ABSTRACT

Title of proposal: Transfer Learning in Natural Language Processing through Interactive Feedback

Michelle Yuan, 2020

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 data domains and applications. For natural language processing (NLP), this is especially detrimental because language is perpetually changing. As people develop new ideas, written records reflect these innovations. Across the globe, there are thousands of distinct languages due to linguistic and cultural differences. Transfer learning transmits knowledge from source to target settings by modifying model architecture and optimization. This dissertation proposal takes a step further to include a "human in the loop". If language is a byproduct of human thought, then human feedback should help transfer knowledge for

NLP problems. Therefore, our goal is to improve model generalization under low-resource settings through interactive learning.

First, we develop an active learning strategy to annotate examples for text classifiers that have trained on little to no data. State-of-the-art language models learn general text representations from predicting token occurrence over large corpora. Thus, our strategy uses the language modeling loss to bootstrap classification uncertainty and sample representative points from surprisal clusters. Next, we refine cross-lingual word embeddings through user feedback for low-resource languages. Bilingual speakers transfer knowledge from English to the target language by aligning the cross-lingual embedding space. Finally, we create a multilingual, interactive topic modeling system for users to refine topics across languages. The user-constructed topic model bridges multilingual gaps in knowledge.

In the proposed work, we plan to explore interactive learning for NLP problems that require a comprehensive understanding of human language. For tasks like coreference resolution and question answering, users can link entities to help the model automate information extraction. Therefore, we will design algorithms and interfaces for users to efficiently transfer knowledge labeling text spans.

Transfer Learning in Natural Language Processing through Interactive Feedback

by

Michelle Yuan

Dissertation proposal 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
2020

Advisory Committee:
Professor Jordan Boyd-Graber, Chair
Professor John Dickerson, Departmental Representative
Professor Rachel Rudinger, Committee Member

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Language is dynamic—it varies 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). Daniels and Bright (1996) states that "Humankind is defined by languages; but civilization is defined by writing". Humans may be ethnically diverse, but almost everyone expresses themselves through writing. It is an universal form of communication and a hallmark of civilization (Harari, 2014). Over time, humans form new beliefs and invent new technology. These innovations cause different themes and topics to emerge in written records. Along with shift in semantics, the form of writing itself is constantly changing. From classical Chinese poetry in first millenium BCE (Yip, 1997) to eighteenth century literary novels (Watt, 1957) to modern-day tweets with hashtags, we see the transformation in written communication. Language is a medium that evolves with humans.

Natural Language Processing (NLP) uses computers to analyze human languages. Since the 1990s, the internet has accumulated vast amounts of text data (Castells, 1996). Thus, NLP models have relied on statistics and machine learning to analyze these online corpora. These include problems like text classification, parsing, and named entity recognition. To solve these tasks, NLP tries to model topics, part of speech, or other linguistic features. By training on extensively annotated datasets, these models can help automate natural language understanding.

A long-standing problem of machine learning is adapting models for unobserved situations. Typically, models are trained on a large, annotated dataset. During evaluation, they infer correct predictions for data from the training distribution. When the model is evaluated on out-of-domain data, results are disappointing. Humans are quick to process new information and link them to past experiences, but artificial intelligence (AI) cannot acquire new knowledge after training on only a few examples. Given that language is

always changing, this brittleness harms NLP. For example, a model trained on only English documents may fail to understand text in Shona, a written language of Zimbabwe. Or, the model may not detect a global pandemic as an important topic because it has not trained on news after 2019. How can NLP models adapt to dynamic shifts in language?

Transfer learning is an area of research that is concerned with adapting machine learning models to new problems (Pan and Yang, 2010). Modern NLP models strive to transfer knowledge across different data domains and downstream tasks. *Pre-training* is one successful way to accumulate general knowledge about language. The term entails preliminary training on massive corpora with a 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. Thus, most research in transfer learning focus on improving model architecture and optimization.

The training pipeline may be important for transfer learning, but there could be other factors to boost model performance. Since language reflects human thought, should humans not be included in model development? How can we incorporate feedback from humans to transfer knowledge? This proposal explores the possibilities and power of interactive NLP for transfer learning.

1.1 Motivation

Beginnings of transfer learning for NLP can be found in Bernhard Vauquois's pyramid (Vauquois, 1968), a framework for transfer-based machine translation. The goal is to transfer meaning from the source to the target language through intermiediate representations of text. Deerwester et al. (1990) proposes latent semantic analysis to create low-dimensional vector representations of documents based on word occurrences. The purpose of these vector representations, also known as *embeddings*, is to encode general, important knowledge about the data. Topic models take on a more statistical approach by representing documents as probabilistic mixtures of topics (Blei et al., 2003). Later on, neural networks are used to train word embeddings (Mikolov et al., 2013c). These word embeddings can then be used to form representations for documents. Bengio et al. (2013) emphasizes the importance of learning data representations that encode information to help transfer knowledge across tasks. Recently, transformer models (Devlin et al., 2019) learn

contextual word embeddings to capture semantics in different context.

The goal of transfer learning is to adapt models for shifts in the data domain or downstream task. Transfer learning problems are either transductive or inductive. *Transductive transfer learning* deals with transferring knowledge across different domains for the same task. Building a sentiment classifier for different languages (Chen et al., 2018b) is transductive transfer learning. *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). The goal of transfer learning is to improve the model generalization across different settings.

Prior work mainly make improvements upon model architecture and optimization to improve transfer learning. Results are impressive, but transfer learning is still impractical for deployment. In many settings, there are not enough computational resources or time to train enormous models. During the 2010 Haiti Earthquake, international relief workers heavily rely on social media to mitigate destruction (Yates and Paquette, 2010). Early in 2020, language scientists gather text data on the Coronavirus pandemic to help with clinical research (Voorhees et al., 2020). These emergencies show that rapid, low-cost transfer of knowledge is necessary to save human lives. Yet, current approaches are still too inefficient. The training algorithm is only a small part of a machine learning system Sculley et al. (2015). Therefore, the system needs a human in the loop who can guide the training process. Users can provide direct feedback on model transfer because they are aware of the problem that needs to be solved. To reach the desired results, human and AI must work together.

The rise of human-in-the-loop machine learning stems from the need for explainability (Swartout, 1983). 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). That is, the outputs of an AI algorithm need to be interpretable. By providing explanations of model behavior, users may improve the model because they know why certain predictions are being made (Ribeiro et al., 2016). Wallace et al. (2019) show that humans can create adversarial examples to strengthen question answering systems. All in all, interactive machine learning is designed to increase transparency and control for users (Renner, 2020).

Beyond transparency and control, human-AI interaction may help with

transfer learning. When humans discover a new problem, they can quickly apply past skills and experiences (Taatgen, 2013). Machine learning models fail to transfer knowledge as successfully as humans do (Griffiths et al., 2019). Therefore, user feedback may help transfer knowledge for NLP models and systems.

1.2 Objectives

The proposal aims to develop methods for interactive transfer learning and understand 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 are as follows:

1. **Scope of Applications:** As transfer learning entails shift in either domain or task, the interactive learning component differs in each situation. We want to explore human interaction for transductive and inductive transfer learning problems.
2. **Low-resource Settings:** Many languages are low-resource, which means that labeled data is scarce. For low-resource NLP, interactive transfer learning can be incredibly useful. Along with having insufficient training data, computational resources may also be limited. Thus, human annotations may reduce computational costs by providing the most useful information about the task.
3. **Natural Language Understanding:** Cognitive science shows that humans can transfer knowledge across various situations. For many tasks, the model may fail to understand a text passage because of misleading biases. Plus, there are certain problems that require domain-specific expertise, which may not be publicly available on the internet. User feedback can instill human intelligence and judgement to the model.

1.3 Organization

Chapter 2 provides background information on transfer learning and interactive learning. First, we discuss failures of supervised learning to motivate the need for transfer learning. Then, we describe various problems and settings in transfer learning. The two main branches are inductive transfer learning, which involves shift in task, and transductive transfer learning,

which centers on shift in the domain space. The chapter then introduces interactive learning by describing ways for machine learning models to learn from user feedback. We discuss how active learning algorithms seek examples for humans to label and we cover work in human-AI collaboration.

Chapter 3 presents application of active learning on inductive transfer learning. We describe our work on ALPS, which is an active learning algorithm that helps efficiently fine-tune pre-trained language models. By knowing the type of text that surpises the model, we can query labels for certain examples from the dataset.

Next, Chapter 4 looks into user interaction for cross-lingual learning, an important problem in transductive transfer learning. We investigate human-AI interaction for low-resource cross-lingual text classification. We propose CLIME, a human-in-the-loop framework that aligns word embeddings according to user feedback. Through the CLIME interface, the user can shift the cross-lingual embedding space to improve classification for neural networks. Additionally, CLIME can be combined with active learning to further improve accuracy. This shows that both instance-level and word-level annotations help transfer learning.

Afterward, Chapter 5 looks at both inductive and transductive learning. We adapt topic models for text classification across different languages through interaction. We create a system called MTAnchor for users to cross-lingually align topics. While topics help transfer knowledge across tasks and languages, human feedback additionally enhances domain knowledge and cross-lingual alignment.

Finally, Chapter 6 discusses future work for interactive learning on NLP tasks other than text classification. Problems, like coreference resolution and question answering, require mastery of natural language understanding. Thus, the annotation pipeline needs to ensure that user feedback can properly guide the model.

Chapter 2: Background

This chapter provides background information on topics relevant to the proposal. First, we introduce supervised learning, the traditional framework for training machine learning models (Section 2.1). We discuss the failures of supervised learning during sudden shifts in tasks and data domains. Next, we delve into transfer learning, which aims to overcome the 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 preview innovative human-in-the-loop interfaces for NLP tasks (Section 2.3.2).

