NATURAL LANGUAGE TECHNOLOGIES FOR LOW-RESOURCE LANGUAGES

by

Young-Bum Kim

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2015

Date of final oral examination: 05/14/2015

The dissertation is approved by the following members of the Final Oral Committee:

Benjamin Synder, Assistant Professor, Computer Sciences
Xiaojin Zhu, Associate Professor, Computer Sciences
Jude Shavlik, Professor, Computer Sciences
Mark Craven, Professor, Biostatistics and Medical Informatics
Alan Ritter, Assistant Professor, Computer Science and Engineering, Ohio University
Asli Celikyilmaz, Senior Scientist, Conversational Understanding Sciences, Microsoft

For my mother, who always prayed for me.

ACKNOWLEDGMENTS

I am grateful to have had an opportunity to pursue my graduate studies at an elite institution like UW-Madison. Fortunately my stay here coincided with the time during which Associate Professor Benjamin Snyder started his affiliation with the university. Throughout my five years at UW-Madison, he has provided me boundless support and guidance to pursue my research activities. This thesis would not been possible without his continued support and motivation. I would like to sincerely thank my thesis advisor Dr. Benjamn Snyder for the trust he had on me and for his efforts in shaping my thesis.

Following which I would like to convey my sincere gratitude to my thesis committee. Their patience and valuable inputs have helped me in improving my thesis to a great extent. In particular, Xiaojin (Jerry) Zhu, Mark Craven, Asli Celikyilmaz, and Alan Ritter have consistently made themselves available for research discussions. Much of the work outlined in this thesis has its seed in these discussions.

I was fortunate to have the opportunity to do two internships at Microsoft Research, both of which were valuable experiences for my thesis. During these short stays at Microsoft, excellent mentors in the form of Ruhi Sarikaya, Asli Celikyilmaz, Anoop Deoras, Tasos Anastasakos and Minwoo Jeong were extremely helpful.

Next, I would also like to thank my colleague Karl Stratos from whom I benefited greatly. Karl is incredibly knowledgeable, and I feel like he has really gone out of his way to discuss research and help me, despite his busy schedule.

I happened to have two excellent advisors as an undergraduate at Hallym University: Chang Geun Song and Yu-Seop Kim, both of whom encouraged my interest in pursuing research and also opened up many opportunities for me. Also I take this opportunity to thank Do Kook Choe,

Jae-Sung You, Dongchan Kim, Geunsoo Lee and Heemoon Chae who have gone out of their way to support me throughout graduate school.

Finally I would like to thank my loving mother, Eunmi Lee. Her boundless support and constant encouragement has gone great lengths in shaping this PhD thesis. I am grateful for all the sacrifices she has made for my success.

DISCARD THIS PAGE

TABLE OF CONTENTS

Page

Appendix

Page

Appendix

Page

DISCARD THIS PAGE

LIST OF TABLES

Table Page

DISCARD THIS PAGE

LIST OF FIGURES

Figure Page

Appendix
Figure Page

ABSTRACT

Over the last decades, supervised statistical learning has become the dominant paradigm for building state-of-the-art natural language technologies. The primary difficulty in achieving good performance with these models is that they necessitate extensive annotated linguistic resources. Such resources often require prohibitive human efforts, and therefore they exist for only a few major languages out of the thousands of native languages spoken today. By targeting only a few widely spoken languages, we leave behind a large percentage of the world's population. To resolve this problem, I propose novel techniques to robustly handle languages lacking annotated resources. Two lines of work will be presented. For the first thread of my argument, I assume that no annotated resources are available for the target language. I propose a joint analysis of a broad array of languages, whereby languages with annotated resources can be used as training data for resource poor languages. In the second thread of my argument, I assume that there is only limited budget or time for supervised annotation. I propose two techniques for identifying an optimal set of examples to be labeled, in order to produce a high-performance supervised model.

Chapter 1

Introduction

Communicating through natural languages is one of the defining characteristics of modern Homo Sapiens. Written languages, developed over the past several thousand years, have increased the human ability to communicate across space and time, and to accumulate knowledge over generations. In recent years, the accessibility of this information has increased tremendously with the advent of digital texts. The Internet, in particular, allows people to communicate nearly anytime and anywhere, and has revolutionized and democratized the spread of information. Among the digital forms of texts, sounds, images, and videos, the texts remains the dominant means of communication between people. We face a vast amount of texts everyday from news articles, magazines, emails, and social media.

The information overload from these vast textual resources makes it challenging to organize and filter out pertinent information. The difficulties arise for various reasons. First, not all available information is relevant to everyone. Second, the validity of the information is not always guaranteed. Therefore, an automatic way to intelligently process digital texts is highly desirable. Researchers in the field of natural language processing (NLP) have been proposing various methods to tackle the challenges of automated text analysis and understanding.

Most state-of-the-art methods currently used in NLP are based on supervised learning, which for the most part requires detailed linguistic annotations for training a model. Unfortunately, constructing such annotated resources is expensive and time consuming. Consequently, such resources exist for only a few major languages out of the thousands of native languages spoken today. By targeting only a few widely spoken languages, we leave behind a large percentage of the world's population. Figure 1.1 showcases the difficulty here: the three most spoken languages in the world

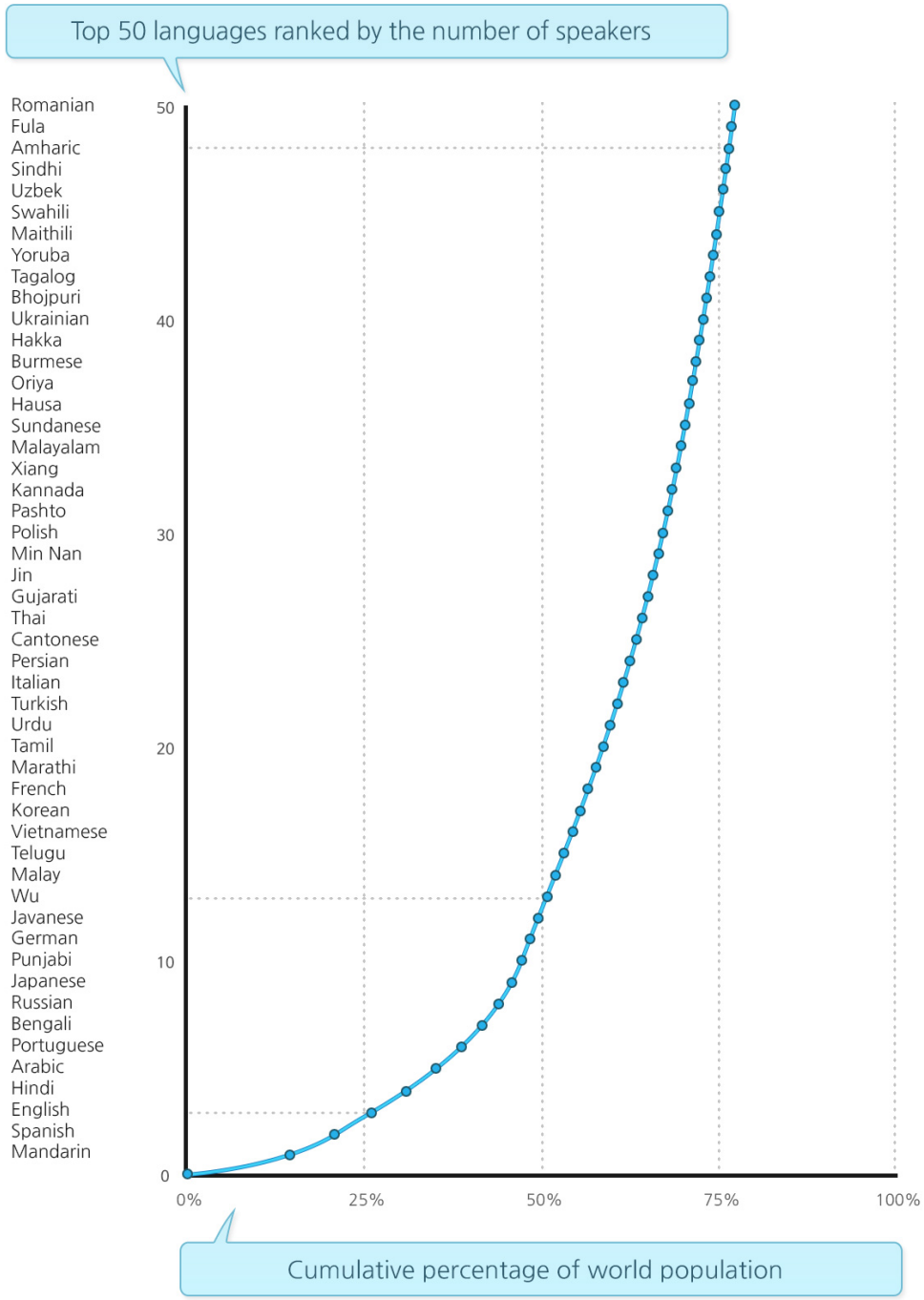

Figure 1.1: A quartiles chart of languages by percentages of speakers in the world population. The horizontal axis gives the cumulative percentage of speakers and vertical axis gives the top 50 languages ranked by the number of speakers.

(Mandarin, Spanish, and English) only handle 25% of world population. And then the number of required languages grows quickly as our desired coverage increases: 13 languages to cover 50% of the population, 47 languages to cover 75%. Thus it is clear that NLP models for a very wide variety of languages will be needed to cover the native languages of most people in the world.

In this thesis, we will propose novel techniques to robustly handle languages lacking annotated resources. Two arguments will be presented for the problem of paucity of the annotated resources, used in supervised statistical learning. For the first thread of our argument, we assume that no annotated resources are available for the target language. This assumption is clearly valid for the vast majority of human languages, and is certainly true when we consider the task of deciphering lost languages. We propose a joint analysis of a broad array of languages, whereby languages with annotated resources can be used as training data for resource-poor languages. In the second thread of our argument, we assume that there is only limited budget or time for supervised annotation. We propose two techniques for identifying an optimal set of examples to be labeled, in order to produce a high-performance supervised model.

