurs.*
joint-ent	[This]$_1$ is an eternal regret handed down from generation to generation and [you]$_2$ are only one of those who languish for (...) followers. [Love]$_3$ is telephone, but it is difficult to seize [the center time for dialing]$_4$, and you will let the opportunity slip if your call is either too early or too late.	*Many entity mentions are sampled but some are difficult for mention detector to detect.*

Table 6.4: The example spans from PRECO documents that are sampled with each active learning strategy.

Strategy	Sampled Spans	Comments
random	The discovery of a tube behind a [fuse box alarms Linda, and the image of stock[ings]$_2$ disturbs the main]$_2$ character due to his guilt over [an encounter with a woman and his son Biff in [Boston]$_4$]$_3$.	*Choice of sampled spans are very random and do not seem to improve learning coreference.*
random-ment	The speaker of one of [this author's works]$_1$ invites the reader to [take]$_2$ a little sun, a little honey, as commanded by [Persephone's]$_3$ bees.	*Diverse set of span types is sampled, including spans that are not entity mentions and ones that do link to entities.*
li-clust-ent	For 10 points, name [this [Moliere]$_2$ play about [Argan who is constantly concerned with [his]$_4$ health]$_3$]$_1$.	*Many sampled spans are nested entity mentions.*
ment-ent	He then sees [Ignorance and Want]$_1$ emerge from [a cloak]$_2$. Earlier, he sees [a door-knocker]$_3$ [transform]$_4$ into [a human figure, which drags a belt made of chains and locks]$_5$.	*Compared to* PRECO*, more entity mentions are sampled but most sampled spans are still difficult to detect.*
clust-ent	[[Its]$_2$ protagonist]$_1$ hires Croton to rescue a different character after listening to a giant - LRB - * - RRB - Christian named Urban [discuss]$_3$ a meeting at Ostranium.	*Compared to* PRECO*, a few sampled spans are not entity mentions.*
cond-ent	While [this work]$_1$ acknowledges the soundness of the arguments that use the example of the ancients, [[its]$_3$ author]$_2$ refuses to reply to [them]$_4$, adding that we are constructing no system here [we]$_5$ are a historian, not a critic.	*More pronouns are sampled because they are obviously entity mentions and hard to cluster. Unlike* PRECO*, repeated sampling occurs less often.*
joint-ent	This man falls in love with [the maid with [lime colored panties]$_2$]$_1$ and dates [Luciana]$_3$.	*Compared to* PRECO*, more entity mentions are sampled.*

Table 6.5: The example spans from QBCOREF documents that are sampled with each active learning strategy.

Chapter 7: Conclusion

Transfer learning in NLP is crucial because practical applications require models that can accommodate the shifts in language. Recent breakthroughs, like the transformer models, have developed intricate models and clever optimization techniques for successful transfer. The thesis takes a step further by introducing a human into the loop. Humans have shown to robustly and quickly generalize cognitive skills. Therefore, we should carefully design AI systems that can update their internal representations or decisions based on human feedback.

In Section 1.2, we list three objectives of the dissertation. First, we want to explore the **scope of applications** of interactive transfer learning. We focus on two categories of transfer learning: inductive (transfer across tasks) and transductive (transfer across domains). Second, we aim to improve knowledge transfer for **low-resource settings**. If data, time, and computing resources are scarce, we can no longer rely on costly, large models. In emergent situations, it becomes crucial to quickly update the model through human input. Third, we seek to investigate how user interaction varies for different levels of **model complexity**. More complex models achieve higher accuracy, but feedback from users may still help transfer knowledge.

We address these three objectives through a summary of the main contributions (Section 7.1). In the summary, we compare and contrast the findings from each chapter. Finally, we conclude with a discussion of future research directions (Section 7.2).

7.1 Summary of Contributions

Scope of Applications The problems in transfer learning are either inductive or transductive, where the former involves shift in task and the latter is defined by shift in domain. The main contributions are thus organized by the types of transfer learning. The chapters in Part I discuss interactive methods for inductive transfer learning where information from one task determines the feedback needed for another task. The chapters in Part II transition to human-in-the-loop work on transductive transfer learning. Here, domain expertise from users is crucial for overcoming missing knowledge gaps between domains or languages.

In Part I, we observe that the general trend for inductive transfer learning involves using information from one task to design the user feedback loop for another task. In Chapter 3, the user refines a cross-lingual topic model to classify text across languages. In Chapter 4, the surprisal values from language modeling indicates the type of text that surprises BERT. The active learning strategy ALPS then requests labels from users for these examples. The hypothesis here is that we can improve classification accuracy of language models from fine-tuning on labels of surprising documents.