2.1 Supervised Learning

Machine learning is broadly construed as a computer program using experience to improve upon performance measures 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 has achieved several breakthroughs. IBM's Watson defeats Jeopardy champions through a supervised question answering system. DeepMind's AlphaGo triumphs over world-champion Go players (Silver et al., 2016). 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 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, $\mathcal{Y}$ is a finite set of items. For regression problems, $\mathcal{Y} = \mathbb{R}$. 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. 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 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 training instances $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and their labels $y_1, \ldots, y_n$ are i.i.d, which means distributed independently and identically to the true distribution $p(\boldsymbol{x}, y)$. Under this assumption, we can train models through minimizing empirical risk $R(f)$ rather than having to compute expected risk $\widehat{R}(f)$.

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.

2.1.1 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 machine learning systems (Roh et al., 2019). For example, an annotator may take fifteen hours or more to label entities and relations for a hundred of news articles (Settles et al., 2008a). 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, is learning still possible?

In cognitive science, the "Poverty of the Stimulus" argument claims that people do not learn language 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. This concept is known as the "Poverty of the Stimulus". 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). 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 society wants to widely use AI systems, these models need to learn from sources beyond labeled data.

Transfer learning entails 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 distribution $p(\boldsymbol{Y} \mid \boldsymbol{X})$.

In transfer learning, we assume that the *source* model is reliable to use for 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 news categorizer for articles in English. Now, we want to train a classification model for news articles in Ilocano. If we do not have enough labeled data in Ilocano, how can we use the English classifier to categorize Ilocano articles?

Formally, suppose that the source model is trained on source domain $\mathcal{D}_S$ for source task $\mathcal{T}_S$. The model can reasonably compute $p_S(\boldsymbol{Y}_S \mid \boldsymbol{X}_S)$. Given target domain $\mathcal{D}_T$ and task $\mathcal{T}_T$, how can the model learn $p_T(\boldsymbol{Y}_T \mid \boldsymbol{X}_T)$ from $\mathcal{D}_S$ and $\mathcal{T}_S$ where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$?

Work in transfer learning can be categorized based on the shift in domain or task (Pan and Yang, 2010; Ruder, 2019). Broadly speaking, transfer learning is split into two main divisions:

1. *Inductive transfer learning* involves shift in target task $\mathcal{T}_T$ from source task $\mathcal{T}_S$, but labeled data is available in the target domain $\mathcal{D}_T$.
2. *Transductive transfer learning* occurs when tasks $\mathcal{T}_S$ and $\mathcal{T}_S$ are the same, but labeled data is only available in the source domain $\mathcal{D}_S$.

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.

Another reason for inductive transfer learning is a sudden change in model utility. When we initially train a model, we did not fully plan out the downstream task. After understanding the problem, we now realize that the trained model needs to be adapted for a specific purpose. For example, an intent detection model may only process customer complaints regarding bill issues. However, the company realizes that many customers are troubled by service issues. To avoid loss in profits, they must quickly change the model to accomodate these new requests.

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) (Section 2.2.1).

After pre-training a source model, we can adapt it to not just one, but several target tasks. For the source model to be applied in many target tasks, it must be pre-trained with general knowledge. Similar to representation learning (Bengio et al., 2013), the goal is to find *universal* representations for the data. For language, we should encode basic semantic and syntactic features about text. If we want to accumulate comprehensive amount of information, then pre-training requires extensive training time and resources. However, the large, pre-trained model can then be quickly adapted to a wide range of target tasks. In many models, the pre-training stage is long and costly, but further adaptation is efficient. 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 work 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.

Another seminal work 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.

While LSA is a dimensionality reduction technique, *topic modeling* uses a probabilistic framework to model text. 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). Inference for LDA is computed with either Gibbs sampling or variational Bayes. Wallach (2006) combine LDA with Dirchlet bigram modeling to improve topic modeling with n-gram statistics. 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. Using co-occurrence between anchor words and other words, the algorithm can determine the topic-word distribution. Then, we fix the topic-word distribution and use an inference algorithm (e.g. Variational

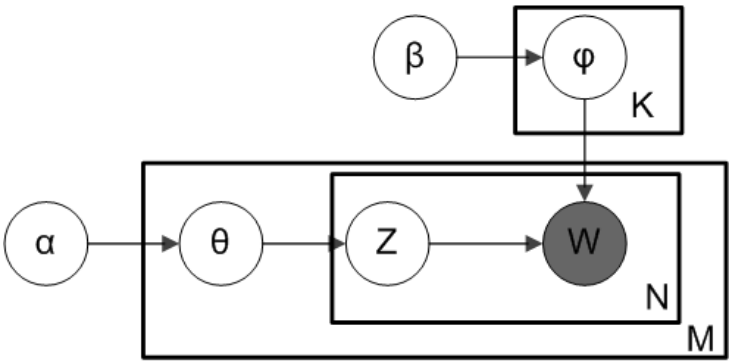


Figure 2.1: LDA topic model where word w is generated from a topic z with distribution ϕ_z, a Dirichlet prior with parameter β. Topic z is generated from θ, the document-topic Dirichlet distribution with parameter α. The generative model uses latent topics to transfer knowledge about language across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

Bayes) to learn topics for each document. Chapter 5 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 downstream task. This type of model adaptation is known as *feature extraction*. By computing word likelihood and co-occurrence, we can 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* maps word types to an embedding space. 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 func-

tion defined as

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

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

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

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. 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

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

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

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 approach.

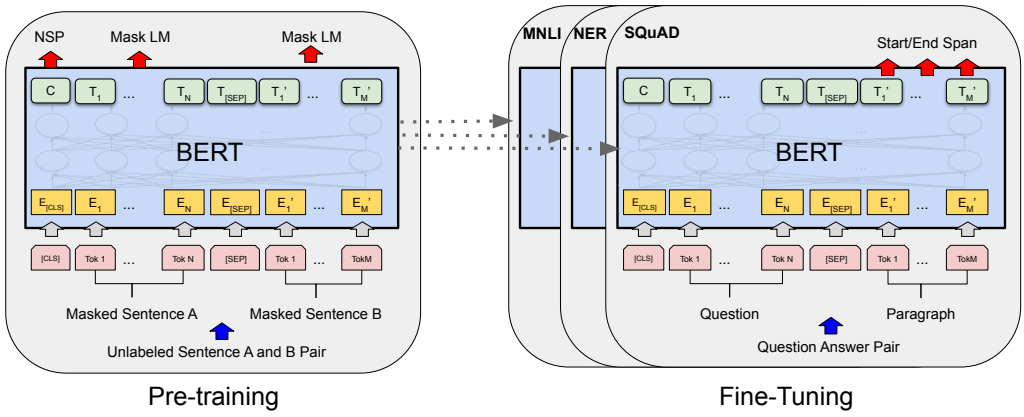


Figure 2.2: BERT (Devlin et al., 2019) is a bidirectional transformer model. The model is pre-trained with masked language modeling and next sentence prediction (left). After pre-training the model, it can be fine-tuned for many downstream tasks (right).

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 accurate representations for rare words in the vocabulary.

The neural architecture of theses 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, 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. 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.

Fine-tuning is 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.

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). 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.

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 $p(\mathbb{X})$. 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*. For instance, the distribution of words is different between news articles and medical abstracts.
2. The feature spaces of source and target domains differ, $\mathcal{X}_S \neq \mathcal{X}_T$. We refer to this as *cross-lingual learning* because feature spaces are different across various languages.

2.2.2.1 Domain Adaptation

Supervised models cannot handle shifts in data domain. The goal of domain adaptation is to accomodate 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.

Distribution similarity Many works in domain adaptation focus on minimizing 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. 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

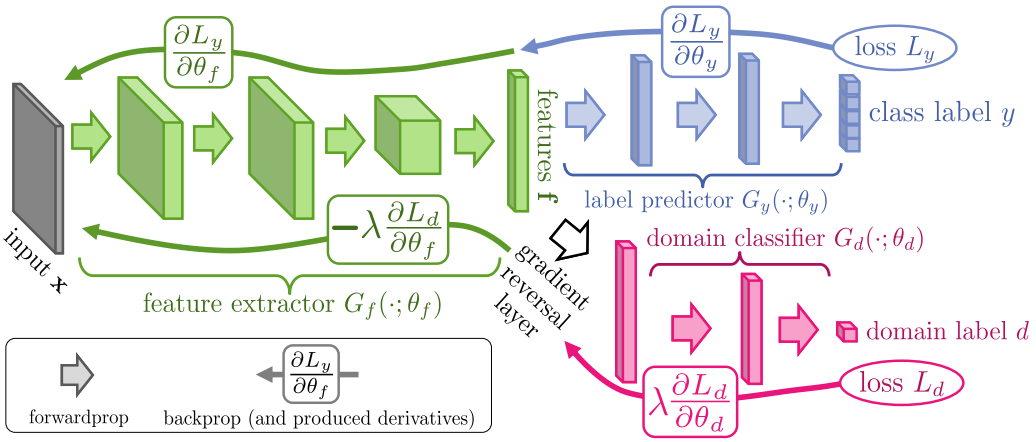


Figure 2.3: Domain-adversarial training (Ganin et al., 2016) first trains a feature extractor. Then, it passes the representation into the label classifier and domain classifier to jointly minimize classification loss and maximize domain loss.

$\mathcal{A}$ 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. 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.

2.2.2.2 Cross-lingual Learning

Given the linguistic diversity in the world, building AI systems for multiple languages is an important 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*, which is typically English, and the low-resource language is the *target language*. 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 tranformer 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. 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, word-level alignment is more commonly used. 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 dictionary entries and co-occurrence patterns. Following the success of neural word embeddings for English (Section 2.2.1), 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.