We will refer to these two strands of research as (i) cross-lingual supervised learning and (ii) optimal dataset selection. In the following sections of this chapter, we introduce some backgrounds for these research lines and give an overview of our results.

1.1 Cross-lingual Supervised Learning

An influential line of prior multilingual work starts with the observation that rich linguistic resources exist for some languages but not for others. The key idea is then to *project* linguistic information from one language onto others via parallel data. First, Yarowsky and his collaborators [107; 106; 105] developed this idea and applied it to the problems of part-of-speech tagging, noun-phrase bracketing, and morphology induction, and other researchers have applied the idea to syntactic and semantic analyses [51; 80]. In these cases, the existence of bilingual parallel texts along with highly accurate predictions for one of the languages was assumed.

Another line of work assumes the existence of bilingual parallel texts without the use of any supervision [24; 86]. This idea has been developed and applied to a wide variety tasks, including

morphological analysis [92; 91], part-of-speech induction [95; 96; 76], and grammar induction [94; 10; 14]. An even more recent line of work does away with the assumption of parallel texts and performs joint unsupervised induction for various languages through the use of coupled priors in the context of grammar induction [18; 7].

In contrast to these previous approaches, the methods proposed in this thesis do *not* assume the existence of any parallel text, but *do* assume that labeled data exists for a variety of languages to be used as training examples for the test language. In other words, I recast the induction of linguistic structures as a *supervised* learning problem, treating each language as a single data point.

Formally, each language consists of an input-label pair (x, y), both of which contain complex internal structures. The input $x \in \mathcal{X}$ consists of all observed properties of raw texts for a language, including, for example, a vocabulary list of all words in a particular monolingual corpus. The corresponding label $y \in \mathcal{Y}$ consists of the correct linguistic analyses of all the observation items in x, for example the part-of-speech of the various words in the vocabulary. The goal then is to build a universal linguistic predictor $\mathcal{X} \mapsto \mathcal{Y}$. This predictor should be able to map the raw texts of any target language to a plausible linguistic analyses, by learning general characteristics of language structures from observed languages. For this to work, we first define a feature function $\mathbf{f}$ which maps each (x, y) pair into an abstract, universal, feature space:

$$\mathbf{f} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$$

The key criterion for choosing the universal feature space is that it should consist of abstract linguistic features that generalize well across languages for the task at hand. As a training data, we use languages, $x_1, \ldots, x_n$, for which corresponding labels $y_1, \ldots, y_n$ are available. We use this data to train a scoring function over the universal feature space:

$$score : \mathbb{R}^d \rightarrow \mathbb{R}$$

The key idea is that $score$ should assign high weights to plausible language-labeling pairs, and low weights to generally implausible interpretations. Finally, for a target language $x \in \mathcal{X}$, we make a prediction based on the scoring function:

$$y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, score\big(\mathbf{f}(x, y)\big)$$

Now, we will describe how we applied the proposed framework to three different tasks: morphological analysis, grapheme-to-phoneme analysis of Latin alphabets, and phonetic decipherment of unknown scripts. Table 1.1 gives an overview of how we applied this framework for these tasks. In each case, the cross-lingual supervision substantially improved performance over strong unsupervised baselines, and our findings were presented at NLP conferences and published in the proceedings.

It is worth noticing the number of languages covered by proposed methods on a year-on-year basis. In 2011, the methods had handled only 8 languages, which (if these had been the most widely spoken languages in the world) would still cover less than 50% of the world population (See Figure 1.1). In 2013, as the number of languages included increased to 503, the world-wide coverage would theoretically reach 75%. In the sense that the methods achieved such high coverage without human annotation effort, we are optimistic about the effectiveness of the framework. Note that for all tasks, we applied a round-robin approach, where we evaluated each language in the sample in turn as the test language, and trained the model on all the remaining languages. Now we now turn to the task of morphological analysis.

1.1.1 Morphological Analysis

Morphological analysis is a task of identifying the morphemes within words, which is an initial step for many downstream tasks in NLP. Here the goal is to automatically decompose a given word into a stem, an optional phonological deletion rule, and an optional suffix. See Figure 1.2 for the details.

In this task, the input $x \in \mathcal{X}$ consists of vocabulary list of all words in a monolingual corpus, in this case consisting largely of Eastern European languages, which tend to have very rich morphology. The label $y \in \mathcal{Y}$ consists of the correct morphological analysis of all the vocabulary items in x. For a universal feature function $\mathbf{f}$, we employed a fairly simple and minimal set of features: 1) size of stem, suffix and deletion rules of lexicons, 2) entropy of corresponding distributions, and

Chapter	2	3	5
Publication	EMNLP 2011	EMNLP 2012	ACL 2013
# of Langs	8	107	503
Task	Morphological Analysis	Grapheme-to-Phoneme Analysis	Decipherment
Method	Structured Nearest Neighbor	Markov Random Fields	Bayesian Hierarchical Model
$\mathcal{X}$	Vocab list for Eastern European languages	Vocab list for languages in Latin Alphabet	Vocab list for world languages in different alphabets
$\mathcal{Y}$	Morphological analysis	Grapheme-Phoneme mapping	Phonetic properties
f	**Abstract properties of morphological lexicon:** relative frequency of stems vs suffixes. Entropy of distributions over morphemes	**Historical properties of Latin alphabet:** Textual context of graphemes. Articulatory features. Language family and region	**Linguistic properties of phoneme sequences:** Transitional regularities between phonemic properties. Size of phoneme categories.

Table 1.1: Overview of our publications for cross-lingual supervised learning

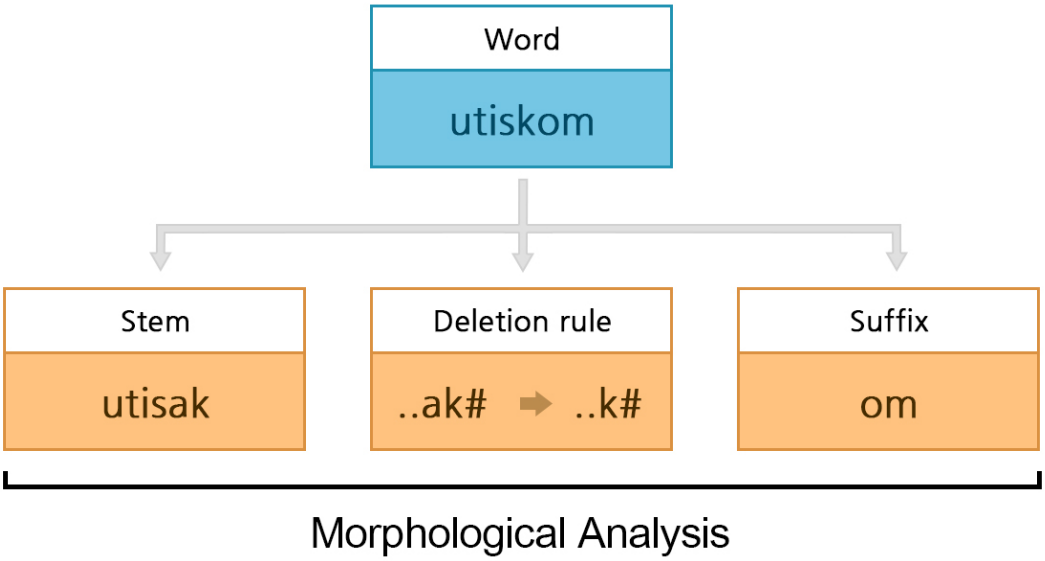

Figure 1.2: Morphological analysis for Serbian word *utiskom* (meaning *appearance*). It would be decomposed into the stem *utisak*, a deletion rule which removes the final vowel, and the instrumental case suffix *om*.

3) percentage of suffix-free words, and words that use phonological deletions. All of them can be plausibly generalized across a wide range of languages. This function depends crucially on the structure of the proposed label y, not simply the input x.

Since the morphological systems of related languages tend to be similar in structure, we proposed a nearest neighbor approach to this problem. However, because our feature function takes into account both the input and label, it is not possible to employ the traditional nearest neighbor procedure, so instead we developed a technique called "structured" nearest neighbor, which seeks to find the morphological analysis for the target language which lies as close as possible in the universal feature space to a training language.