In Part II, interactive transductive learning typically requires users to provide some sort of alignment across domains. The common pattern is to have users align words that are critical for learning the underlying task. In Chapter 5, the user modifies the nearest neighbors of words in order to align cross-lingual word embeddings. For our CLIME system, the words of interest are ones that are most relevant to the classification task. Finally, Chapter 6 involves users labeling coreference of words in text from the target domain. This type of labeling serves as an alignment because users may indicate words that are no longer entity mentions or group them with a new

entity cluster.

For inductive transfer learning, the interaction is less straightforward and requires additional assumptions about the relationship between tasks. The novelty lies in how information from the source task can be used for interaction in the target task. For transductive transfer learning, the overall feedback framework is more obvious: have users align words across domains. However, this user feedback loop depends on two conditions. First, our algorithm must find which words need alignment from users. Second, users should be equipped with the necessary domain knowledge to provide the necessary alignment (e.g. fluent in multiple languages). For both inductive and transductive transfer learning, user interaction needs to be carefully designed to be helpful for adapting models.

Low-resource Settings Labeled data is a bottleneck in machine learning. Theoretically, supervised learning is only guaranteed to succeed if there is an abundance of labeled data (Section 2.1). Active learning is an extensively studied approach to resolve some issues with data annotation by sampling a subset of data to label (Section 2.3.1). Thus, we build upon prior work to develop active learning approaches for transfer learning. In Chapter 4, our active learning strategy ALPS relies entirely on pre-training for data sampling. This bypasses popular strategies, like uncertainty sampling, that require a model already trained on labeled data. As a result, we can quickly adapt BERT to a classification task with a smaller number of annotated examples. ALPS and prior active learning works focus on text classification. In Chapter 6, we investigate active learning for coreference resolution. The active learning setup is more complicated because CR is a clustering task and involves labeling spans rather than documents. In both user studies and experiments, sampling spans from a constrained set of documents is more efficient for adapting the model to a

new domain.

In some situations, there might not be enough unlabeled data to begin with. This is true of datasets for low-resource languages. Chapter 5 is primarily focused with adapting models trained on a high-resource language to a low-resource one. If there is not enough data, aligning cross-lingual word embeddings becomes an issue. Thus, the CLIME interface has users mark similarity and dissimilarity between nearest neighbors as a way to reshape the embedding space. We show significant increase in accuracy from user interaction for four low-resource languages. Chapter 3 also discusses application to low-resource languages. Through MTAnchor, users can easily create relevant topics to align the high-resource and low-resource languages.

Beyond data scarcity, we might encounter scenarios where computing power or time is limited. This could happen during times of natural disasters or political upheaval. Chapters 3 and 5 address these issues. The model in Chapter 3 is relatively small, but the advantage is that user updates are fast. In mere seconds, the anchoring algorithm changes model parameters according to user feedback. This type of tool would be useful in emerging situations. The CLIME interface is also quick as it does not need to retrain the model on labeled data. It involves a post-processing step that simply shifts the annotated word embeddings around.

Model Complexity For Parts I and II, the chapters in each part are ordered by increasing model complexity. With larger models, we can solve more complicated problems. For different levels of model complexity, interaction can be helpful. In low-resource settings, smaller models are more practical and certainly need input from humans. For problems that require more elaborate language understanding, larger models are more appropriate but may rely on user feedback for cheap updates.

Part I begins with topic models in Chapter 3 and Part II starts with word embeddings in Chapter 5. In both chapters, the models have relatively low complexity and updates are quick. Users can update MTAnchor in an online fashion and immediately see a new topic model within seconds (Chapter 3). For CLIME, the updates are offline but fast because the system only needs to shift a batch of word embeddings (Chapter 5).

Parts I and II end with chapters that involve large language models. In Chapter 4, we develop ALPS, an active learning strategy for BERT. In Chapter 6, we use active learning to help adapt SPANBERT. In these chapters, the models have relatively high complexity and have been to learn general information about language. Since the internal representations are high-dimensional and dense, it makes it harder to modify internal representations from a small amount of user feedback. At the same time, it is unnecessary to update the internal representations because the model has already undergone extensive pre-training. Therefore, these chapters focus on active learning to sample data for labeling. While the user interaction for these complex models is simple, the methods are effective for adapting them to new domains or tasks.