Vulić and Korhonen (2016) emphasize the influence of dictionary entries on the quality of cross-lingual word embeddings, so high-quality 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. Thus, retrofitting does not learn a translation matrix $\boldsymbol{W}$ and instead trains on individual constraints for each word. In 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 4 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 performs better than cross-lingual word embeddings for zero-shot cross-lingual transfer for tasks like natural language inference, document classification, and dependency parsing. Thus, efforts in multilingual NLP shift toward building cross-lingual trans-

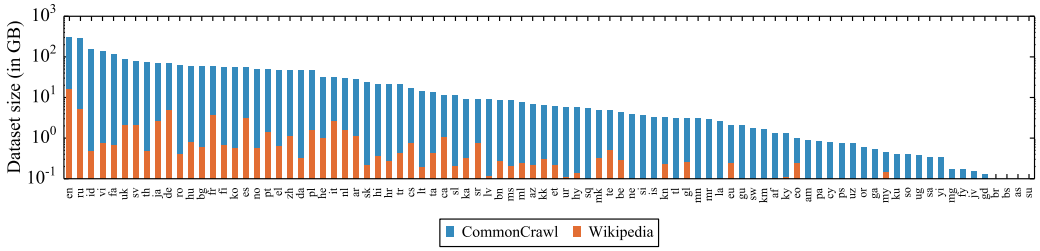


Figure 2.4: The size of pre-training data (GiB) across eighty-eight languages for multilingual transformers (Conneau et al., 2020). The mBERT and XLM models are pre-trained on the Wiki-100 corpus (orange). XLM-R trains on the CC-100 dataset (blue), which is much larger and covers more languages than Wiki-100.

former 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. While XLM-R shows state-of-the-art performance on low-resource languages, training the model is computationally expensive. Future work in cross-lingual learning should focus on harnessing the information from models like XLM-R and avoiding excessive computing costs.

2.3 Interactive Learning

Machine learning practicioners understand theory and implementation of AI systems. Others do not know as much and may not trust machines to make decisions. If we want people to use AI, then we need to make machine learning *interpretable*. As Biran and Cotton (2017) loosely define, "systems are interpretable if their operations can be understood by a human, either through introspection or through a produced explanation". The call for interpretability is also tied to FATE, which stands for fairness, accountability, transparency, and ethics.[1] If people can justify outputs of a model, then they are more inclined to deploy AI in real-life situations. The movement spurs research in developing methods to improve interpretability of machine learning models. Ribeiro et al. (2016) present LIME, a method that helps explain model outputs through submodular optimization. LIME decomposes the linear classifier to highlight the components that affect decision-making. Koh and Liang (2017) introduce using influence functions to measure the impact of a training example on black-box model predictions.

With tools like LIME and influence functions, black-box models are more transparent to users. However, we can take a step further to include a human in the loop during model development. To build AI that follows FATE principles, we can include interactivity to the training pipeline. Through human-AI collaboration, the human can refine the model for downstream tasks or debug the system to fix any harmful biases (Feng and Boyd-Graber, 2019). This kind of interaction is necessary to properly deploy machine learning systems in the real world. 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

[1] https://www.microsoft.com/en-us/research/theme/fate/

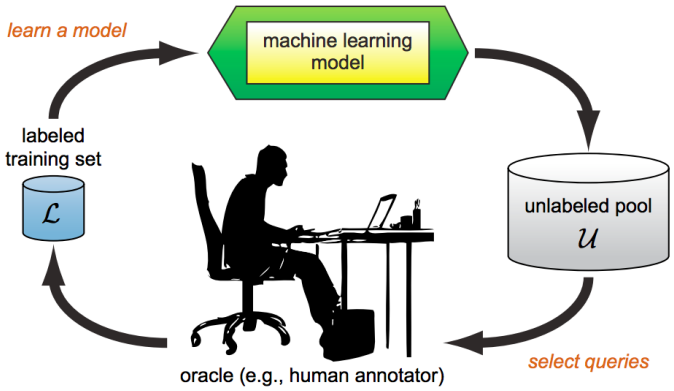


Figure 2.5: The active learning cycle involves repeatedly selecting examples in the unlabeled data pool, having an oracle annotate selected examples, adding these examples to the training set, and training the model on the new dataset (Settles, 2009).

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 operate as well for neural networks. Recent work in active learning attempt to adapt active learning for deep learning. The rest of the section looks at traditional active learning algorithms and recent work in deep active learning.

Traditional Active Learning One of the earliest work 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 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. For deep learning, uncertainty or diversity sampling may not work as well. Therefore, active learning research shifts focus toward developing algorithms for deep learning.

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 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.,

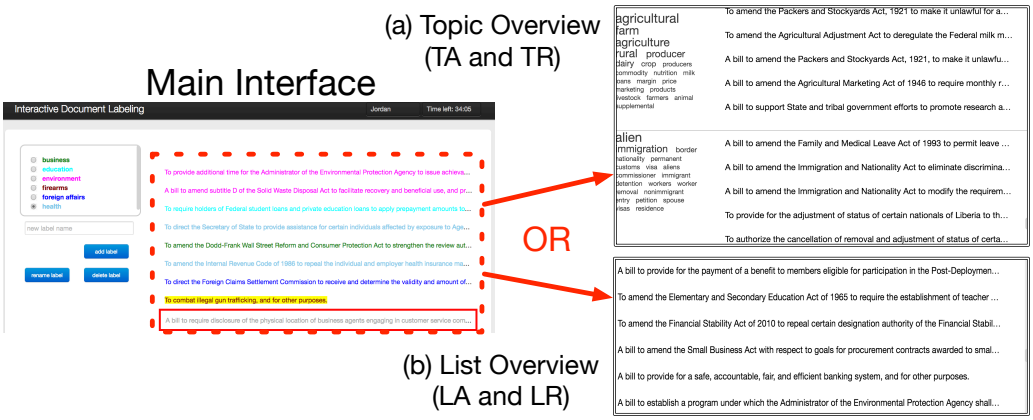


Figure 2.6: The ALTO (Poursabzi-Sangdeh et al., 2016) interface that presents documents in topic groups or sequential list order.

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 work in active learning. However, these algorithms have not been closely analyzed for NLP tasks.

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.

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 annotate both documents and words. DUALIST enhances traditional active learning through selecting features most relevant

to downstream task. User study experiments show success on word sense disambiguation, sentiment analysis, and information extraction. Blessing et al. (2013) create an interface to help political scientists apply NLP to social science research.

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 active 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 comptutation 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. However, the goal of these interactive systems focuses on interpretability and fairness (Section 2.3). In the following chapters, we use apply interaction to problems in transfer learning (Section 2.2).

Chapter 3: Active Inductive Transfer Learning[1]

3.1 Introduction

Problems in inductive transfer learning commonly occur in text for all fields. For instance, policymakers and physicians want to quickly fine-tune a text classifier to understand emerging medical conditions (Voorhees et al., 2020). Or, companies want to detect intent of clients from their messages, but the pool of intents is constantly changing (Wohlwend et al., 2019). In both situations, there is a shift in the classification task because of changes in language.

Adapting to new tasks is difficult because obtaining labeled data is costly. For example, labeling new data for medical text is challenging because of privacy issues or shortage in expertise (Dernoncourt and Lee, 2017). The transformer models can generalize across several NLP tasks (Section 2.2.1). Yet the price of adopting transformer-based models is to use more 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). 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. We develop ALPS (Active Learning by Processing Surprisal),[2] an active learning strategy for BERT-based models. While many methods randomly choose an 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

[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`

Algorithm 1 Active learning for Sentence Classification

Require: Inital 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 $M_t(\boldsymbol{x})$, data $\mathcal{U}$
  $\mathcal{D}_t \leftarrow$ 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 $f(\boldsymbol{x}; \theta_T)$

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.

3.2 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 (Section 2.2.1). 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, $f(\boldsymbol{x}; \theta) = \sigma(V \cdot h(\boldsymbol{x}; \boldsymbol{W}))$ where σ is a softmax function.

Let D be the labeled data for our classification task where the labels belong to set $\mathcal{Y} = \{1, ..., C\}$. 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 1). 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 5.4.

3.3 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.

3.3.1 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

$$(\boldsymbol{g}_x)_i = (f(\boldsymbol{x};\theta)_i - \mathbb{1}(\hat{y} = i))h(\boldsymbol{x};\boldsymbol{W}). \tag{3.1}$$

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.

3.3.2 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.

3.3.2.1 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 3.1). 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.

3.3.2.2 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 increase, 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 3.1). These issues make distance comparison between gradient embeddings less meaningful and raises costs to compute those distances.

3.4 A Self-supervised Active Learner

Cold-start active learning is challenging because of the shortage in labeled data. Prior work, 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 3.3.2.1). 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 uncertainty and diversity sampling for cold-start active learning.