To be specific, the training algorithm for our structured nearest neighbor method is similar to standard nearest neighbor: take each input-label pair from the training data $(x_1, y_1), \ldots, (x_n, y_n)$,

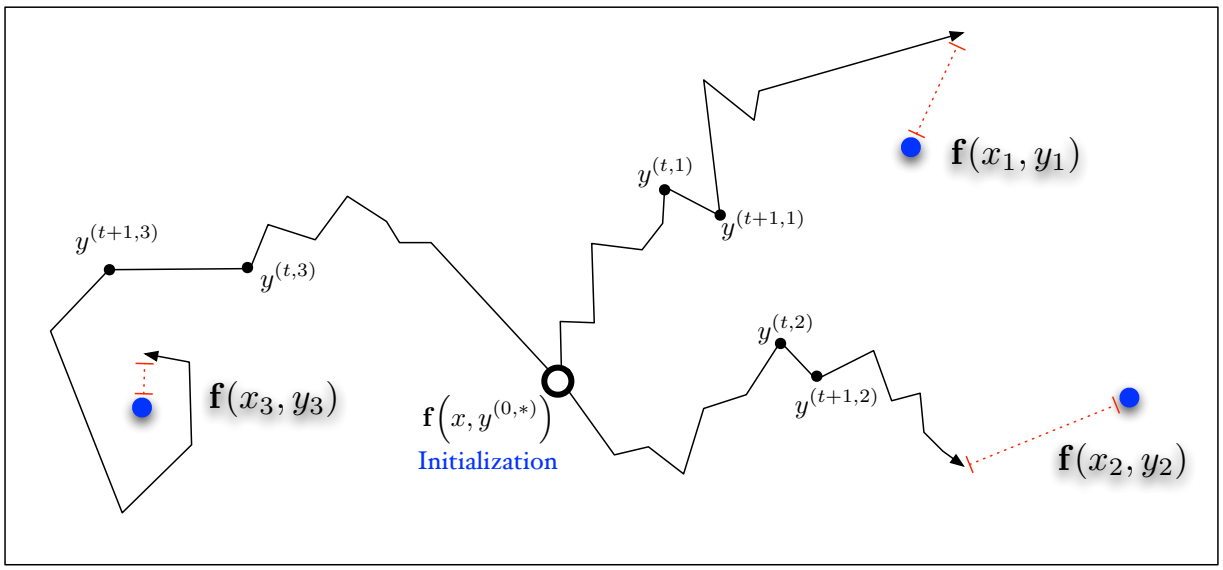

Figure 1.3: **Structured Nearest Neighbor Search:** The inference procedure for unlabeled test language x, when trained with three labeled languages, $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Our search procedure iteratively attempts to find labels for x which are as close as possible in feature space to each of the training languages. After convergence, the label which is closest in distance to a training language is predicted, in this case being the label near training language (x_3, y_3).

map them into the universal feature space, $\mathbf{f}(x_\ell, y_\ell) \in \mathbb{R}^d$, and memorize the resulting vectors. At test time, when a target language is given, our method performs parallel greedy searches in an attempt to drive the test language feature vector towards the feature vector of each of the training languages. Each search procedure starts with a random morphological analysis, and iteratively performs greedy updates to bring the target language's feature representation closer to the features of the particular training language.

See Figure 1.3 for a graphical illustration of this procedure. After iterating this procedure to convergence, we are left with a set of analyses $\left\{y^{(\ell)}\right\}_\ell$, each of which yields (approximate) minimal distances to a particular training language ℓ:

$$y^{(\ell)} \approx \operatorname*{argmin}_{y \in \mathcal{Y}} \parallel \mathbf{f}(x, y) - \mathbf{f}(x_\ell, y_\ell) \parallel .$$

We finally select from amongst these analyses and make our prediction:

$$\ell^* = \underset{\ell}{\operatorname{argmin}} \parallel \mathbf{f}(x, y^{(\ell)}) - \mathbf{f}(x_\ell, y_\ell) \parallel$$
$$y^* = y^{(\ell^*)}$$

Empirical findings validate this approach: On a set of eight different languages, our method yielded substantial accuracy gains over the state-of-the-art monolingual baseline, yielding relative error reduction to the supervised upperbound of 42%. Also, we found out that as the number of training languages increases, test performance continuously improves. It reflects that if more languages will be added to the training set, we might be able to construct more robust an universal model.

1.1.2 Grapheme-to-Phoneme Analysis of Latin Alphabets

In the previous task, we have developed the idea that supervised knowledge of some number of languages can help guide the unsupervised induction of linguistic structures in the absence of parallel texts. Now we extend this idea to a task of grapheme-to-phoneme analysis of Latin alphabets as a test case for larger scale cross-lingual learning by using information from dozens of other languages. The goal of this task is to automatically select a subset of phonemes for each language's graphemes. For example, depending on the language, the Latin alphabet grapheme "c" can represent any of the following phonemes:

$$\{/\text{k}/, /\text{c}/, /\text{s}/, /|/, /\text{dʒ}/, /\text{tʃ}/, /\text{ts}/\}$$

Among these phonemes, we would like to predict the set $\{/\text{s}/, /\text{k}/\}$ for the grapheme "c" in English, using only raw texts as our input. In other words, we consider this as a binary prediction problem over each grapheme-phoneme pair employed by any Latin alphabet across a wide variety of languages. Taken together, these predictions yield the grapheme-phoneme mapping for a particular language using a form of the Latin Alphabet.

In this task, the input $x \in \mathcal{X}$ consists of vocabulary list of all words in a monolingual corpus as before, but this time covering a much wider range of languages, all of which use some form

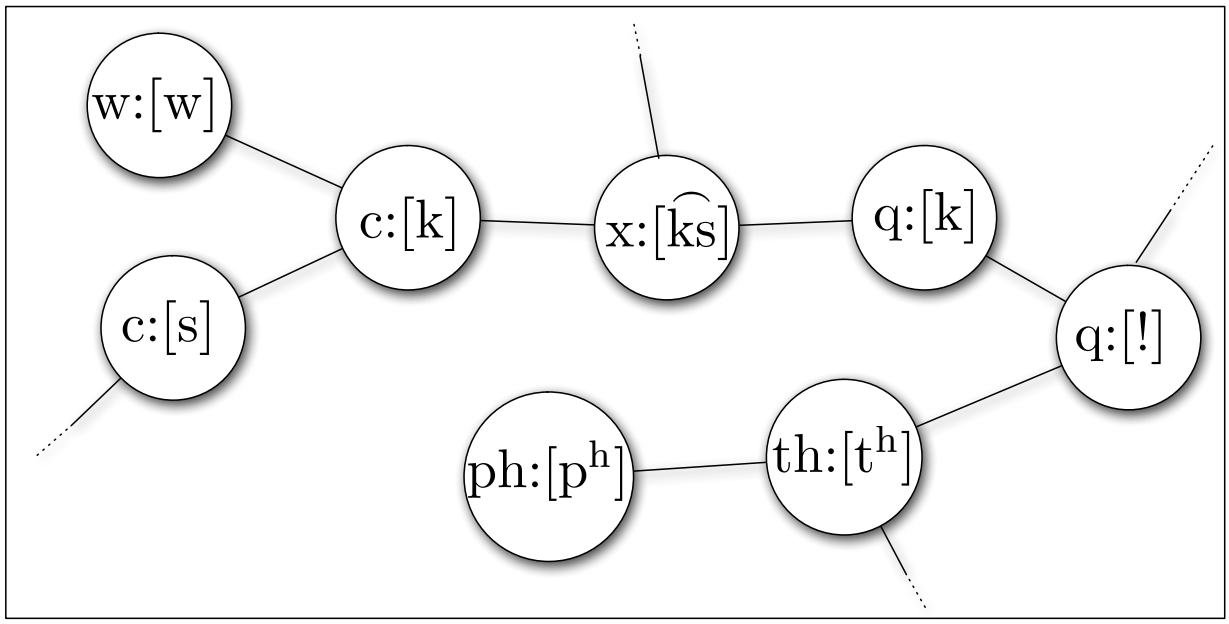

Figure 1.4: **A snippet of our undirected graphical model.** The binary-valued nodes represent whether a particular grapheme-phoneme pair is allowed by the language. Sparse edges are automatically induced to allow joint training and prediction over related inventory decisions.

of the Latin alphabet. The label $y \in \mathcal{Y}$ maps each individual grapheme to a set of phonemic values used in the language. The goal is to learn the phonotactic regularities inherent in all the different ways that the Latin alphabet has been used historically. For example, if the grapheme "c" appears before the non-front vowels such as /a/ many times, its sound might be /k/, voiceless velar stop. For a universal feature function $\mathbf{f}$ to capture the phonotactic regularities, we designed the following set of features: 1) language family and region, 2) counts of all character unigrams and bigrams surrounding each grapheme, and 3) approximate articulatory features of the surrounding phonemes.

This method was realized in the Markov Random Fields shown in Figure 1.4, allowing us to perform joint inference over related grapheme-to-phoneme pair. To be specific, our model has a node for each grapheme-phoneme pair in each language. A binary label on the node indicates whether the language allow that particular mapping or not. For each node and edge, we have a set of features and a corresponding set of parameters. The node features are simply the features

discussed above, and the edge features are combinations of the two node features. To induce the edge-structure of our graph, we used the PC algorithm [97].[1] The induced graph structure which we used is shown in Figure 1.5.

Once the graph structure has been induced, we learn weights over our features which optimally related the input features of the training languages to observed labels. In other words, the weights are estimated by maximizing the label-likelihood over training languages conditioned on features. Numerical optimization of label-likelihood can simply be performed using L-BFGS with its gradient. At test time, the learned weights are used to predict the label of the target language with highest probability.

Empirical findings validate this approach: On a set of 107 different languages, our model correctly predicted grapheme-phoneme pairs with over 88% F1-measure. It reflects that our model automatically learns how to map plausible phonemic interpretations using induced phonotactic regularities.

1.1.3 Interlude: Tracing the History of the Roman Alphabet

As a short interlude, we want to show that phonotactic regularities can be viewed as historical knowledge of alphabet propagation. To explain this fact, we briefly introduced a data-set and a method for the automatic reconstruction of the history of the Roman alphabet as it has spread across time and space. Our data-set consists of grapheme-phone mappings for close to 300 languages, collated with meta-data including language family and geographical location. Using only two simple distance measures, our clustering results are remarkably consistent with what is known from historical knowledge. The first one measures the degree to which the language represent their common phonemes using the same graphemes. The second one measures the degree of overlap between non-standard graphemes employed by the two languages, including accented letters, digraphs, and non-Latin characters. By using the unsupervised technique of hierarchical agglomerative clustering with these two measures, we induced tree structure with the various clusters shown in Figure 1.6.

[1] PC stands for Peter and Clark, the first names of the two inventors of PC algorithm.