7.2 Future Directions

The discussion in Chapter 7.1 summarizes our contributions with the dissertation objectives in mind. In this section, we discuss some limitations of the thesis and suggest potential avenues for future work.

User-Centric Active Learning The active learning strategies in Chapters 4 and 6 have focused on the model and the data. In general, active learning strategies have either been model-centric or data-centric. Uncertainty sampling is model-centric because it finds examples that confuse

the model the most. Diversity sampling is data-centric as it searches for a representative set of examples. However, these methods neglect user preferences when sampling data. The users might prefer or be more proficient at labeling certain documents. If we find those documents for users to label, then users may annotate data more efficiently.

User-centric active learning can help avoid some pitfalls of uncertainty sampling. When dealing with noisy data, uncertainty sampling may select garbled passages that are difficult for users to read and interpret. An example of this would be,

> "#&!?the univrty of MD %CALL 1-800!!!! on thursday,,, two students encounter a rare sighting of (((CALL NOW FOR THE BEST DEALS))) a comet shower…".

There possibly might exist another passage that is much less noisy yet still confuses the model,

> "Scientists have discovered new life forms on Jupiter with resilient anatomy and can speak the Selkup language…".

We prefer the cleaner passage, especially if users have to perform span-level annotations within the text.

Lee et al. (2020) introduce a user-centric sampling strategy for active learning. Their strategy measures the difficulty of an instance for a user to annotate and chooses examples that are not too challenging or easy for the user. In some cases, the user-centric objective may not align with model-centric uncertainty sampling. They also combine the model-centric and user-centric strategies into one joint optimization strategy. This joint strategy leads to higher accuracy in the experiments. However, the paper only simulates an artificial user for one particular language learning task. Ideally, research in user-centric active learning should experiment with real human users and generalize across different NLP tasks.

First, we can consider a simple, user-centric strategy based on readability. If the passage is less noisy and shorter, then reading and labeling it would require less effort. So, we would sample easy passages for the users to label. Through simulations and user studies, we can compare this baseline to uncertainty sampling. Later on, we can develop more complex user-centric strategies and also a hybrid user-model strategy. Our experiments can first focus on text classification but then move on to harder tasks, like machine translation. Thus, we can fully understand the **scope of applications** for user-centric active learning.

Proactive Learning One crucial flaw of active learning is that it assumes the annotators are perfect. Realistically, the users are humans who are subject to making mistakes. In Chapter 3.4, the Mechanical Turk workers have to refine English-Chinese topic models. Their interactive update improves topic coherence and classification accuracy for English more drastically than for Chinese. We assume there are many more English speakers on Mechanical Turk (Pavlick et al., 2014), which might be why the Chinese scores are lower. In this scenario, we would ideally want bilingual English-Chinese speakers to interact with the topic model.

Proactive learning seeks to generalize active learning by removing these four assumptions about the user: "infallible (never wrong), indefatigable (always answers), individual (only one oracle), and insensitive to costs (always free or always charges the same)" (Donmez and Carbonell, 2008). The initial approach to proactive learning is to view it as a utility optimization problem with budget constraints. The goal is to jointly find examples to label and the users who should label those examples. They simulate the user with logistic regression classifiers trained on different data subsets.

Despite the strong motivations for proactive learning, there are only a few published works

on this topic. Nghiem et al. (2021) set up proactive learning strategies in an annotation toolkit. Their strategy is straightforward: send difficult examples to the expert annotator and easy examples to the novice user. Future research should develop proactive learning strategies for more complicated situations, especially ones that involve transfer learning. If we need expertise in a specific domain or task, we have to find annotators with that set of knowledge. That way, we can recruit users who will annotate data with higher precision and speed. For example, we recruit user study participants through Upwork who are fluent in the target, low-resource language in Chapter 5. To accomplish this, we look through worker's profiles to determine their language skills. Other researchers vet annotators before letting them participate in the user study (Briakou and Carpuat, 2020).

Seeking users to label data or interact with the AI system is non-trivial. We need people who are equipped with the knowledge needed to adapt the machine learning models. The problem becomes even more challenging when we have to consider budget constraints. If the most expert annotator is too expensive, we may need to find another knowledgeable annotator who can complete the task for a cheaper cost. A proactive learning strategy should resolve this issue by matching unlabeled data subsets to the appropriate users.