3.4.1 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 masked language modeling (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

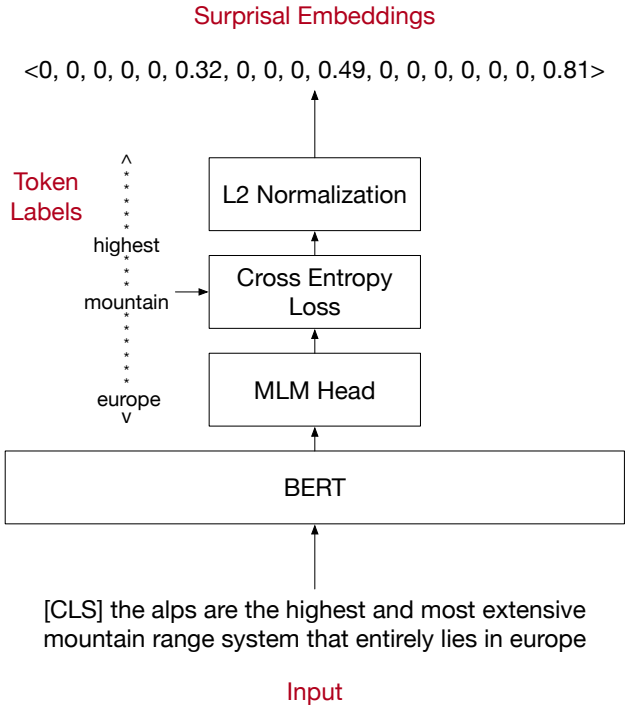


Figure 3.1: To form surprisal embedding $\boldsymbol{s}_x$, we pass in unmasked $\boldsymbol{x}$ through the BERT MLM head and compute cross-entropy loss for a random 15% subsample of tokens against the target labels. The unsampled tokens have entries of zero in $\boldsymbol{s}_x$.

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 3.1).

These decisions for not masking input and evaluating only 15% of tokens are made because of experiments on the validation set (Section 3.7). 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*

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 $\mathcal{Q}$

$(\boldsymbol{s}_x)_i$ estimates $I(w_i)$, the surprisal of its corresponding token within the context of sentence $\boldsymbol{x}$.

Proof. Extending notation from Section 3.2, 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,

$$p_m(w_i \,|\, \boldsymbol{x}) = m(\boldsymbol{x}; \phi)_{i,j}. \tag{3.2}$$

Salazar et al. (2020) have a similar definition as Equation 3.2 but instead have defined it in terms of the masked input. We argue that their definition can be extended to the unmasked input $\boldsymbol{x}$. 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 $I(w) = -\log p(w \,|\, \boldsymbol{c})$, 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,

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

□

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 3.1: Sentence classification datasets used in experiments.

Proposition 1 shows that the surprisal embeddings comprise 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 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 3.7).

3.5 Active Sentence Classification

We evaluate ALPS on sentence classification for three different domains: sentiment reviews, news articles, and medical abstracts (Table 3.1). 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 1). The initial encoder $h(\boldsymbol{x}; \theta_0)$, is an already pre-

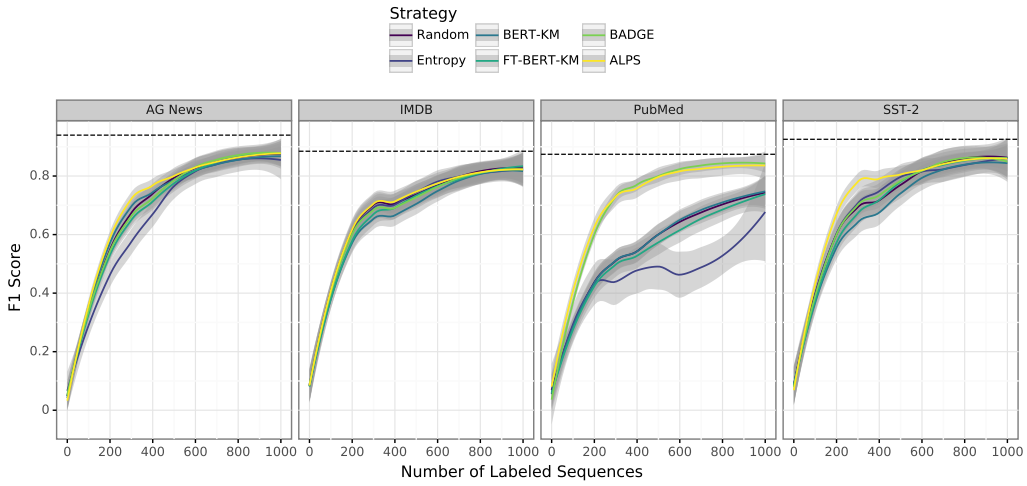


Figure 3.2: Test accuracy of simulated active learning over ten iterations with 100 sentences queried per iteration. The dashed line is the test accuracy when the model is fine-tuned on the entire dataset. Overall, models trained with data sampled from ALPS have the highest test accuracy, especially for the earlier iterations.

trained, BERT-based model (Section 3.5.2). 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.

3.5.1 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 3.3.1).

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 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.

3.5.2 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.[3] 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 3.2.

3.5.3 Results

The model fine-tuned with data sampled by ALPS has higher test accuracy than the baselines (Figure 3.2). 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 accuracy. Both methods strive to optimize for uncertainty and diversity, which alleviates problems with class imbalance.

[3] `https://huggingface.co/transformers/`

	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 3.2: 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.

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 3.2). Once the cold-start issue subsides, uncertainty-based methods can be employed to further query the most confusing examples for the model to learn.

3.6 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 3.2). 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

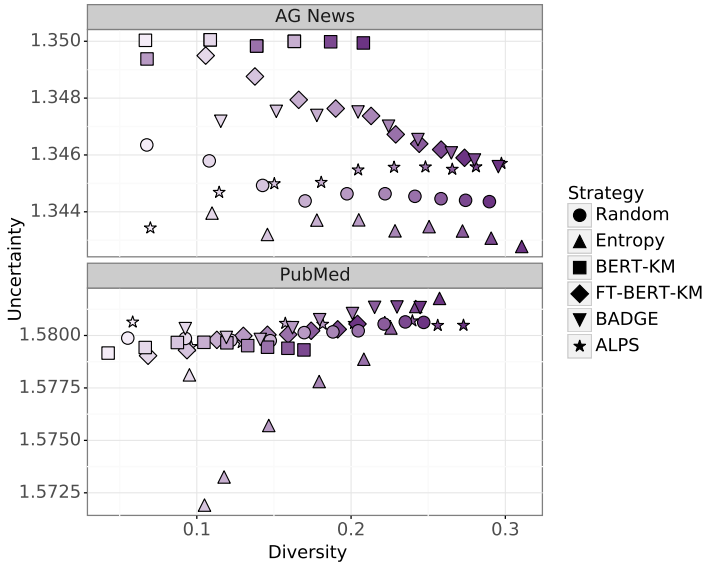


Figure 3.3: Plot of diversity against uncertainty estimates from active learning simulations. Each point represents a sampled batch of sentences. The shape indicates the active learning strategy and the color represents the sample iteration. The lightest color corresponds to the first iteration and the 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 5.4). 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 $\mathcal{U} \setminus \mathcal{D}$,

$$G_d(\mathcal{D}) = J(\mathcal{V}_D, \mathcal{V}_{D'}) = \frac{|\mathcal{V}_D \cap \mathcal{V}_{D'}|}{|\mathcal{V}_D \cup \mathcal{V}_{D'}|}. \tag{3.3}$$

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 3.2) 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 esimate uncertainty of the sampled sentences,

$$G_u(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{i=1}^{C} (f(x; \theta_*)_i) \ln(f(x; \theta_*)_i)^{-1}. \tag{3.4}$$

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 3.3). 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.

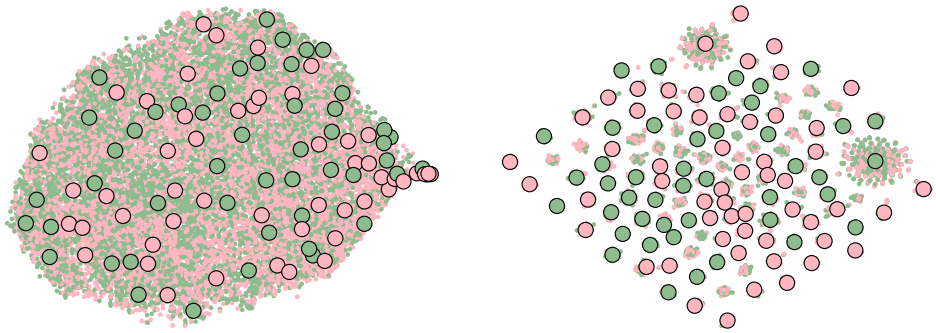
Sample Sentences We take a closer look at the kind of sentences that are sampled by ALPS. Table 3.3 compares sentences that are chosen by ALPS and random sampling in the first active learning iteration. The tokens highlighted are the ones evaluated with surprisal loss. Random sampling can fall prey to data idiosyncracies. 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 information content across the sentence, ALPS is less likely to repeatedly choose sentences with similar patterns in surprisal. This may possibly diversify syntactic structure in a sampled batch.

Surprisal Clusters Prior work use 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 3.4). 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 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.

	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)	Flu maz eni l 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 3.3: 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.

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



(a) BERT embeddings with k-MEANS centers

(b) Surprisal embeddings with k-MEANS centers

Figure 3.4: 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.

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.

Single-iteration Sampling In Section 5.4, we sample data iteratively (Algorithm 1) 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 3.4). 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.

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 3.4: 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 Section 5.4 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.

3.7 Ablation

Token masking In our preliminary experiments on the validation set, we notice improvement in accuracy after passing in the original input with no masks (Table 3.5). 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.

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 3.5). 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.

	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 3.5: 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.