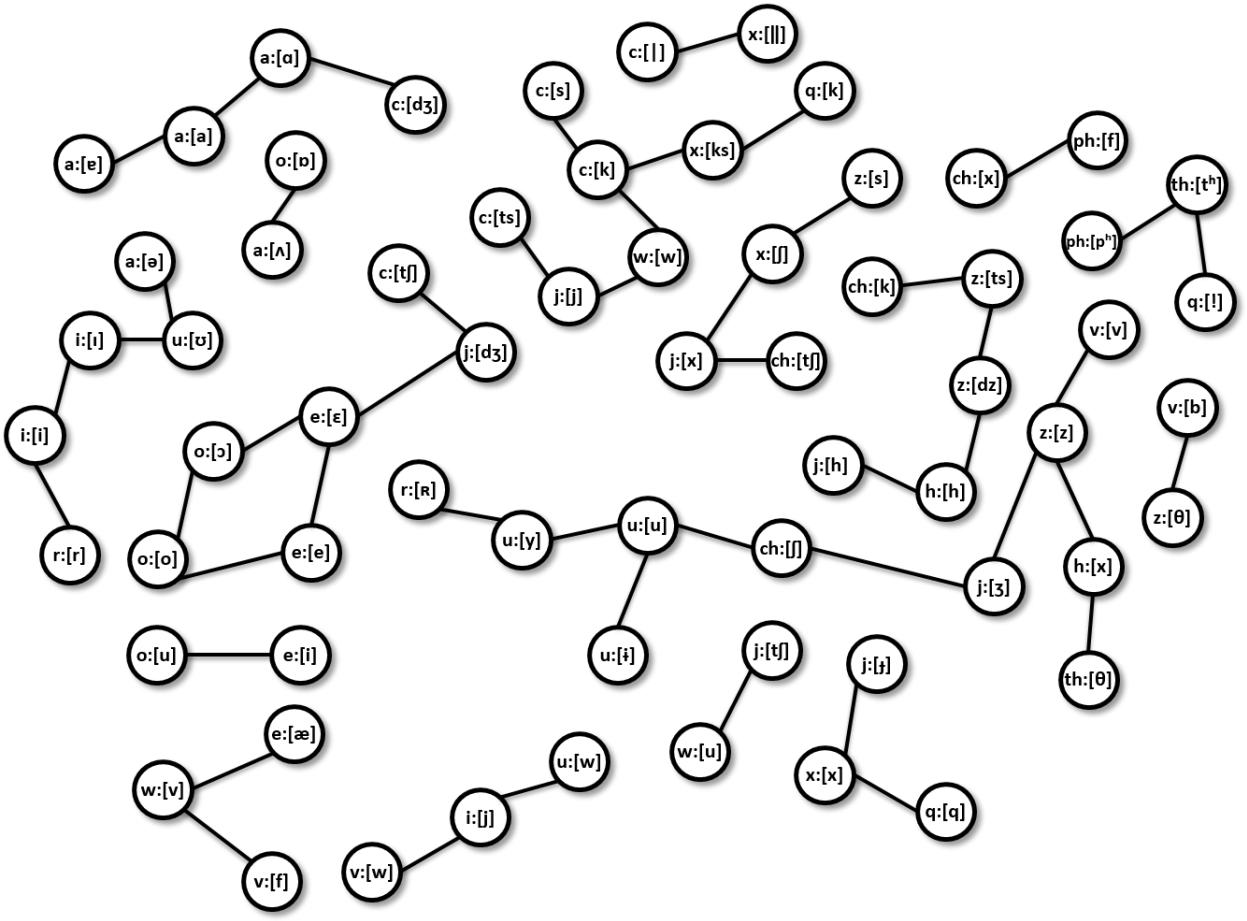

Figure 1.5: Edge structure induced by the PC algorithm. The graph has 75 nodes and about 50 edges between these nodes.

By examining the induced clusters in this figure, we were able to assess qualitatively how well the resulting clusters match up with what we know of the history of the alphabet. Interestingly, the height of the clusters seems to correspond quite nicely with the time at which the languages adopted the Latin alphabet: The highest clusters are dominated by European languages. A few steps down, we find a cluster consisting of Turkic and other central Asian languages. Near the bottom of the tree, we see clusters of native American and African languages, and at the very bottom of the tree, we find a cluster consisting of southeast Asian and Pacific island languages.

So, we argue that corsslingual grapheme-phone ambiguity can serve as a means for the automatic analysis of the history of the alphabet. Our key premise is that as an alphabet gets passed

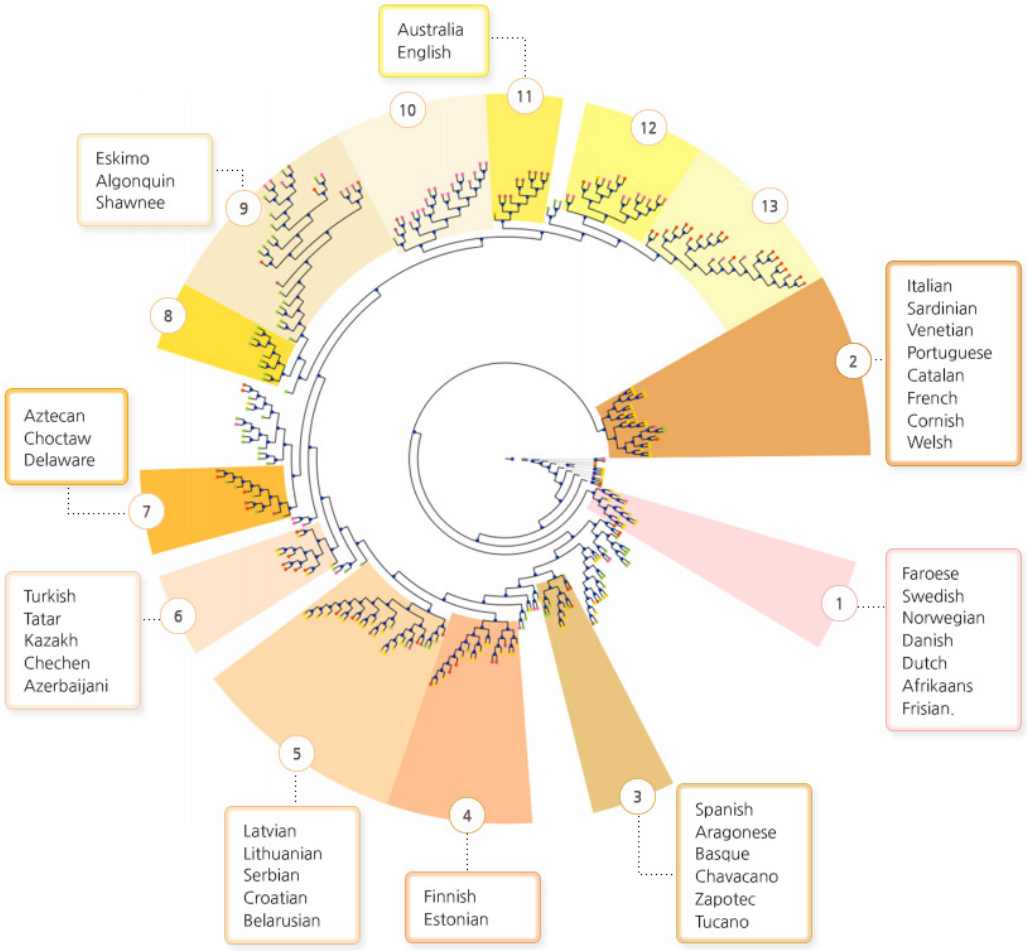

Figure 1.6: Hierarchical clustering of languages, based on alphabetic variations. After examining and qualitatively assessing clusters, the coherent clusters are highlighted and labeled with numbers.

from one language to another, the idiosyncratic writing features of the source language get passed on to the target language alphabets, allowing us retrace the history of the writing system over time and place.

1.1.4 Phonetic Decipherment

In the previous work, we have developed a model which related patterns of symbols in texts to plausible phonetic interpretations with high accuracy by harnessing supervised knowledge over a hundred languages. However, we assumed that the target language was written in a known (Latin) alphabet, greatly reducing the difficulty of the prediction task. In this work, we consider the language decipherment task, where no knowledge of any relationship between the writing system of the target language and known languages exists. Specifically, we focus on one aspect of decipherment tasks: automatically identifying basic phonetic properties of letters in an unknown alphabetic writing system.