To develop proactive learning strategies, Quizbowl would be an interesting domain to focus on (Rodriguez et al., 2019). Wallace et al. (2019) have humans write adversarial Quizbowl questions to evaluate the robustness of QA systems. Quizbowl questions belong to different trivia categories. We may also want users to label entity links or coreference relationships in Quizbowl questions. Regardless of the type of interaction, the user should have prior knowledge about the particular question. Given a set of questions, the proactive learning strategy can recruit people who specialize in a particular category. Or, the strategy may be more fine-grained and recruit ex-

perts on a specific topic. With Quizbowl as a testbed, we can investigate how proactive learning can improve the adaptation of QA models to new domains.

The research on proactive learning is critical for **low-resource settings**. By finding the appropriate annotators, we can obtain high-quality labeled data for reduced costs. Proactive learning also can help recruit people with the necessary background for esoteric domains like low-resource languages.

Online, Interactive Updates Active learning is primarily used in an offline setting. First, users annotate data that is used to train the model. Then, the model is deployed online. This cycle of labeling, retraining, and deploying is repeated. However, a more attractive solution is to adapt the AI in an online fashion. The user interacts with the system and the model quickly learns from the feedback. In the online scenario, the user can rapidly provide feedback without having to wait for the model to retrain.

The interactive topic model in Chapter 3 follows this online setting. When the users refine the MTAnchor topic model through modifying anchor words for each topic, the algorithm takes merely seconds to update the topic-word distributions. While MTAnchor is fast, it may not be applicable to many NLP tasks. The coreference resolution models in Chapter 6 are much more complex, so it would be interesting to develop an online, interactive approach to adapt them. Could users label antecedents of a few spans of text and see immediate changes in the coreference resolution model? For online algorithms, **model complexity** is an important factor to consider.

A possible solution is to have users interact with a simple model (e.g. logistic regression) where online updates are fast and effortless. Then, the user may continuously interact with the model and provide as much feedback as they want. After a session of interactions, we accumulate

the user feedback to update the larger model in an offline approach like retraining. With this setup of two models, the user can rapidly interact with the smaller model while we spend more time updating the larger model for improved accuracy.

One concern with this approach is that the user feedback may only apply to the smaller model and not appropriately update the larger model. Lowell et al. (2019) show negative results when training a model on a dataset that is acquired through active learning with another model. However, their experiments are limited to active learning for text classification and named entity recognition. We may observe more significant gains for more complicated tasks like question answering. So, this proposed approach might be useful for interactions beyond active learning and tasks other than simple classification. The research for online user-AI interaction is worthwhile to pursue yet remains unexplored.

Bibliography

Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. 2005. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Dana Angluin. 1988. Queries and concept learning. *Machine Learning*, 2(4):319–342.

Andrew Arnold, Ramesh Nallapati, and William W Cohen. 2007. A comparative study of methods for transductive transfer learning. In *Seventh IEEE international conference on data mining workshops (ICDMW 2007)*. IEEE.

Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the International Conference of Machine Learning*.

Sanjeev Arora, Rong Ge, and Ankur Moitra. 2012. Learning topic models–going beyond SVD. In *Foundations of Computer Science (FOCS)*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the Association for Computational Linguistics*.

David Arthur and Sergei Vassilvitskii. 2006. k-means++: The advantages of careful seeding. Technical report, Stanford.

Jordan T. Ash and Ryan P. Adams. 2019. On warm-starting neural network training. *arXiv preprint arXiv:1910.08475*.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *Proceedings of the International Conference on Learning Representations*.

Caglar Aytekin, Xingyang Ni, Francesco Cricri, and Emre Aksu. 2018. Clustering and unsupervised anomaly detection with L2 normalized deep auto-encoder representations. In *International Joint Conference on Neural Networks*.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*.

David Bamman, Olivia Lewke, and Anya Mansoor. 2020. An annotated dataset of coreference in English literature. In *Proceedings of the Language Resources and Evaluation Conference*.

Peter L Bartlett and Shahar Mendelson. 2002. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482.

Jonathan Baxter. 2000. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of Empirical Methods in Natural Language Processing*.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Proceedings of Advances in Neural Information Processing Systems*.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. When is "nearest neighbor" meaningful? In *International Conference on Database Theory*.

Or Biran and Courtenay Cotton. 2017. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the Association for Computational Linguistics*.

Zalán Bodó, Zsolt Minier, and Lehel Csató. 2011. Active learning with clustering. In *Active Learning and Experimental Design Workshop in Conjunction with AISTATS 2010*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jordan Boyd-Graber and David M. Blei. 2009. Multilingual topic models for unaligned text. In *Proceedings of Uncertainty in Artificial Intelligence*.