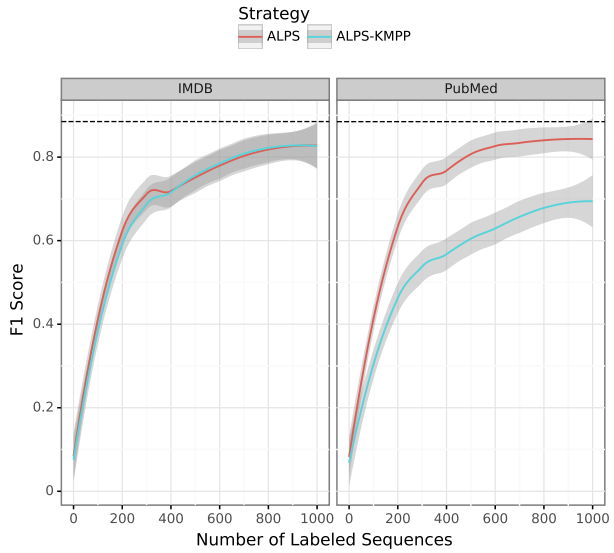
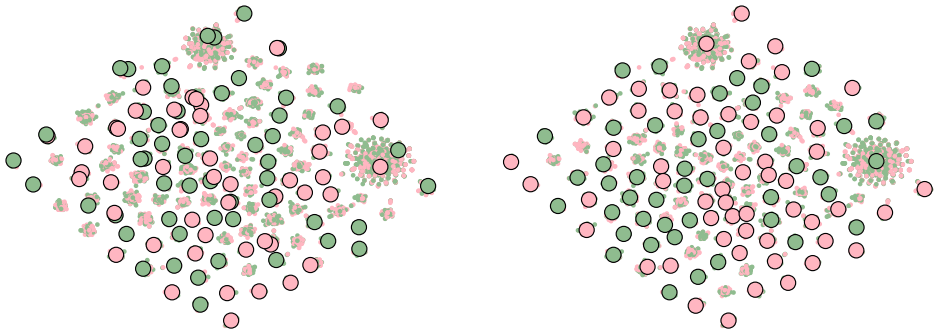


Figure 3.5: Comparing validation accuracy between using k-MEANS and k-MEANS++ to select centroids in the surprisal embeddings. Using k-MEANS reaches higher accuracy.



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

(b) Surprisal embeddings with k-MEANS centers

Figure 3.6: 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 3.5). Additionally, the t-SNE plots show that k-MEANS selects centers that are further apart compared to the ones chosen by k-MEANS++ (Figure 3.6). This shows that k-MEANS can help sample a more diverse batch of data.

3.8 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 annotations 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.

In the next chapter, we focus on application of interactive learning to transductive transfer learning. Additionally, we investigate how models learn from word-level feedback in addition to instance-level labels. We hold a user study with participants to understand the effect of human-in-the-loop methods for cross-lingual text classification.

Chapter 4: Interactive Transductive Transfer Learning [1]

4.1 Introduction

Chapter 3 focuses on active learning techniques to adapt the source model on target classification tasks. This chapter investigates interactive learning methods for shifts in the feature space. In particular, we look at text classification across 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 (Mikolov et al., 2013b, CLWE). 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 them? 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] CLIME builds upon active learning to obtain word-level feedback, rather than instance-level labeling, that can modify the word embedding space. First, CLIME uses loss gradients in downstream tasks to find keywords with high salience (Section 4.2.1). Focusing on these keywords allows the bilingual user to most efficiently refine CLWE by marking their similarity or dissimilarity (Section 4.2.2). After collecting annotations, CLIME pulls similar words closer and pushes dissimilar words apart (Section 4.3). Feedback from bilingual users establishes desired geometry in the embedding space (Figure 4.1).

In our user study, we compare CLIME with an active learning baseline that

[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*
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.

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

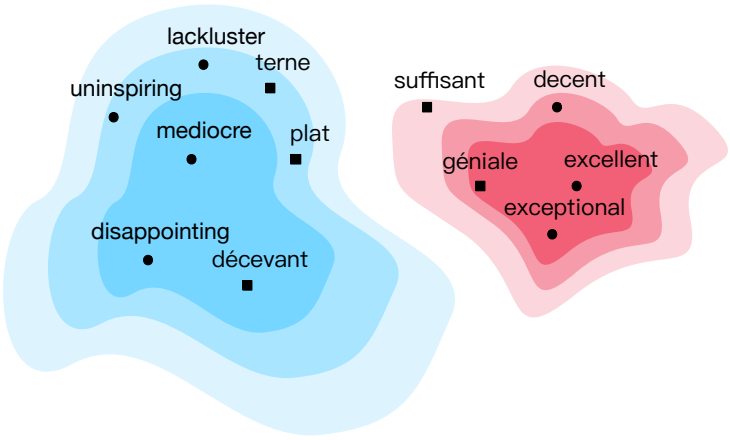


Figure 4.1: A hypothetical topographic map of an English–French embedding space tailored for sentiment analysis. Dots are English words, and squares are French words. Positive sentiment words are grouped together (red), while negative sentiment words are placed together (blue).

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.

4.2 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.

4.2.1 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

$$S_{\boldsymbol{x}}(w_i) = \left\| \nabla_{\boldsymbol{E}_{w_i}} L(f(\boldsymbol{x}), y) \right\|_2 . \tag{4.1}$$

Equation 4.1 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:

$$S(w) = \text{IDF}(w, \boldsymbol{X}) \cdot \sum_{\boldsymbol{x} \in \boldsymbol{X} : w \in \boldsymbol{x}} S_{\boldsymbol{x}}(w). \tag{4.2}$$

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 4.2, 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.

4.2.2 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 "génial" but dissimilar to

Figure 4.2: 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.

"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 5.3). 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. The added word can then be marked as positive or negative. Section 4.3 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 word appears—to disambiguate and clarify words. Users can click on any word to find example sentences.

4.3 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.

4.3.1 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 4.2.2). 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:

$$C_f(\boldsymbol{E}) = \sum_{k \in \mathcal{K}} \left(\sum_{n \in \mathcal{N}_k} \boldsymbol{E}_k^\top \boldsymbol{E}_n - \sum_{p \in \mathcal{P}_k} \boldsymbol{E}_k^\top \boldsymbol{E}_p \right), \tag{4.3}$$

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.

4.3.2 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:

$$R(\boldsymbol{E}) = \sum_{w \in \mathcal{V}} \left\| \hat{\boldsymbol{E}}_w - \boldsymbol{E}_w \right\|_2^2. \tag{4.4}$$

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 4.3) and the regularizer (Equation 4.4):

$$C(\boldsymbol{E}) = C_f(\boldsymbol{E}) + \lambda R(\boldsymbol{E}), \tag{4.5}$$

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.

4.4 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 ask an annotator to improve the model by labeling more training documents in the target language. 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 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.

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

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.

4.4.1 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.[3] To simplify the task, we consider a binary classification problem of detecting whether the documents are associated with medical needs. Table 4.1 shows an example document. To balance the label distribution, we sample an 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

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

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.[4] Low-resource language speakers are hard to find, so we have a different number of 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 4.3). For each keyword, we show five nearest neighbors for each language. Each user provides about nine constraints for each keyword.

4.4.2 Comparisons

After receiving feedback, we update the embeddings (Section 4.3). 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 4.3) against two baselines. The first baseline is a classifier trained on original embeddings (**Base** in Figure 4.3). 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 4.3). 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

[4] `https://upwork.com/`

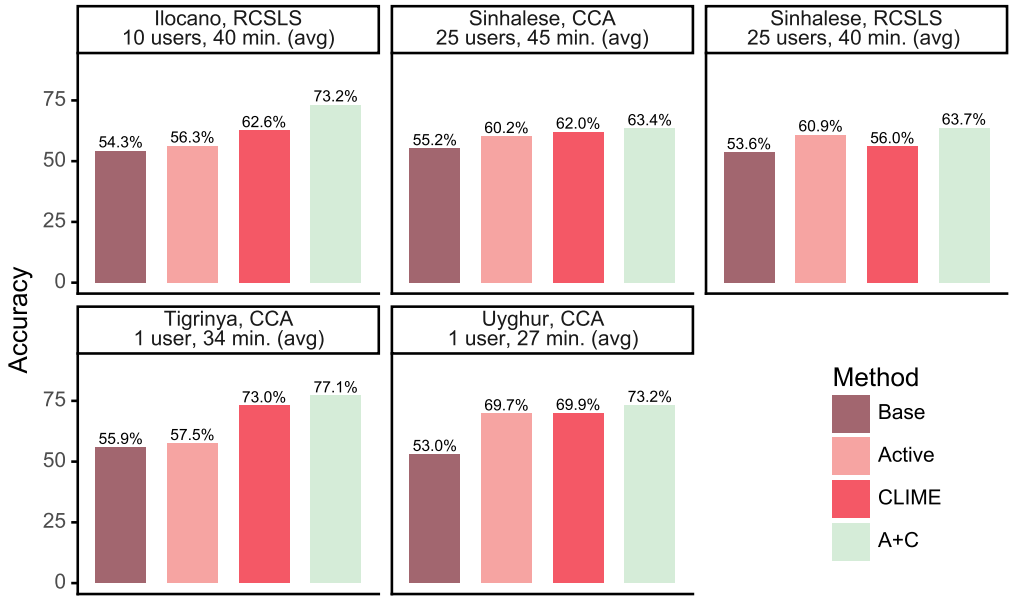


Figure 4.3: Test accuracy on four target languages and two CLWE methods. The Sinhalese and Ilocano results are averaged over multiple users, while we only have one user for other languages. Each subcaption indicates the target language, embedding alignment, number of users, and average time per user. **CLIME** has higher accuracy than **Active** on four of the five embeddings, and the combined **A**+**C** model has the highest.

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 4.3). 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 4.3). 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 4.2: 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).

4.4.3 Results and Analysis

Effectiveness of CLIME Figure 4.3 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 4.2).

Keyword Detection We inspect the list of the fifty most salient keywords (Section 4.2.1). 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 or artifacts from training data (Feng et al., 2018).