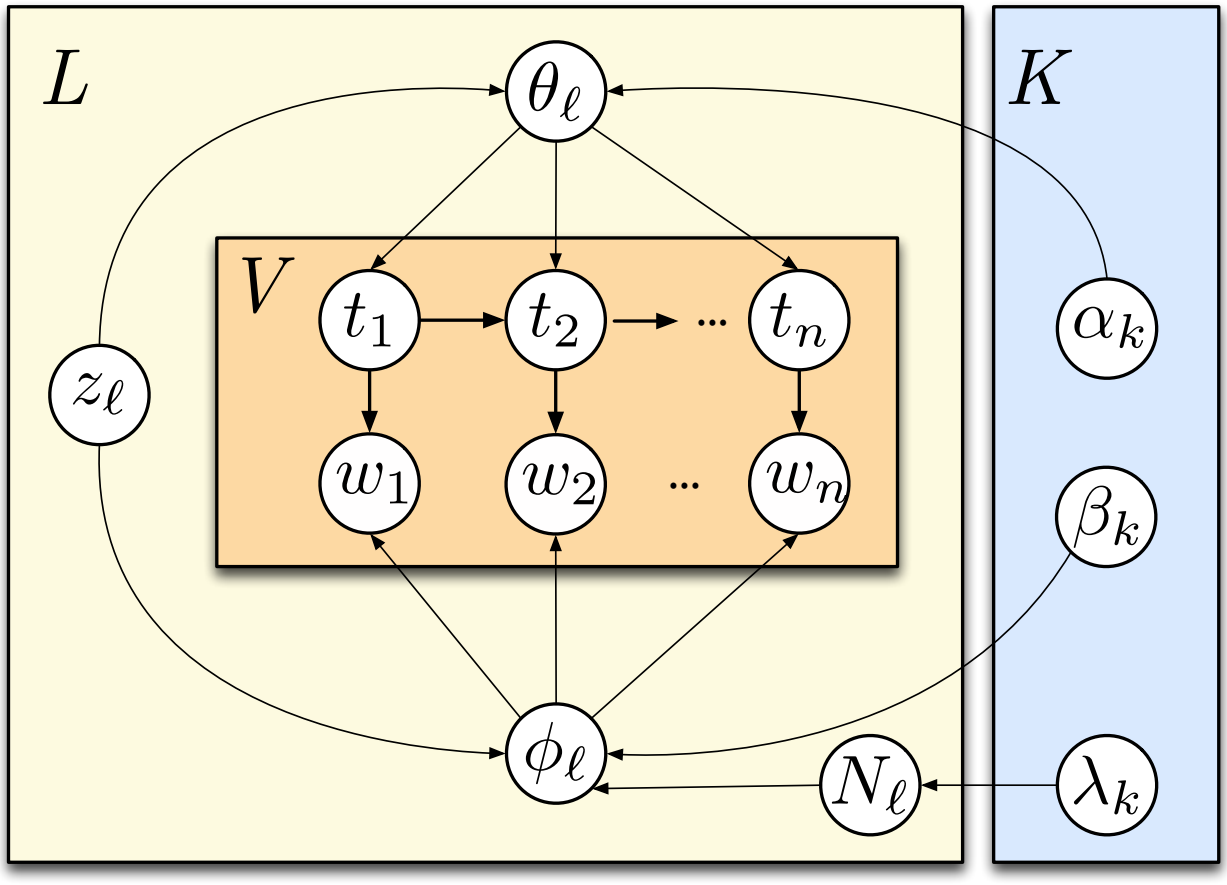

Figure 1.7: Graphical representation of our model. We have K language clusters, L languages, and V words in each language.

In this task, the input $x \in \mathcal{X}$ is the vocabulary of all words in a monolingual corpus as before, but this time the words are collected from more than 500 languages with many different writing systems. The label $y \in \mathcal{Y}$ maps an individual character to its coarse phonetic category. For example, a label indicates if a character is vowel or consonant and if it is non-nasal consonant or nasal. The goal of our universal feature function $\mathbf{f}$ is to capture transitional regularities of phonemes. For example, we want to capture consonants, or vowels tends to be followed by the opposite categories. We also wish to represent the number of character types in each category, as well as the entropy of distribution over phonemes in each category.

To this end, we formulated our Bayesian hierarchical model over the observed vocabularies of hundreds of languages, which is presented in Figure 1.7. The inner box in the figure represents a set of observation sequences from an HMM. Each sequence represents a single word in the vocabulary through a set of character observations and hidden phoneme category states.

The HMM parameters are specific to a particular language, but share (unobserved) Dirichlet hyperparameters across a cluster of languages to model the shared transitional regularities. To learn the shared hyperparameters, clustering, and tags for the target language, we perform inference using the Gibbs sampling from the posterior distributions.

Empirical findings validate this approach: On a set of 503 different languages, we achieved average accuracy in unsupervised consonant/vowel prediction task of 99%. We further showed that our methodology can be used to predict more fine-grained phonetic distinctions. On a three-way classification task between vowels, nasals, and non-nasal consonants, our model yields unsupervised accuracy of 89% across the same set of languages. It mirrors that our universal probabilistic model can successfully decode new languages by using knowledge of the phonetic regularities encoded in known language vocabularies.

1.1.5 Part-of-speech tagging for Low-resource Languages

Up to this point, we have examined the application of cross-lingual supervised learning to several longstanding NLP problems such as morphological analysis, grapheme-to-phoneme analysis, and phonetic decipherment. These tasks can be viewed as word-level structural analyses, where

the input $x \in \mathcal{X}$ is a list of all words. However, some of NLP tasks can benefit by analyzing sentences and paragraphs. Therefore, we apply our concepts to a sentence-level structural analysis of the part-of-speech (POS) prediction . In this task, given a sentence such as "Dogs chase cats", we automatically predict the sequence of POS tags such as `NOUN VERB NOUN`. Note that the words "dogs" and "chase" can be both `NOUN` and `VERB` depending on their contexts.

As before we do not assume the existence of a training data for the target language, or any prior knowledge of it. However, we do not directly use annotated resources for different languages. Instead, we base our methods entirely on the existence of parallel data between the target language and a set of resource-rich languages. For this new setting, we first collected electronic Bible translations for 650 languages. Among these languages, we defined "resource-rich" languages to be ones with reasonably accurate POS taggers. The resource-rich languages are Bulgarian, Czech, Danish, German, English, French, Spanish, Italian, Dutch, and Portuguese. We POS-tagged the sentences of these languages, word-aligned sentences of these languages, and also word-aligned resource-rich sentences to the sentences of the remaining target languages.

With the above setting, we apply an instance-learning approach using latent distributional features to the task of POS tagging. While applying the method, we first select words tagged with high confidence scores to keep high-quality tags. To induce these latent features, we design a new method via Canonical Correlation Analysis (CCA) to transfer the tags of known words to unknown words. This method looks at each word with the following three fundamental views: (1) the token view (the word's context), (2) the type view (the word identity), and (3) the transferred tags in the neighboring words. We perform a two-step CCA to induce latent continuous vector representations for each view that maximizes their correlations to one another. The first step produces word representations using word contexts and the second step creates tag-incorporated word representations using learned word representations and transferred tags.

The experiments show this approach works: By taking into account the vastly multilingual nature of our parallel data, our method both individually and jointly outperforms the state-of-the-art baselines to achieve average tagging accuracy of 85% across a test-suite of ten languages in an entirely different genres.

1.2 Optimal Data Selection

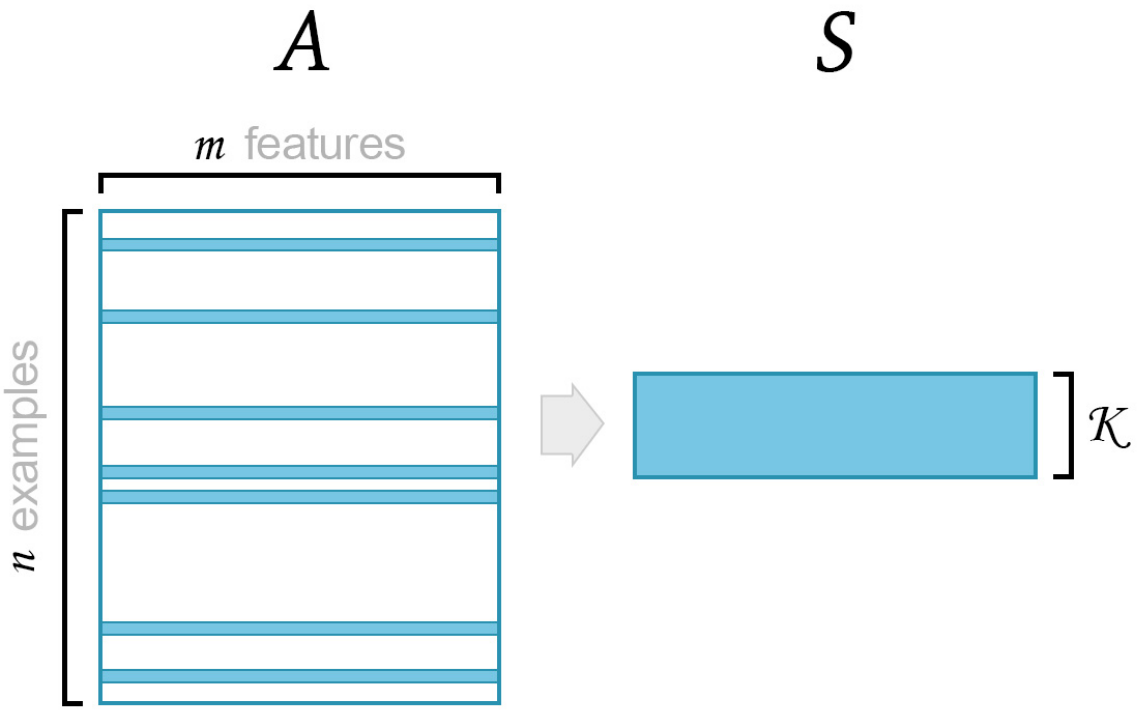

Figure 1.8: Given a large set $\mathcal{A}$ of n unlabeled examples of m features, we must select a subset $\mathcal{S} \subset \mathcal{A}$ of size $k \ll n$ to label. Our goal is to select such a subset which, when labeled, will yield a high performance supervised model over the entire data set $\mathcal{A}$

So far we have assumed that no annotated resources are available for the target language. From this section, we assume that we do have ability to create labeled data to train a model, but with limited time and budget. Under this limitation, the amount of data to be annotated might be small, especially in the prototyping stage, when the exact specification of the prediction task may still be in flux, and rapid prototypes are desired. When multilingual and multi-domain models are desired, the problem of annotation scarcity becomes even more severe.

In this section, we propose the idea that when the examples to be labeled are chosen carefully and intelligently, the amount of annotation required to achieve performance goals can be drastically reduced. To achieve this goal, we introduce the novel task of unsupervised optimal data set selection, illustrated in Figure 1.8.

Unfortunately, we are not aware of previous work proposing optimal data set selection as a general research problem. Of course, active learning strategies can be employed for data selection by starting with a small random seed of examples and incrementally adding small batches. This unfortunately can lead to data-sets that are biased to work well for one particular class of models and task, but may otherwise perform worse than a random set of examples [90, Section 6.6]. Furthermore the active learning set-up can be prohibitively tedious and slow. To illustrate, Dwyer and Kondrak [34] showed that the underlying prediction model is trained 1,900 times in total considering the number of iteration, the number of sample for bootstrapping and the number of learners. In contrast, our selection methods are fast, can select any number of data points in a single step, and not tied to a particular prediction task or model. Furthermore, these methods can be combined with active learning in selecting the initial seed set.