Eleftheria Briakou and Marine Carpuat. 2020. Detecting Fine-Grained Cross-Lingual Semantic Divergences without Supervision by Learning to Rank. In *Proceedings of Empirical Methods in Natural Language Processing*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. Desouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–480.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Bo Chen, Wai Lam, Ivor Tsang, and Tak-Lam Wong. 2009. Extracting discriminative concepts for domain adaptation in text mining. In *Knowledge Discovery and Data Mining*.

Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille, and Shu Rong. 2018a. PreCo: A large-scale dataset in preschool vocabulary for coreference resolution. In *Proceedings of Empirical Methods in Natural Language Processing*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018b. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Noam Chomsky. 1975. *Syntactic Structures*. De Gruyter Mouton.

Noam Chomsky. 1980. On cognitive structures and their development: A reply to Piaget. In *Language and Learning: The debate between Jean Piaget and Noam Chomsky*. Harvard University Press.

Jaegul Choo, Changhyun Lee, Chandan K Reddy, and Haesun Park. 2013. Utopian: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE transactions on visualization and computer graphics*, 19(12):1992–2001.

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All that's 'human' is not gold: Evaluating human evaluation of generated text. In *Proceedings of the Association for Computational Linguistics*.

Katherine M. Collins, Catherine Wong, Jiahai Feng, Megan Wei, and Joshua B. Tenenbaum. 2022. Structured, flexible, and robust: benchmarking and improving large language models towards more human-like behavior in out-of-distribution reasoning tasks. *arXiv preprint arXiv:2205.05718*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the Association for Computational Linguistics*.

Noah Constant, Christopher Davis, Christopher Potts, and Florian Schwarz. 2009. The pragmatics of expressive content: Evidence from large corpora. *Sprache und Datenverarbeitung*, 33(1–2):5–21.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *Association for the Advancement of Artificial Intelligence*.

Sanjoy Dasgupta. 2011. Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Franck Dernoncourt and Ji Young Lee. 2017. PubMed 200k RCT: a dataset for sequential sentence classification in medical abstracts. *International Joint Conference on Natural Language Processing*, 2:308–313.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Pinar Donmez and Jaime G Carbonell. 2008. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the ACM International Conference on Information and Knowledge Management*.

Long Duong, Hadi Afshar, Dominique Estival, Glen Pink, Philip R Cohen, and Mark Johnson. 2018. Active learning for deep semantic parsing. In *Proceedings of the Association for Computational Linguistics*.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig. 2020. *Ethnologue:Languages of the World*. SIL International.

Allyson Ettinger, Philip Resnik, and Marine Carpuat. 2016. Retrofitting sense-specific word vectors using parallel text. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

European Parliament and Council of the European Union. 2016. General data protection regulation.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of Empirical Methods in Natural Language Processing*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

Paul Felt, Eric Ringger, Kevin Seppi, Kevin Black, and Robbie Haertel. 2015. Early gains matter: A case for preferring generative over discriminative crowdsourcing models. In *Proceedings of the Association for Computational Linguistics*.

Shi Feng and Jordan Boyd-Graber. 2019. What can AI do for me: Evaluating machine learning interpretations in cooperative play. In *International Conference on Intelligent User Interfaces*.

Shi Feng, Eric Wallace, Alvin Grissom II, Pedro Rodriguez, Mohit Iyyer, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretation difficult. In *Proceedings of Empirical Methods in Natural Language Processing*.

Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. 2017. Domain adaptation for relation extraction with domain adversarial neural network. In *International Joint Conference on Natural Language Processing*.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *Proceedings of the International Conference of Machine Learning*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):1–35.

Yang Gao, Christian M Meyer, and Iryna Gurevych. 2018. APRIL: Interactively learning to summarise by combining active preference learning and reinforcement learning. In *Proceedings of Empirical Methods in Natural Language Processing*.

Goran Glavaš and Ivan Vulić. 2018. Explicit retrofitting of distributional word vectors. In *Proceedings of the Association for Computational Linguistics*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Thomas L. Griffiths, Frederick Callaway, Michael B. Chang, Erin Grant, Paul M. Krueger, and Falk Lieder. 2019. Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, 29:24–30.

Anupam Guha, Mohit Iyyer, Danny Bouman, and Jordan Boyd-Graber. 2015. Removing the training wheels: A coreference dataset that entertains humans and challenges computers. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. *Journal of Machine Learning Research*, 70:1321–1330.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the Association for Computational Linguistics*.