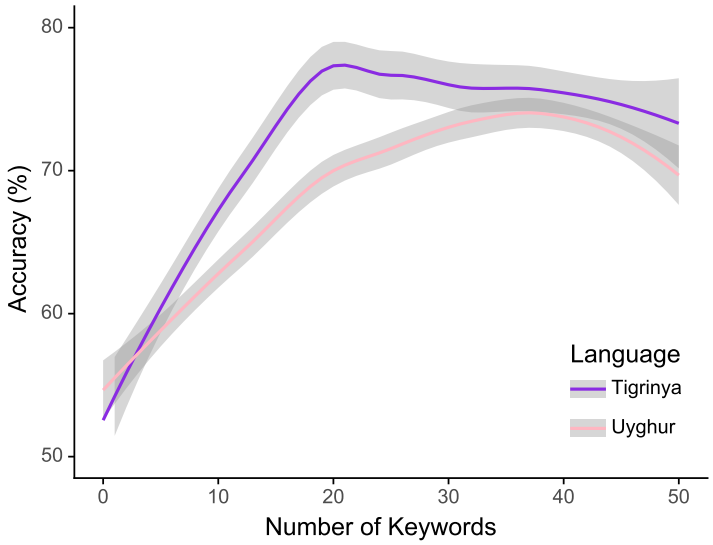
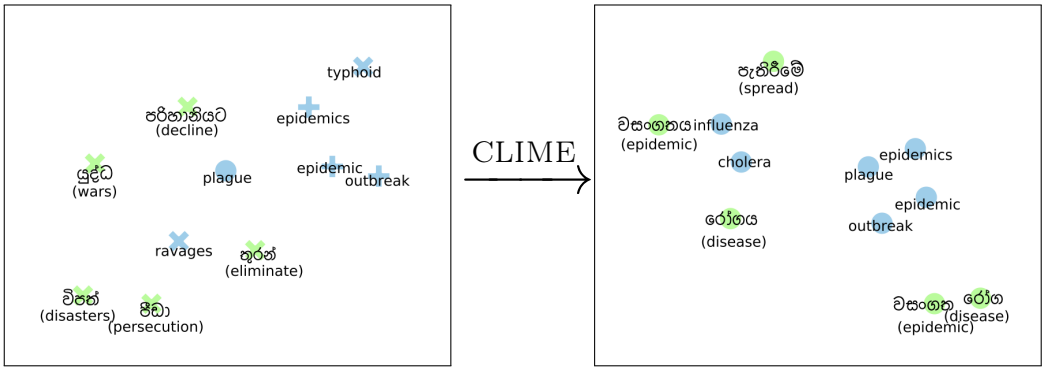


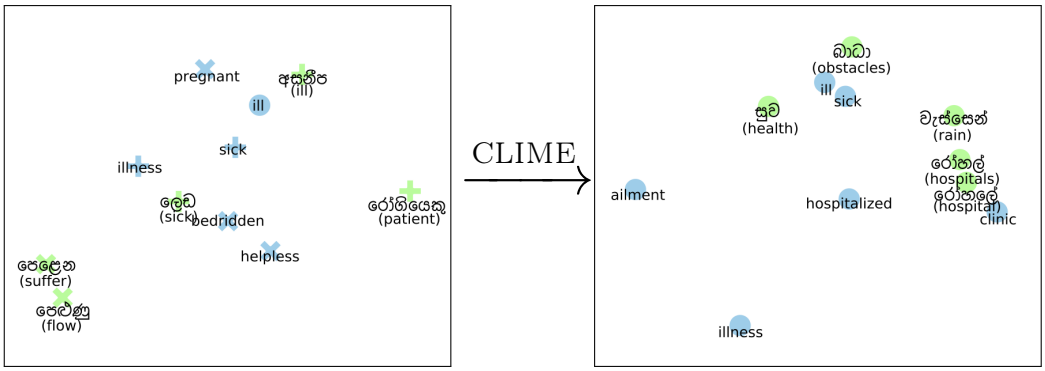
Figure 4.4: For Uyghur (pink) and Tigrinya (purple), we compare test accuracy between sets of CLWE that differ in the number of keywords used to refine them. The leftmost point corresponds to the **Base** model in Figure 4.3, while the rightmost point corresponds to the **CLIME** model. Test accuracy generally improves with more feedback at the beginning but slightly drops after reaching an optimal number of keywords.

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 4.4). 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.

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



(a) Neighborhood of "plague"



(b) Neighborhood of "ill"

Figure 4.5: 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.

Sinhalese CCA embeddings (Figure 4.5). We confirm that positive neighbors are pulled closer and negative neighbors are pushed further away. The user marks "epidemic" and "outbreak" as similar to the keyword "plague", and these words are closer after updates (Figure 4.5a). 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 4.5b). However, a few positive neighbors have inadvertently moved away, such as the

Sinhalese translation for "ill". The update algorithm tries to satisfy constraints for multiple keywords, so soft constraints may be overlooked. This motivates repeated CLIME sessions where the user can continue fixing errors.

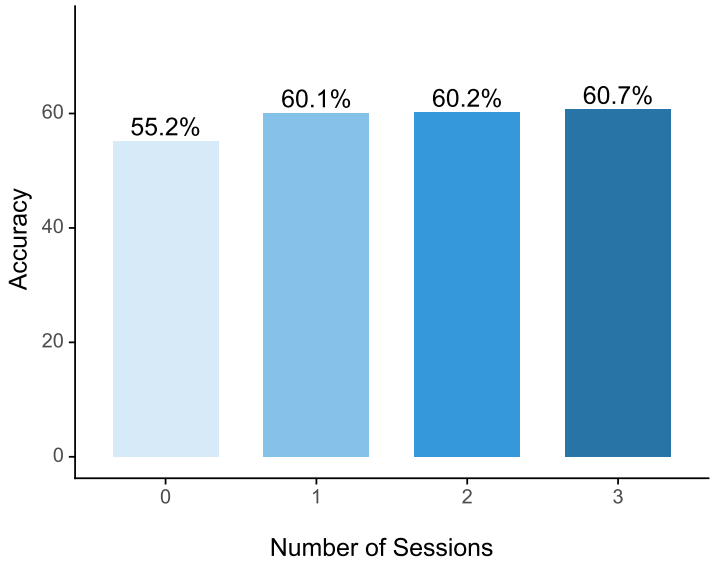


Figure 4.6: Progress of five Sinhalese users over three CLIME sessions. Largest increase in test accuracy occurs after first session. The leftmost point is the **Base** model from Figure 4.3. Average accuracy for first session is not the same as Figure 4.3 because only a subset of users are asked to complete three sessions.

4.4.4 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 4.6). By the end of the three sessions, one user reaches 65.2% accuracy, a significant improvement from the 55.2% baseline.

4.4.5 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.

4.5 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.

The next chapter focuses on problems in both inductive and transductive transfer learning. However, we do not require users to have expertise knowledge in multiple languages or domains. We delve deeper into interactive topic modeling, a useful framework for analyzing text. Whereas Chapter 3 is about active learning with instance-level labels and Chapter 4 uses word-level feedback to refine models, Chapter 5 looks at users building unsupervised topic models. The goal instead is to create interpretable topics that are also useful for downstream applications.

Chapter 5: Interactive Transfer Learning with Topic Modeling [1]

5.1 Introduction

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. Therefore, machine learning non-experts can easily use topic models to analyze text. The ease of using these models rise to interactive topic modeling (Section 2.3.2).

In this chapter, we apply interactive topic modeling to both inductive and transductive transfer learning. The setting is inductive because the topic model needs to adapt for different classification tasks. The setting is also transductive because we evaluate the topic model across different languages. Therefore, we fully investigate the effect of interactive topic modeling on both kinds of transfer learning settings.

To handle both inductive and transductive transfer learning, we propose a multilingual anchoring algorithm, which is an extension to anchor-based topic inference for comparable corpora (Section 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. These vector representations improve knowledge transfer for text classification. The topic model generates coherent topic aligments for comparable corpora because users themselves align topics.

[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`.

5.2 Anchor-based Topic Models

A computationally attractive way to model topics is the anchor word algorithm (Section 2.2.1). The algorithm uses the row-normalized word co-occurrence matrix $\bar{\boldsymbol{Q}}$, where $\bar{\boldsymbol{Q}}_{i,j} = p(w_2 = j \,|\, w_1 = i)$. 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.

5.2.1 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

$$\bar{\boldsymbol{Q}}_i \approx \sum_{k=1}^{K} \boldsymbol{C}_{i,k} \bar{\boldsymbol{Q}}_{a_k} \quad \text{subject to} \quad \sum_{k=1}^{K} \boldsymbol{C}_{i,k} = 1 \text{ and } \boldsymbol{C}_{i,k} \geq 0. \tag{5.1}$$

The coefficient $\boldsymbol{C}_{i,k}$ represents $p(z = k \,|\, w = i)$, 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 $\boldsymbol{A}$,

$$\boldsymbol{A}_{i,k} = p(w = i \,|\, z = k) \propto p(z = k \,|\, w = i) p(w = i) = \boldsymbol{C}_{i,k} \sum_{j=1}^{V} \bar{\boldsymbol{Q}}_{i,j}. \tag{5.2}$$

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 5.1), 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).

5.2.2 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,

$$\bar{\boldsymbol{Q}}_{V+k,j} = \left(\frac{\sum_{i \in \mathcal{G}_k} \bar{\boldsymbol{Q}}_{i,j}^{-1}}{|\mathcal{G}_k|} \right)^{-1}. \tag{5.3}$$

The motivation for using the harmonic mean (Equation 5.3) 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.