Finally, we note that optimal data selection has similar property with methods which have been applied to the problem of *unsupervised feature selection* [100; 71; 103; 108; 12]. These methods are related to dimensionality reduction techniques such as Principal Components Analysis (PCA), but instead of truncating features in the eigenbasis representation (where each feature is a linear combination of all the original features), the goal is to remove dimensions in the standard basis, leading to a compact set of interpretable features. Techniques we used in this work employ similar linear algebraic methods but operate over datapoints rather than features.

Now, we will describe how we applied our proposed method to a suite of four natural language understanding tasks: Pronunciation Dictionary Induction, Part-of-speech Prediction, Named Entity Recognition and Semantic Tagging.

1.2.1 Pronunciation Dictionary Induction

In this task, we are given an out-of-vocabulary word, with the goal of predicting a sequence of phonemes corresponding to its pronunciation. For example, given *calcium*, we should predict /kælsiəm/. The task is related with grapheme-to-phoneme prediction discussed in the previous section. Since here we must predict actual pronunciations of words beyond phonemic value of characters, this irregularity makes our task even harder.

1.2.2 Part-of-speech Prediction

In this task, our goal is to predict the parts-of-speech of a sequence of words forming a sentence. For example, given the sentence

Can you show me a map of Madison?

The goal would be to determine that the word *Can* is a `verb`, rather than a `noun`, and the word *map* is a `noun`, rather than a `verb`.

1.2.3 Named Entity Recognition

In this task, the goal is to recognize mentions of entities in text and speech data. For example, in the sentence

Show me a list of movies directed by Steven Spielberg.

our goal is to determine that the words *Steven Spielberg* refer to the famous film director of that name.

1.2.4 Semantic Tagging

In our final task, we consider the important problem of semantic tagging of sentences, sometimes known as slot-filling. This task is an essential component of natural language understanding systems, allowing the machine to understand the desires of the speaker. For example, in the sentence

Give me a list of 5 best selling books printed by Pearson in 2014.

the goal would be ascertain that the speaker desires a list of products for sale with the following attributes:

- **type:** *book*
- **publisher:** *Pearson*
- **popularity:** $\geq$ *Rank 5*
- **year:** *2014*

For these tasks, our first proposed method, based on the rank-revealing QR matrix factorization, selects a subset of data which span the entire data-space effectively. In this method, k rows are selected from an unlabeled data matrix A, whose rows correspond to words and whose columns correspond to features (character or word 4-grams). In other words, we select data points which form the best possible basis for the entire data space. The intuition of this idea is that the labels of these data would be the most informative in terms of how all data relate to their labels.

For our second method, we developed the concept of feature coverage which we optimize with a greedy algorithm. In the method of feature coverage, we assume that the benefit of seeing a feature in a selected data point bears some positive relationship to the frequency of feature in the unlabeled pool. However, we further assume that the lion's share of benefit accrues the first few times that we label a data point with that feature, with the marginal utility quickly tapering off as more such examples have been labeled. We formalize this notion and provide an exact greedy algorithm for selecting the k data points with maximal feature coverage.

Empirical findings validate this approach: On a set of 20 different languages and 6 domains, our selection methods are effective at yielding a small, but optimal set of labeled examples. In all scenarios when fed into a state-of-the-art supervised model for individual task, our data set selection methods lead to significant increases in performance over randomly labeled examples. Average reductions in error range from 20% to 31% across the various tasks.

1.3 Conclusions

In this first chapter, we explained the importance of this line of research by showing that the number of languages needed to cover the most of world population is very large.

For the first line of our argument, we assumed that no annotated resources are available for the target language. For this scenario, we proposed a new framework of cross-lingual supervised learning. The key idea is that through a joint analysis of a broad array of languages, languages with annotated resources can be used as training data for resource poor languages. We showed the vision of cross-lingual learning across a broad range of classical NLP problems, including morphological analysis, grapheme-to-phoneme analysis of Latin alphabet, language decipherment and part-of-speech prediction. In all cases, the methods yielded substantial performance gains without any human annotation for the target language.

In the second line of our argument, we assumed that there is only limited budget or time for supervised annotation. For this scenario, we proposed two techniques for identifying an optimal set of examples to be labeled, in order to produce a high-performance supervised model. When we applied these techniques to the task of pronunciation dictionary induction, the selection methods proved effective at yielding a small, but optimal set of labeled examples. Existing state-of-art supervised models trained with these examples yielded substantial performance gains over randomly selected examples. In the next chapters, we will go over each of our published results in great depth and technical details.

Chapter 2

Morphological Analysis

In this chapter, we explore the application of cross-lingual supervised learning to morphological analysis in an unsupervised setting. For this task, the assumption is that, we are given written texts in a test language without any human annotation and written texts in other languages with their own annotation. The goal is to automatically decompose a given word into its stem, an optional phonological deletion rule, and an optional suffix. For example, the Serbian word *utiskom* (meaning *appearance*, would be decomposed into the stem *utisak*, a deletion rule which removes the final vowel, and the instrumental case suffix *om*. This work was originally published in [57].

This chapter is organized as follows: Section 2.1 gives a broad introduction to universal morphology analysis. We argue that languages for which we *have* gold-standard morphological analyses can be used as training data for languages *lacking* such resources. Section 2.2 compares our approach to the state-of-the art unsupervised morphological analyzers. Section 2.3 describes our approach and search procedure in great detail. Section 2.4 outlines our experiments on eight European language corpora and reports our results. Section 2.5 completes the chapter with some concluding remarks.

2.1 Introduction

Most previous natural language processing research have focused on the development of text processing tools and techniques for English [6], which is a morphologically simple language. In recent years, there has been increasing need for natural language processing in the wide variety of other languages in use throughout the world. Most of these languages pose major challenges due

to (i) lack of available linguistic resources, such as annotated corpora, and (ii) the fact that they exhibit linguistic structures not found in English, and are thus not immediately susceptible to many traditional NLP techniques.

Consider the example of nominal part-of-speech analysis. The Penn Treebank defines four English language noun tags [73], reflecting the fact that English nominal morphology and orthography make only two category distinctions within nouns: singular vs. plural, and common vs. proper. It is often easy to treat these as four completely distinct tags, and to treat the words themselves as completely distinct from one another, sharing no internal morphological structure. In contrast, a comparable tagset for Hungarian includes 154 distinct noun tags [37], reflecting Hungarian's rich inflectional nominal morphology, which explicitly marks nouns with person, case, and number. In this case, treating words as distinct atomic units would lead to massive word sparsity problems, then motivating the need for word-internal morphological analysis.

Because of the absence of annotated resources for most morphologically rich languages, previous research has mostly focused on unsupervised methods for morphological analysis, with relying heavily on appropriate inductive biases. However, inductive biases and declarative knowledge are notoriously difficult to encode in well-founded models, and putting aside this practical matter. For example, the hidden Markov model for part-of-speech induction can encode a bias toward transitional regularity of the hidden states, but the resulting model becomes overly complicated to learn. In addition, a universally correct inductive bias, if there is one, is unlikely to be discovered by *a priori* reasoning alone.

In this chapter, we argue that languages for which we *have* gold-standard morphological analyses can be used as effective and active guides for languages *lacking* such resources. In other words, instead of treating each language's morphological analysis as a *de novo* induction problem to be solved with a one-size-fits-all hand-coded bias, we instead empirically *learn* from our available labeled languages what linguistically plausible morphological analyses looks like, and guide our analysis in that direction.

More formally, we recast the unsupervised morphological induction as a new kind of supervised structured prediction problem, where each annotated language serves as a single training

example, in which the noun lexicon serves as input x, and the analysis of the nouns into stems and suffixes serves as a complex structured label y.

The first step is to define a universal morphological feature space, into which each language and its morphological analysis can be mapped. We opt for a simple and intuitive universal feature space, measuring the sizes of the stem and suffix lexicons, the entropy of the distributions over such stems and suffixes, and the fraction of word forms which appear without any inflection.

Since languages tend to cluster into well defined morphological groups, we cast our learning and prediction problem in the nearest neighbor framework. However, in contrast to its typical use in classification problems, where one can simply pick the label of the nearest training example, we are faced with a structured prediction problem, where locations in feature space depend jointly on the input-label pair (x, y). Finding a nearest neighbor thus consists of *searching* over the space of morphological analyses, until a point in the universal feature space is reached which lies closest to one of the labeled languages.

We applied our model to eight languages with inflectional nominal morphology, ranging in complexity from very simple (English) to very complex (Hungarian). In all eight cases, our approach yields substantial improvements over a comparable monolingual baseline [44], which uses the Minimum Description Length principle (MDL) as its inductive bias.

2.2 Related Work

Unsupervised morphology induction remains an active area of research [88; 44; 1; 22; 27; 23; 83]. Most existing algorithms derive morpheme lexicons by identifying recurring patterns in words. The goal lies in optimizing the compactness of the data representation by finding a small lexicon of highly frequent strings, resulting in a *minimum description length* (MDL) lexicon and corpus [43; 44]. Later work cast this idea in a probabilistic framework in which the MDL solution is equivalent to a MAP estimate in a suitable Bayesian model [22]. All these approaches use a task-specific greedy search, resulting in a locally optimal segmentation.

2.3 Model: Structured Nearest Neighbor

Unlike previous approaches, we re-formulate unsupervised morphological induction as a *supervised* learning task, where each annotated language serves as a single training example for our language-independent model. Each such example consists of an input-label pair (x, y), both of which contain complex internal structure: The input $x \in \mathcal{X}$ consists of a vocabulary list of all words observed in a particular monolingual corpus, and the label $y \in \mathcal{Y}$ consists of the correct morphological analysis of all the vocabulary items in x. Technically, the label space of each input, $\mathcal{Y}$, should be thought of as a function of the input x. We suppress this dependence for notational clarity. Because our goal is to generalize across languages, we first have to define a feature function which maps each (x, y) pair to a universal feature space: $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$.