E. Dario Gutiérrez, Ekaterina Shutova, Patricia Lichtenstein, Gerard de Melo, and Luca Gilardi. 2016. Detecting cross-cultural differences using a multilingual topic model. *Transactions of the Association for Computational Linguistics*, 4:47–60.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the Association for Computational Linguistics*.

Wei-Ning Hsu and Hsuan-Tien Lin. 2015. Active learning by learning. In *Association for the Advancement of Artificial Intelligence*.

Rong Hu, Brian Mac Namee, and Sarah Jane Delany. 2010. Off to a good start: Using clustering to select the initial training set in active learning. In *Florida Artificial Intelligence Research Society Conference*.

Xia Hu, Lingyang Chu, Jian Pei, Weiqing Liu, and Jiang Bian. 2021. Model complexity of deep learning: A survey. *Knowledge and Information Systems*, 63(10):2585–2619.

Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2014a. Interactive topic modeling. *Machine Learning*, 95(3):423–469.

Yuening Hu, Ke Zhai, Vlad Eidelman, and Jordan Boyd-Graber. 2014b. Polylingual tree-based topic models for translation domain adaptation. In *Proceedings of the Association for Computational Linguistics*.

Jagadeesh Jagarlamudi and Hal Daumé. 2010. Extracting multilingual topics from unaligned comparable corpora. In *Proceedings of the European Conference on Information Retrieval*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of Empirical Methods in Natural Language Processing*.

Constantinos Karouzos, Georgios Paraskevopoulos, and Alexandros Potamianos. 2021. UDALM: Unsupervised domain adaptation through language modeling. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. 2004. Detecting change in data streams. In *International Conference on Very Large Databases*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of Empirical Methods in Natural Language Processing*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. 2019. BatchBALD: Efficient and diverse batch acquisition for deep Bayesian active learning. In *Proceedings of Advances in Neural Information Processing Systems*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of International Conference on Computational Linguistics*.

Jan-Christoph Klie, Richard Eckart de Castilho, and Iryna Gurevych. 2020. From zero to hero: Human-in-the-loop entity linking in low resource domains. In *Proceedings of the Association for Computational Linguistics*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems*.

Jonathan K. Kummerfeld and Dan Klein. 2013. Error-driven analysis of challenges in coreference resolution. In *Proceedings of Empirical Methods in Natural Language Processing*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. In *Proceedings of Advances in Neural Information Processing Systems*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

Ji-Ung Lee, Christian M. Meyer, and Iryna Gurevych. 2020. Empowering Active Learning to Jointly Optimize System and User Demands. In *Proceedings of the Association for Computational Linguistics*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of Empirical Methods in Natural Language Processing*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Moontae Lee and David Mimno. 2014. Low-dimensional embeddings for interpretable anchor-based topic inference. In *Proceedings of Empirical Methods in Natural Language Processing*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of Advances in Neural Information Processing Systems*.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

David D. Lewis. 1992. Feature selection and feature extraction for text categorization. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

Belinda Z Li, Gabriel Stanovsky, and Luke Zettlemoyer. 2020. Active learning for coreference resolution using discrete annotation. In *Proceedings of the Association for Computational Linguistics*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, Antonios Anastasopoulos, Patrick Littell, and Graham Neubig. 2019. Choosing transfer languages for cross-lingual learning. In *Proceedings of the Association for Computational Linguistics*.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning to actively learn neural machine translation. In *Conference on Computational Natural Language Learning*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations*.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. 2019. Practical obstacles to deploying active learning. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jeffrey Lund, Connor Cook, Kevin Seppi, and Jordan Boyd-Graber. 2017. Tandem anchoring: A multiword anchor approach for interactive topic modeling. In *Proceedings of the Association for Computational Linguistics*.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the Association for Computational Linguistics*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the Association for Computational Linguistics*.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Tony McEnery and Andrew Wilson. 2003. Corpus linguistics. *The Oxford handbook of computational linguistics*, pages 448–463.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems*.

Timothy Miller, Dmitriy Dligach, and Guergana Savova. 2012. Active learning for coreference resolution. In *BioNLP: Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*.

David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of Empirical Methods in Natural Language Processing*.

Tom Mitchell. 1997. *Machine Learning*. Mcgraw Hill.

Tom M Mitchell. 1980. The need for biases in learning generalizations. Technical report, Rutgers University.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *Proceedings of Empirical Methods in Natural Language Processing*.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association for Computational Linguistics*, 5:309–324.