5.3 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

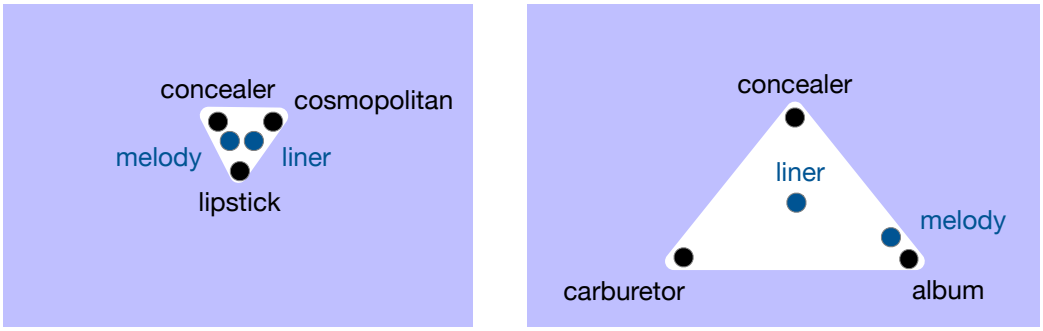


Figure 5.1: Visualizing the importance of choice in anchor words for approximating conditional distributions. The chosen anchor words are the black dots and their span is the white triangle. On the left, the span of anchor words is small, so the words "melody" and "liner" are too close together. On the right, the span of anchor words is large, so the conditional distributions of words "melody" and "liner" are approximated more accurately.

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).

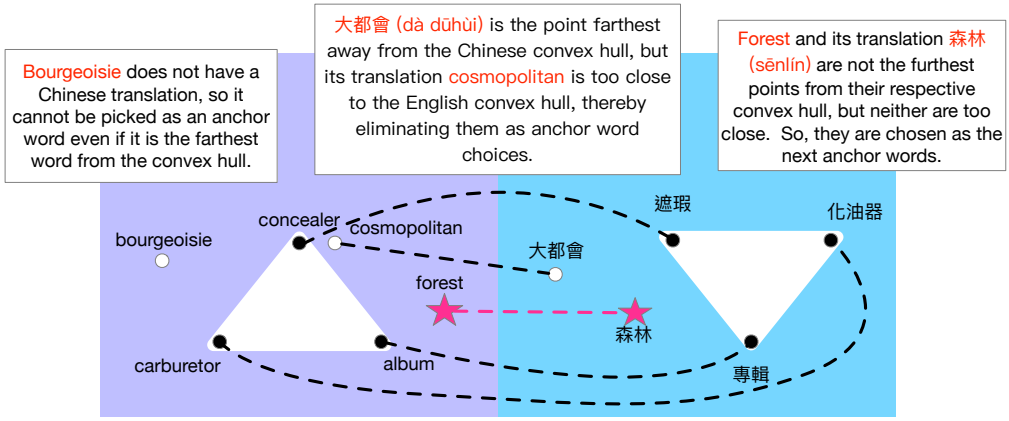


Figure 5.2: Selecting anchor links for multilingual anchoring. The purple (blue) area represents the conditional distribution space of words in the English (Chinese) corpus. The white triangle designates the space spanned by chosen anchor words. Dashed lines depict anchor links across spaces. Black points 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.

5.3.1 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 5.2). 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 5.1). 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 word a_k such that

$$a_k = \arg\max_w d\left(\text{span}\left(\bar{\boldsymbol{Q}}_{a_1}, ..., \bar{\boldsymbol{Q}}_{a_{k-1}}\right), \bar{\boldsymbol{Q}}_w\right), \tag{5.4}$$

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,

$$\mathcal{P}_j^l = \text{span}\left(\bar{\boldsymbol{Q}}^l_{a_1^l}, ..., \bar{\boldsymbol{Q}}^l_{a_j^l}\right). \tag{5.5}$$

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 5.2). 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

$$a_k^S, a_k^T = \arg\max_{w,v} \min\left\{d\left(\mathcal{P}_{k-1}^S, \bar{\boldsymbol{Q}}_w^S\right), d\left(\mathcal{P}_{k-1}^T, \bar{\boldsymbol{Q}}_v^T\right)\right\} \quad \text{subject to } (w,v) \in \mathcal{B}. \tag{5.6}$$

We greedily select anchors $a_k^S \in \mathcal{V}_S, a_k^T \in \mathcal{V}_T$ such that Equation 5.6 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 5.3.2). After picking a set of anchor words for each language, multilingual anchoring follows FastAnchorWords (Section 5.2.1). Topic matrices $\boldsymbol{A}_S$ and

$\boldsymbol{A}_T$ are separately recovered (Equations 5.1, 5.2). 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.

5.3.2 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 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 comparable corpora and a bilingual dictionary as inputs. Next, it uses multilingual anchoring (Section 5.3.1) 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 5.3). 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)

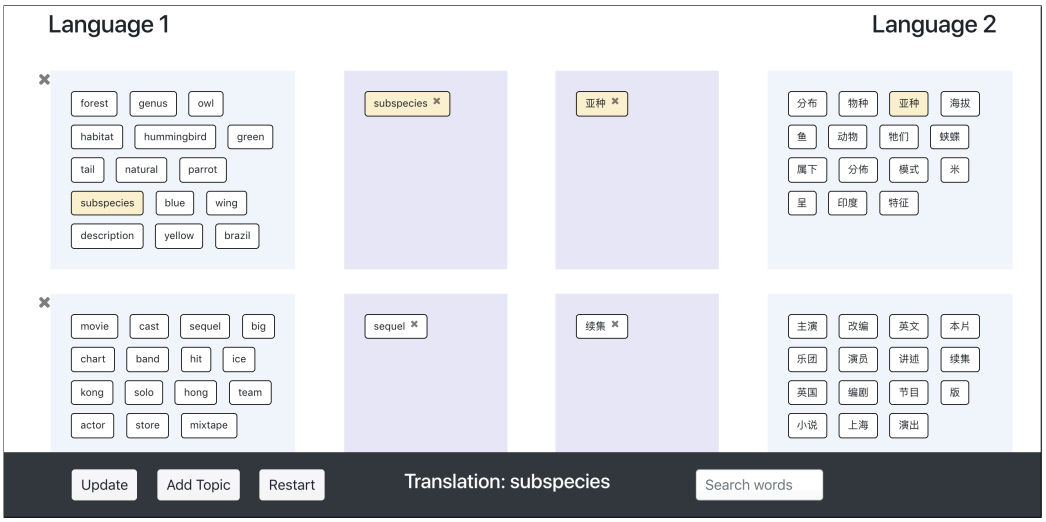


Figure 5.3: The user interface for exploring topics in English and Chinese documents. Anchor words are in the center, while the most likely words for each topic are on the left and right sides of the interface. The user can drag words from the side and add them as anchor words. When the user hovers over "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.

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 5.2.2). 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 5.6. After the initial alignment, users define anchors and customize the topic model to their own needs.

5.4 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 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 are 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 of unique word types, rather than number

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

[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 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.

5.4.1 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 5.3.1. 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 the algorithm's ability to generalize). For topic coherence, we use the NPMI (normalized pointwise mutual information) variant of automated topic intepretability 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.

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

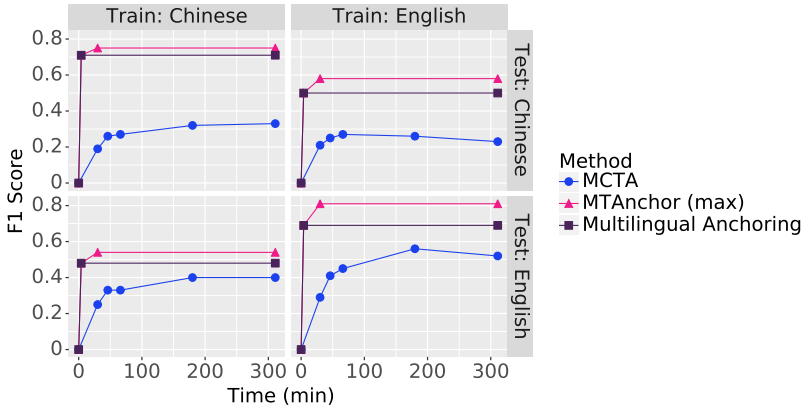


Figure 5.4: Classification accuracy over time until MCTA converges. For the Wikipedia dataset, multilingual anchoring converges within 5 minutes, but MCTA takes 5 hours and 18 minutes to converge. Multilingual anchoring outperforms MCTA in speed and classification accuracy.

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 5.1: 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.

5.4.2 Results

In experiments, multilingual anchoring converges much faster than MCTA (Figure 5.4). We compare test accuracy across experiments for multilingual anchoring, MTAnchor, and MCTA, but only report the maximum and median scores from

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 5.2: 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.

MTAnchor user experiments (Table 5.1). We also compare topic coherence across all experiments (Table 5.2). 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 spending more time interacting with the model (Figure 5.5). Within thirty minutes, a user can improve topic coherence and reach up to a 0.40 increase in any one of the classification scores.