For each unlabeled input language x, our goal is to predict a complete morphological analysis $y \in \mathcal{Y}$ which maximizes a scoring function on the feature space, $score : \mathbb{R}^d \rightarrow \mathbb{R}$. This scoring function is trained using the n labeled-language examples: $(x, y)_1, \ldots, (x, y)_n$, and the resulting prediction rule for unlabeled input x is given by:

$$y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, score\big(\mathbf{f}(x, y)\big)$$

Languages can be typologically categorized by the type and richness of their morphology. On the assumption that for each test language, at least one typologically similar language will be present in the training set, we employ a *nearest neighbor* scoring function. In the standard nearest neighbor classification setting, one simply predicts the label of the closest training example in the input space[1]. In our structured prediction setting, the mapping to the universal feature space depends crucially on the structure of the proposed label y, not simply the input x. We thus generalize nearest-neighbor prediction to the structured scenario and propose the following prediction rule:

$$y^* = \underset{y \in \mathcal{Y}}{\operatorname{argmin}} \min_{\ell} \parallel \mathbf{f}(x, y) - \mathbf{f}(x_\ell, y_\ell) \parallel, \tag{2.1}$$

[1] More generally the majority label of the k-nearest neighbors

where the index ℓ ranges over the training languages. In other words, we predict the morphological analysis y for our test language which places it as close as possible in the universal feature space to one of the training languages ℓ.

2.3.1 Morphological Analysis

In this work, we focus on nominal inflectional suffix morphology. Consider the word *utiskom* in Serbian, meaning *impression* with the instrumental case marking. A correct analysis of this word would divide it into a stem (*utisak* = *impression*), a suffix (*-om* = instrumental case), and a phonological deletion rule on the stem's penultimate vowel ($..ak\# \rightarrow ..k\#$).

More generally, as we define it, a morphological analysis of a word type w consists of (i) a stem t, (ii), a suffix f, and (iii) a deletion rule d. Either or both of the suffix and deletion rule can be $NULL$. We allow three types of deletion rules on stems: deletion of final vowels ($..V\# \rightarrow ..\#$), deletion of penultimate vowels ($..VC\# \rightarrow ..C\#$), and removals and additions of final accent marks (e.g. $..\tilde{a}\# \rightarrow ..a\#$). We require that stems must be at least three characters long and that suffixes must be no more than four. And, of course, we require that after (1) applying deletion rule d to stem t, and (2) adding suffix f to the result, we obtain word w.

2.3.2 Universal Feature Space

We employ a fairly simple and minimal set of features, all of which could plausibly generalize across a wide range of languages. Consider the set of stems T, suffixes F, and deletion rules D, induced by the morphological analyses y of the words x. Our first three features simply count the sizes of these three sets.

These counting features consider only the raw number of unique morphemes (and phonological rules) being used, but not their individual frequency or distribution. Our next set of features considers the empirical *entropy* of these occurrences as distributed across the lexicon of words x by analysis y. For example, if the (x, y) pair consists of the analyzed words {*kiss, kiss-es, hug*}, then the empirical distributions over stems, suffixes, and deletion rules would be:

- $P(t = kiss) = 2/3$

- $P(t = hug) = 1/3$
- $P(f = NULL) = 2/3$
- $P(f = -es) = 1/3$
- $P(d = NULL) = 1$

The three entropy features are defined as the Shannon entropies of these stem, suffix, and deletion rule probabilities: $H(t), H(f), H(d)$. Note that here and throughout the section, we operate over word types, ignoring their corpus frequencies.

Finally, we consider two simple *percentage* features: the percentage of words in x which according to y are left unsegmented (i.e. have the null suffix, 2/3 in the example above), and the percentage of segmented words which employ a deletion rule (0 in the example above). Thus, in total, our model employs 8 universal morphological features. All features are scaled to the unit interval and are assumed to have equal weight.

2.3.3 Search Algorithm

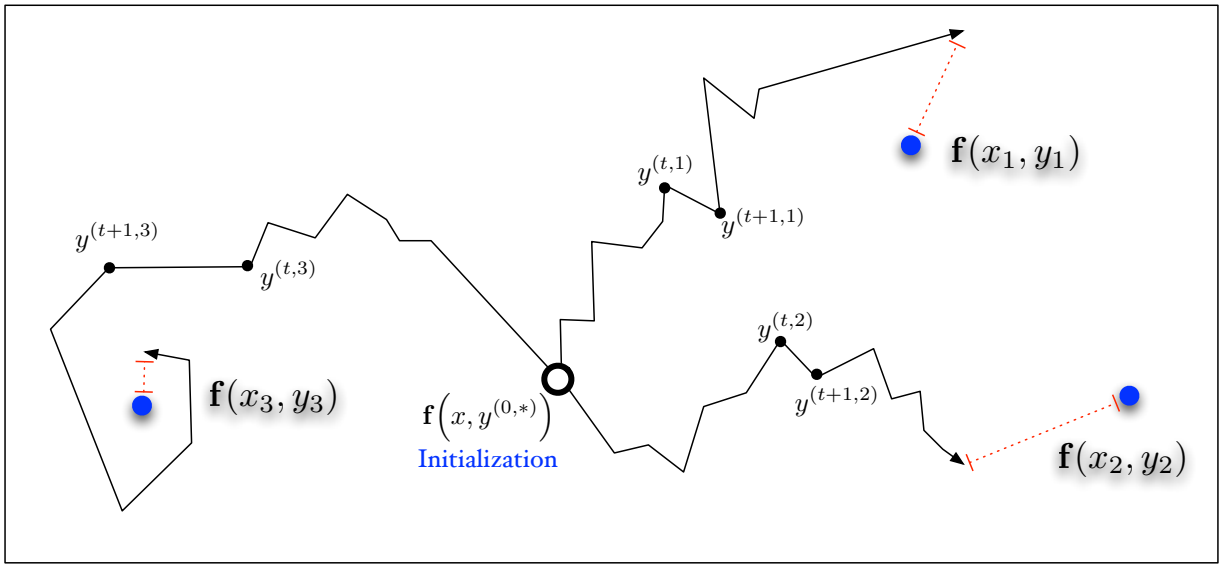

Figure 2.1: Illustration of Structured Nearest Neighbor Search

The main algorithmic challenge of our model is to efficiently compute the best morphological analysis y for each language-specific word set x, according to Equation 2.1. Exhaustive search through the set of all possible morphological analyses is impossible, as the number of such analyses is exponential in the size of the vocabulary. Instead, we develop a greedy search algorithm in the following fashion. The search procedure is visually depicted in Figure 2.1, where the inference procedure for unlabeled test language x, when trained with three labeled languages, $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Our search procedure iteratively attempts to find labels for x which are as close as possible in feature space to each of the training languages. After convergence, the label which is closest in distance to a training language is predicted, in this case being the label near training language (x_3, y_3).

Formally, at each time-step t, we maintain a set of frontier analyses $\left\{y^{(t,\ell)}\right\}_\ell$, where ℓ ranges over the training languages. The goal is to iteratively modify each of these frontier analyses $y^{(t,\ell)} \rightarrow y^{(t+1,\ell)}$ so that the location of the test language in universal feature space — $\mathbf{f}\left(x, y^{(t+1,\ell)}\right)$ — is as close as possible to the location of the training language ℓ: $\mathbf{f}\left(x_\ell, y_\ell\right)$.

After iterating this procedure to convergence, we are left with a set of analyses $\left\{y^{(\ell)}\right\}_\ell$, each of which approximates the analyses which yield minimal distances to a particular training language:

$$y^{(\ell)} \approx \operatorname*{argmin}_{y \in \mathcal{Y}} \parallel \mathbf{f}(x, y) - \mathbf{f}(x_\ell, y_\ell) \parallel .$$

We finally select from amongst these analyses and make our prediction:

$$\ell^* = \operatorname*{argmin}_{\ell} \parallel \mathbf{f}(x, y^{(\ell)}) - \mathbf{f}(x_\ell, y_\ell) \parallel$$
$$y^* = y^{(\ell^*)}$$

The main outline of our search algorithm is based on the MDL-based greedy search heuristic developed and studied by [44]. At a high level, this search procedure alternates between individual analyses of words (keeping the set of stems and suffixes fixed), aggregate discoveries of new stems (keeping the suffixes fixed), and aggregate discoveries of new suffixes (keeping stems fixed). As input, we consider the test words x in our new language, and we run the search in parallel for each training language (x_ℓ, y_ℓ). For each such test-train language pair, the search consists of the following stages:

2.3.3.1 Stage 0: Initialization

We initially analyze each word $w \in x$ according to peaks in *successor frequency*. The *successor frequency* of a string prefix s is defined as the number of unique characters that occur immediately after s in the vocabulary. If w's n-character prefix $w_{:n}$ has successor frequency > 1 and the surrounding prefixes, $w_{:n-1}$ and $w_{:n+1}$ both have successor frequency $= 1$, then we analyze w as a stem-suffix pair: $(w_{:n}, w_{n+1:})$. Note that with the restriction that at this stage, we only allow suffixes up to length 5, and stems of at least length 3. Otherwise, we initialize w as an unsuffixed stem. As this procedure tends to produce an overly large set of suffixes F, we further prune F down to the number of suffixes found in the training language, retaining those which appear with the largest number of stems. This initialization stage is carried out once, and afterwards the following three stages are repeated until convergence.