Minh-Quoc Nghiem, Paul Baylis, and Sophia Ananiadou. 2021. Paladin: An annotation tool based on active and proactive learning. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*.

Thang Nguyen, Jordan Boyd-Graber, Jeffrey Lund, Kevin Seppi, and Eric Ringger. 2015. Is your anchor going up or down? fast and accurate supervised topic models. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. 2010. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev, and Chris Callison-Burch. 2014. The language demographics of Amazon Mechanical Turk. *Transactions of the Association for Computational Linguistics*, 2:79–92.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jonas Pfeiffer, Naman Goyal, Xi Victoria Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022. Lifting the curse of multilinguality by pre-training modular transformers. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Corbèn Poot and Andreas van Cranenburgh. 2020. A benchmark of rule-based and neural coreference resolution in Dutch novels and news. In *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*.

Forough Poursabzi-Sangdeh, Jordan Boyd-Graber, Leah Findlater, and Kevin Seppi. 2016. ALTO: Active learning with topic overviews for speeding label induction and document labeling. In *Proceedings of the Association for Computational Linguistics*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Conference on Computational Natural Language Learning*.

Geoffrey K Pullum and Barbara C Scholz. 2002. Empirical assessment of stimulus poverty arguments. *The linguistic review*, 19(1-2):9–50.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Piyush Rai, Avishek Saha, Hal Daumé III, and Suresh Venkatasubramanian. 2010. Domain adaptation meets active learning. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the Association for Computational Linguistics*.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Alison Renner. 2020. *Designing for the Human in the Loop: Transparency and Control in Interactive Machine Learning*. Ph.D. thesis, University of Maryland.

Pedro Rodriguez, Shi Feng, Mohit Iyyer, He He, and Jordan Boyd-Graber. 2019. Quizbowl: The case for incremental question answering. *arXiv preprint arXiv:1904.04792*.

Yuji Roh, Geon Heo, and Steven Euijong Whang. 2019. A survey on data collection for machine learning: A big data-AI integration perspective. *IEEE Transactions on Knowledge and Data Engineering*.

Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction. In *Proceedings of the International Conference of Machine Learning*.

Sebastian Ruder. 2019. *Neural transfer learning for natural language processing*. Ph.D. thesis, NUI Galway.

Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Mrinmaya Sachan, Eduard Hovy, and Eric P Xing. 2015. An active learning approach to coreference resolution. In *International Joint Conference on Artificial Intelligence*.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the Association for Computational Linguistics*.

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. In *Proceedings of Advances in Neural Information Processing Systems*.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *Proceedings of the International Conference on Learning Representations*.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Burr Settles. 2011. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of Empirical Methods in Natural Language Processing*.

Burr Settles, Mark Craven, and Lewis Friedland. 2008a. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*.

Burr Settles, Mark Craven, and Soumya Ray. 2008b. Multiple-instance active learning. In *Proceedings of Advances in Neural Information Processing Systems*.

Darsh Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. Adversarial domain adaptation for duplicate question detection. In *Proceedings of Empirical Methods in Natural Language Processing*.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal*, 27.

Kendrick Shen, Robbie Jones, Ananya Kumar, Sang Michael Xie, Jeff Z. HaoChen, Tengyu Ma, and Percy Liang. 2022. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation. *arXiv preprint arXiv:2204.00570*.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. 2018. Deep active learning for named entity recognition. In *Proceedings of the International Conference on Learning Representations*.

Bei Shi, Wai Lam, Lidong Bing, and Yinqing Xu. 2016. Detecting common discussion topics across culture from news reader comments. In *Proceedings of the Association for Computational Linguistics*.

Ben Shneiderman. 2022. *Human-Centered AI*. Oxford University Press.

Aditya Siddhant and Zachary C. Lipton. 2018. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of Empirical Methods in Natural Language Processing*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of Empirical Methods in Natural Language Processing*.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the Association for Computational Linguistics*.

Stephanie Strassel, Ann Bies, and Jennifer Tracey. 2017. Situational awareness for low resource languages: the LORELEI situation frame annotation task. In *Exploitation of Social Media for Emergency Relief and Preparedness*.

Stephanie Strassel and Jennifer Tracey. 2016. LORELEI language packs: Data, tools, and resources for technology development in low resource languages. In *Language Resources and Evaluation Conference*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the Association for Computational Linguistics*.

William R. Swartout. 1983. Xplain: A system for creating and explaining expert consulting programs. *Artificial Intelligence*, 21(3):285–325.