5.5 Discussion

Multilingual anchoring is a spectral approach to modeling multilingual topics. The algorithm converges much faster than generative methods (Figure 5.4) and resulting topics form better vector representations for documents (Table 5.1). 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 con-

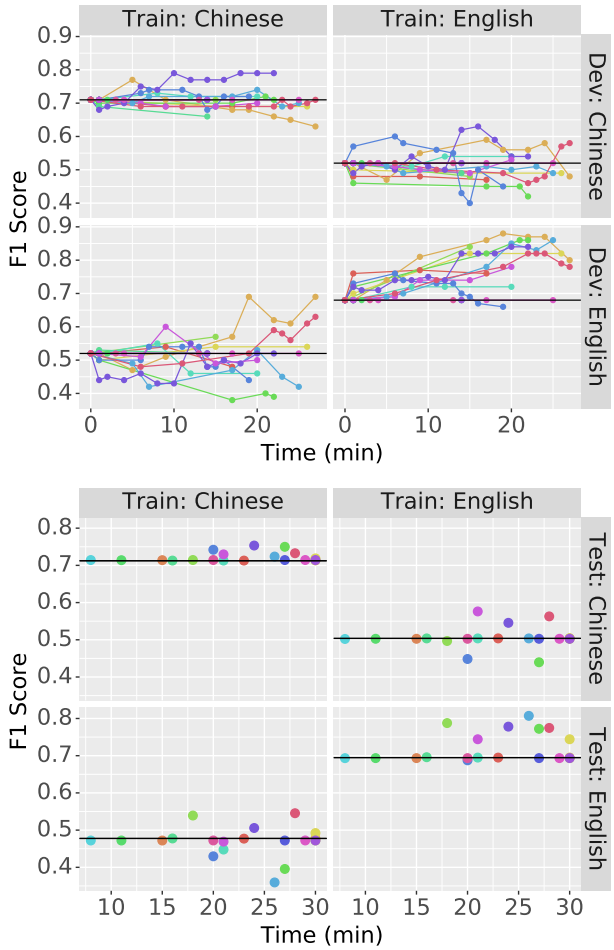


Figure 5.5: Classification accuracy of each participant in the MTAnchor user study over time. Each plot indicates the language of topics that the classifier is trained on and the language of topics that the classifier is tested on. The black horizontal line denotes multilingual anchoring score (no interactive 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).

tains reviews with only one to three sentences. The health topic for multilingual anchoring is more interpretable than that of MCTA (Table 5.3).

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 5.3: 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.

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" and "nbro" (Table 5.3). 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.

5.6 Conclusion

Interactive topic modeling show promising results for human-in-the-loop transfer learning. First, we notice that users can create more coherent topics. Second, a few of them can significantly improve classification accuracy. While constructing an interpretable topic model, they are also generating document representations for a classification task. This exemplifies inductive transfer learning: constructing coherent topics, the source task, transfers knowledge over to text classification, the target task. There is also a cross-lingual component, which involves transductive transfer learning. Through choosing anchor words, users provide information needed to bridge the shift in tasks and domains.

So far, the past chapters have applied interactive feedback to text classification. However, many NLP tasks are more complex than document-level labeling. 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 propose future work that uses interactive learning to transfer knowledge for difficult NLP problems.

Chapter 6: Proposed Work

6.1 Interactive Learning for Natural Language Understanding

The chapters in this proposal are mainly focused on text classification. While text classification can generalize to many tasks, there are more challenging and interesting NLP problems. Researchers have proposed benchmark datasets to evaluate natural language understanding. The GLUE and SuperGLUE datasets include tasks like question answering, natural language inference, reading comprehension, and commonsense reasoning (Wang et al., 2019b,a). Hu et al. (2020) introduce XTREME, a multilingual benchmark dataset covering forty languages. To reach high accuracy on these datasets, models need to successfully conduct transfer learning. Therefore, we focus on interactive learning for tasks that require mastery of natural language.

Coreference resolution is a difficult problem for NLP models because it requires understanding the links between entities in the text. The goal of a coreference resolution is to correctly predict the antecedent of each entity mention. For example, a model needs to know that "it" refers to "my dog" in: "I walk my dog while calling my friend and it starts to bark at the birds". This skill is essential for other NLP tasks like summarization and information retrieval. Several works apply their models to the OntoNotes 5.0 dataset (Pradhan et al., 2012). A pretrained SpanBERT model (Joshi et al., 2020) can successfully link mentions for OntoNotes after exhaustive fine-tuning on OntoNotes. However, the same model may fail to link mentions for other datasets, such as PreCo (Chen et al., 2018a), LitBank (Bamman et al., 2020), and cross-lingual data from SemEval-2010 (Recasens et al., 2010). The PreCo dataset contains text with various writing styles, whereas LitBank provides documents that are four times as long as the sentences in OntoNotes. The SemEval-2010 data contains coreference resolution examples for languages like German, Catalan, Spanish, Dutch, and Italian.

For the proposed work, we will use interactive feedback to transfer knowledge between coreference resolution models. Given a source model already fine-tuned on OntoNotes, how can we efficiently label text spans to quickly transfer the model for other coreference resolution datasets? How can we transfer information across

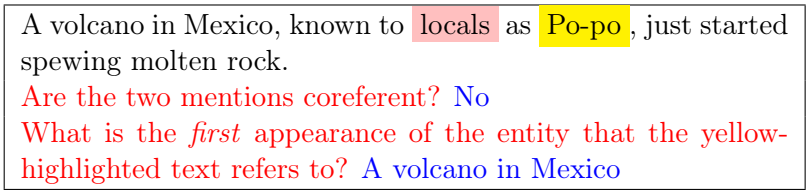


Figure 6.1: An example annotation interface for coreference resolution (Li et al., 2020). The annotator is shown the document, a span (yellow), and the span's predicted antecedent (pink). If the pair of spans are not coreferent, the interface presents a follow-up question to correct mention linking.

different annotation guidelines and languages? We envision that a human in the loop can quickly correct any wrong references of entity mentions. Our plan is to select text spans that can improve *mention detection* and *mention linking*. To improve mention detection, we can look at the *mention score*, which is the probability of the span being an entity mention. To improve mention linking, we can look at the *clustered entropy*, which is the entropy over antecedent cluster predictions for a span. The challenge is to balance the trade-off between mention detection and mention linking. We will investigate ways to leverage this trade-off through some scoring function, a fixed combination, or a bandit-learning approach (Hsu and Lin, 2015).

For the user interaction component, we will build an interface similar to prior work (Figure 6.1). Li et al. (2020) conduct user experiments to evaluate their active learning approaches for coreference resolution. They only focus on mention linking, so they simply ask whether a pair of spans are coreferent. If not, they then ask the user to provide the correct antecedent for a given span. We will extend their interface to also resolve issues with mention detection. A coreference resolution system must consider both mention detection and mention linking (Wu and Gardner, 2020; Lu and Ng, 2020). We will also include additional interactive features that can help streamline user annotation. For example, Klie et al. (2020) design an interface for low-resource entity linking that shows entity candidate list and other information about the entities in text. In the proposed work, we would like to investigate the type of features that help improve coreference resolution.

Along with coreference resolution, question answering is another area of interest for interactive feedback. Wallace et al. (2019) collect adversarial examples from users to analyze and strengthen question answering systems. Possibly, proposed work could look at humans creating adversarial questions for training question answering systems. Kratzwald et al. (2020) propose a human-in-the-loop framework for question answering that involves learning a cost-effective policy and a semi-

supervised annotation scheme. Their annotation setup resembles CLIME (Chapter 4) because users provide binary feedback. Thus, we may build upon their work and CLIME for cross-lingual question answering. All in all, the proposed work will focus on applying interaction to problems that require natural language understanding.

Appendix A: Reading List

A.1 Inductive Transfer Learning

1. 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

2. 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

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

4. Hanna M Wallach. 2006. Topic modeling: Beyond bag-of-words. In *Proceedings of the International Conference of Machine Learning*

5. 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*

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

7. 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

8. 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*

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

10. 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*

A.2 Transductive Transfer Learning

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

2. 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*

3. 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*

4. 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*

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

6. 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*

7. 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

8. 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

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

10. 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*

A.3 Interactive Learning

1. 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*

2. 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*

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

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

5. 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*

6. 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*

7. 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*

8. 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*

9. 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

10. 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*

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.

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*.

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*.

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.

Andre Blessing, Jonathan Sonntag, Fritz Kliche, Ulrich Heid, Jonas Kuhn, and Manfred Stede. 2013. Towards a tool for interactive concept building for large scale analysis in the humanities. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*.

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*.

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*.

Manuel Castells. 1996. *The information age*. Oxford Blackwell Publishers.

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. 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.

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*.

Peter T. Daniels and William Bright. 1996. *The world's writing systems*. Oxford University Press.

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*.

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.

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*.

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.

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.

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.

Yuval Noah Harari. 2014. *Sapiens: A brief history of humankind*. Random House.

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*.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *Proceedings of the International Conference of Machine Learning*.

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*.

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*.

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*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the International Conference of Machine Learning*.

Bernhard Kratzwald, Stefan Feuerriegel, and Huan Sun. 2020. Learning a cost-effective annotation policy for question answering. 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*.

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 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*.

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*.

Jing Lu and Vincent Ng. 2020. Conundrums in entity reference resolution. 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*.

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*.

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*.

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.

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.

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*.

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, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual

unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*.

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.

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*.

Marta Recasens, Lluís Màrquez, Emili Sapena, M. Antònia Martí, Mariona Taulé, Véronique Hoste, Massimo Poesio, and Yannick Versley. 2010. SemEval-2010 task 1: Coreference resolution in multiple languages. In *Proceedings of the Workshop on Semantic Evaluation*.

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

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?" explaining the predictions of any classifier. In *Knowledge Discovery and Data Mining*.

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.

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.

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*.

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*.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

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*.

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*.

Bernard Vauquois. 1968. A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In *IFIP Congress-68*.

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.

Hanna M Wallach. 2006. Topic modeling: Beyond bag-of-words. In *Proceedings of the International Conference of Machine Learning*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Proceedings of Advances in Neural Information Processing Systems*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the International Conference on Learning Representations*.

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

Ian Watt. 1957. *The Rise of the Novel: Studies in Defoe, Fielding, and Richardson*. Chatto & Windus, London.

Jeremy Wohlwend, Ethan R Elenberg, Sam Altschul, Shawn Henry, and Tao Lei. 2019. Metric learning for dynamic text classification. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*.

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*.

Zhaofeng Wu and Matt Gardner. 2020. Understanding mention detector-linker interaction for neural coreference resolution. *arXiv preprint arXiv:2009.09363*.

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.

Wai-lim Yip. 1997. *Chinese Poetry, Revised: An Anthology of Major Modes and Genres*. Duke University Press.

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, 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*.

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*.