Niels A. Taatgen. 2013. The nature and transfer of cognitive skills. *Psychological review*, 120(3):439.

Nenad Tomasev, Milos Radovanovic, Dunja Mladenic, and Mirjana Ivanovic. 2013. The role of hubness in clustering high-dimensional data. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):739–751.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. In *Journal of Machine Learning Research*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the Association for Computational Linguistics*.

Alan M Turing and J Haugeland. 1950. Computing machinery and intelligence. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, pages 29–56.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland*.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. TREC-COVID: Constructing a pandemic information retrieval test collection. *arXiv preprint arXiv:2005.04474*.

Thuy Vu, Ming Liu, Dinh Phung, and Gholamreza Haffari. 2019. Learning how to active learn by dreaming. In *Proceedings of the Association for Computational Linguistics*.

Ivan Vulić and Anna-Leena Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of the Association for Computational Linguistics*.

Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, and Jordan Boyd-Graber. 2019. Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering. *Transactions of the Association for Computational Linguistics*, 7:387–401.

Chengyu Wang, Minghui Qiu, Jun Huang, and Xiaofeng He. 2020. Meta fine-tuning neural language models for multi-domain text mining. In *Proceedings of Empirical Methods in Natural Language Processing*.

Dan Wang and Yi Shang. 2014. A new active labeling method for deep learning. In *International Joint Conference on Neural Networks*.

Zijie J. Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. 2021. Putting humans in the natural language processing loop: A survey. In *Proceedings of the First Workshop on Bridging Human–Computer Interaction and Natural Language Processing*.

Gert W. Wolf. 2011. *Facility location: concepts, models, algorithms and case studies*. Taylor and Francis.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of Empirical Methods in Natural Language Processing*.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the Association for Computational Linguistics*.

Zhaofeng Wu and Matt Gardner. 2021. Understanding mention detector-linker interaction for neural coreference resolution.

Patrick Xia, João Sedoc, and Benjamin Van Durme. 2020. Incremental neural coreference resolution in constant memory. In *Proceedings of Empirical Methods in Natural Language Processing*.

Patrick Xia and Benjamin Van Durme. 2021. Moving on from OntoNotes: Coreference resolution model transfer. In *Proceedings of Empirical Methods in Natural Language Processing*.

Min Xiao and Yuhong Guo. 2013. A novel two-step method for cross language representation learning. In *Proceedings of Advances in Neural Information Processing Systems*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. 2003. Representative sampling for text classification using support vector machines. In *Proceedings of the European Conference on Information Retrieval*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Proceedings of Advances in Neural Information Processing Systems*.

Dave Yates and Scott Paquette. 2010. Emergency knowledge management and social media technologies: a case study of the 2010 haitian earthquake. In *Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem - Volume 47*, ASIS&T '10, pages 42:1–42:9, Silver Springs, MD, USA. American Society for Information Science.

Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020a. Cold-start active learning through self-supervised language modeling. In *Proceedings of Empirical Methods in Natural Language Processing*.

Michelle Yuan, Benjamin Van Durme, and Jordan Boyd-Graber. 2018. Multilingual anchoring: Interactive topic modeling and alignment across languages. In *Proceedings of Advances in Neural Information Processing Systems*.

Michelle Yuan, Patrick Xia, Chandler May, Benjamin Van Durme, and Jordan Boyd-Graber. 2022. Adapting coreference resolution models through active learning. In *Proceedings of the Association for Computational Linguistics*.

Michelle Yuan, Mozhi Zhang, Benjamin Van Durme, Leah Findlater, and Jordan Boyd-Graber. 2020b. Interactive refinement of cross-lingual word embeddings. In *Proceedings of Empirical Methods in Natural Language Processing*.

Mikhail Yurochkin, Aritra Guha, and XuanLong Nguyen. 2017. Conic scan-and-cover algorithms for nonparametric topic modeling. In *Proceedings of Advances in Neural Information Processing Systems*.

Li-Ming Zhan, Haowen Liang, Bo Liu, Lu Fan, Xiao-Ming Wu, and Albert Y.S. Lam. 2021. Out-of-scope intent detection with self-supervision and discriminative training. In *Proceedings of the Association for Computational Linguistics*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of Advances in Neural Information Processing Systems*.

Ye Zhang, Matthew Lease, and Byron C. Wallace. 2017. Active discriminative text representation learning. In *Association for the Advancement of Artificial Intelligence*.

Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of International Conference on Computational Linguistics*.