2.3.3.2 Stage 1: Reanalyze each word

In this stage, we reanalyze each word (in random order). We use the set of stems T and suffixes F obtained from the previous stage, and do not permit the addition of any new items to these lists. Instead, we focus on obtaining better analyses of each word, while also building up a set of phonological deletion rules D. For each word $w \in x$, we consider all possible segmentations of w into a stem-suffix pair (t, f), for which $f \in F$, and where either $t \in T$ or some $t' \in T$ such that t is obtained from t' using a deletion rule d (e.g. by deleting a final or penultimate vowel). For each such possible analysis y', we compute the resulting location in feature space $\mathbf{f}(x, y')$, and select the analysis that brings us closest to our target training language: $y = \text{argmin}_{y'} \parallel \mathbf{f}(x, y') - \mathbf{f}(x_\ell, y_\ell) \parallel$.

2.3.3.3 Stage 2: Find New Stems

In this stage, we keep our set of suffixes F and deletion rules D from the previous stage fixed, and attempt to find new stems to add to T through an aggregate analysis of unsegmented words. For every string s, we consider the set of words which are currently unsegmented, and can be analyzed as a stem-suffix pair (s, f) for some existing suffix $f \in F$, and some deletion rule $d \in D$. We then consider the joint segmentation of these words into a new stem s, and their respective

suffixes. As before, we choose the segmentation if it brings us closer in feature space to our target training language.

2.3.3.4 Stage 3: Find New Suffixes

This stage is exactly analogous to the previous stage, except we now fix the set of stems T and seek to find new suffixes.

2.3.4 A Monolingual Supervised Model

In order to provide a plausible upper bound on performance, we also formulate a supervised monolingual morphological model, using the structured perceptron framework [19]. Here we assume that we are given some training sequence of inputs and morphological analyses (all within one language): $(x_1, y_1), (x_2, y_2)$, ..., (x_n, y_n). We define each input x_i to be a noun w, along with a morphological tag z, which specifies the gender, case, and number of the noun. The goal is to predict the correct segmentation of w into stem, suffix, and phonological deletion rule: $y_i = (t, f, d)$.While the assumption of the correct morphological tag as input is somewhat unrealistic, this model still gives us a strong upper bound on how well we can expect our unsupervised model to perform.

To do so, we define a feature function over input-label pairs, (x, y), with the following binary feature templates:

- According to label y_i, the stem is t (one feature for each possible stem).
- According to label y_i, the suffix and deletion rule are (f, d) (one feature for every possible pair of deletion rules and suffixes).
- According to label y_i and morphological tag z, the suffix, deletion rule, and gender are respectively (f, d, G).
- According to label y_i and morphological tag z, the suffix, deletion rule, and case are (f, d, C).

- According to label y_i and morphological tag z, the suffix, deletion rule, and number are (f, d, N).

We train a set of linear weights on our features using the averaged structured perceptron algorithm [19].

2.4 Experiments and Analysis

In this section we turn to experimental findings to provide empirical support for our proposed framework.

2.4.1 Corpus

	Type Counts				Entropy			Percentage	
	# words	# stems	# suffs	# dels	stem entropy	suff entropy	del entropy	unseg	deleted
BG	4833	3112	21	8	11.4	2.7	0.9	.45	.29
CS	5836	3366	28	12	11.5	3.2	1.6	.38	.53
EN	4178	3453	3	1	11.7	1.0	0.1	.73	.06
ET	6371	3742	141	5	11.5	5.0	0.2	.31	.04
HU	8051	3746	231	7	11.3	5.8	0.5	.23	.11
RO	5578	3297	23	8	11.5	2.9	1.4	.48	.51
SL	6111	3172	32	6	11.3	3.2	1.5	.33	.56
SR	5849	3178	28	5	11.4	2.9	1.4	.33	.53

Table 2.1: Corpus statistics for the eight languages.

To test our cross-lingual model, we apply it to a morphologically analyzed corpus of eight languages [37]. The corpus includes a roughly 100,000 word English text, Orwell's novel "Nineteen Eighty Four," and its translation into seven languages: Bulgarian, Czech, Estonian, Hungarian, Romanian, Slovene, and Serbian. All the words in the corpus are tagged with morphological stems and a detailed morpho-syntactic analysis. Although the texts are parallel, we note that parallelism

is nowhere assumed nor exploited by our model. See Table 2.1 for a summary of relevant corpus statistics. In this table, the first four columns give the number of unique word, stem, suffix, and phonological deletion rule types. The next three columns give, respectively, the entropies of the distributions of stems, suffixes (including $NULL$), and deletion rules (including $NULL$) over word types. The final two columns give, respectively, the percentage of word types occurring with the $NULL$ suffix, and the number of non-$NULL$ suffix words which use a phonological deletion rule. Note that the final eight columns define the universal feature space used by our model. BG = Bulgarian, CS = Czech, EN = English, ET = Estonian, HU = Hungarian, RO = Romanian, SL = Slovene, SR = Serbian.

As indicated in the table, the raw number of nominal word types varies quite a bit across the languages, almost doubling from 4,178 (English) to 8,051, (Hungarian). In contrast, the number of stems appearing within these words is relatively stable across languages, ranging from 3,112 (Bulgarian) and only increasing by 20% with 3,746 unique stems in Hungarian.

Conversely, the number of suffixes across the languages varies quite a bit. Hungarian and Estonian, both Uralic languages with very complex nominal morphology, use 231 and 141 nominal suffixes, respectively. Besides English, the remaining languages employ between 21 and 32 suffixes, and English is the outlier in the other direction, using just three nominal inflectional suffixes.

2.4.2 Baselines and Evaluation

As our unsupervised monolingual baseline, we use the Linguistica program [43; 44]. We apply Linguistica's default settings, and run the "suffix prediction" option. Our model's search procedure closely mirrors the one used by Linguistica, with the crucial difference that instead of attempting to greedily minimize description length, our algorithm instead tries to find the analysis as close as possible in the universal feature space to that of another language.

As a plausible upper bound on performance, we implemented the structured perceptron outlined in Section 2.3.4. For each language, we train the perceptron on a randomly selected set of 80% of the nouns, and test on the remaining 20%.

	Linguistica	Our Model						Supervised
		Nearest Neighbor		Self (oracle)		Avg.		
		Accuracy	Distance	Accuracy	Distance	Accuracy	Distance	
BG	68.7	84.0 (RO)	0.13	88.7	0.03	68.6	3.90	94.7
CS	60.4	82.8 (BG)	0.40	84.5	0.03	66.3	4.05	93.5
EN	81.1	75.8 (BG)	1.29	89.3	0.10	58.3	4.30	93.4
ET	51.2	66.6 (HU)	0.35	80.9	0.03	52.8	4.57	86.5
HU	64.5	69.3 (ET)	0.81	66.5	1.10	68.0	4.94	94.9
RO	65.6	71.0 (CS)	0.11	71.2	0.15	62.3	3.95	89.1
SL	61.1	82.8 (SR)	0.07	85.5	0.04	61.7	3.69	95.4
SR	64.2	79.1 (SL)	0.06	82.2	0.04	63.0	3.71	94.8
avg.	64.6	76.4	0.40	81.1	0.19	62.6	4.14	92.8

Table 2.2: **Prediction accuracy** over word types for the Linguistica baseline, our cross-lingual model, and the monolingual supervised perceptron model.

To apply our model, we treat each of the eight languages in turn as the test language, with the other seven serving as training examples. For each test language, we iterate the search procedure for each training language (performed in parallel), until convergence. The number of required iterations varies from 6 to 36 (depending on the test-training language pair), and each iteration takes no more than 30 seconds of run-time on a 2.4GHz Intel Xeon E5620 processor. We also consider two variants of our method. In the first (**Self (oracle)**), we train each test language to minimize the distance to its *own* gold standard feature values. In the second variant (**Avg.**), we average the feature values of all seven training languages into a single objective. As a plausible upper bound on performance, we implemented the structured perceptron described in the 2.3.4. For each language, we train the perceptron on a randomly selected set of 80% of the nouns, and test on the remaining 20%.

The prediction accuracy for all models is calculated as the fraction of word types with correctly predicted suffixes. See Table 2.2 for the results. For our model, we provide both prediction accuracy and resulting distance to the training language in three different scenarios: (i) **Nearest**

Neighbor: The training languages include all seven other languages in our data set, and the predictions with minimal distance to a training language are chosen (the nearest neighbor is indicated in parentheses). (ii) **Self (oracle):** Each language is trained to minimize the distance to its *own* gold-standard analysis. (iii) **Average:** The feature values of all seven training languages are averaged together to create a single objective.

For all languages other than English (which is a morphological loner in our group of languages), our model improves over the baseline by a substantial margin, yielding an average increase of 11.8 absolute percentage points, and a reduction in error relative to the supervised upper bound of 42%. Some of the most striking improvements are seen on Serbian and Slovene. These languages are closely related to one another, and indeed our model discovers that they are each others' nearest neighbors. By guiding their morphological analyses towards one another, our model achieves a 21 percentage point increase in the case of Slovene and a 15 percentage point increase in the case of Slovene.

Perhaps unsurprisingly, when each language's gold standard feature values are used as its *own* target (**Self (oracle)** in Table 2.2), performance increases even further, to an average of 81.1%. By the same token, the resulting distance in universal feature space between training and test analyses is cut in half under this variant, when compared to the non-oracular nearest neighbor method. The remaining errors may be due to limitations of the search procedure (i.e. getting caught in local minima), or to the coarseness of the feature space (i.e. incorrect analyses might map to the same feature values as the correct analysis). Finally, we note that minimizing the distance to the *average* feature values of the seven training languages (**Avg.** in Table 2.2) yields subpar performance and very large distances between between predicted analyses and target feature values (4.14 compared to 0.40 for nearest neighbor). This result may indicate that the average feature point between training languages is simply unattainable as an analysis of a real lexicon of nouns.

2.4.3 Visualizing Locations in Feature Space

Besides assessing our method quantitatively, we can also visualize the the eight languages in universal feature space according to (i) their gold standard analyses, (ii) the predictions of our

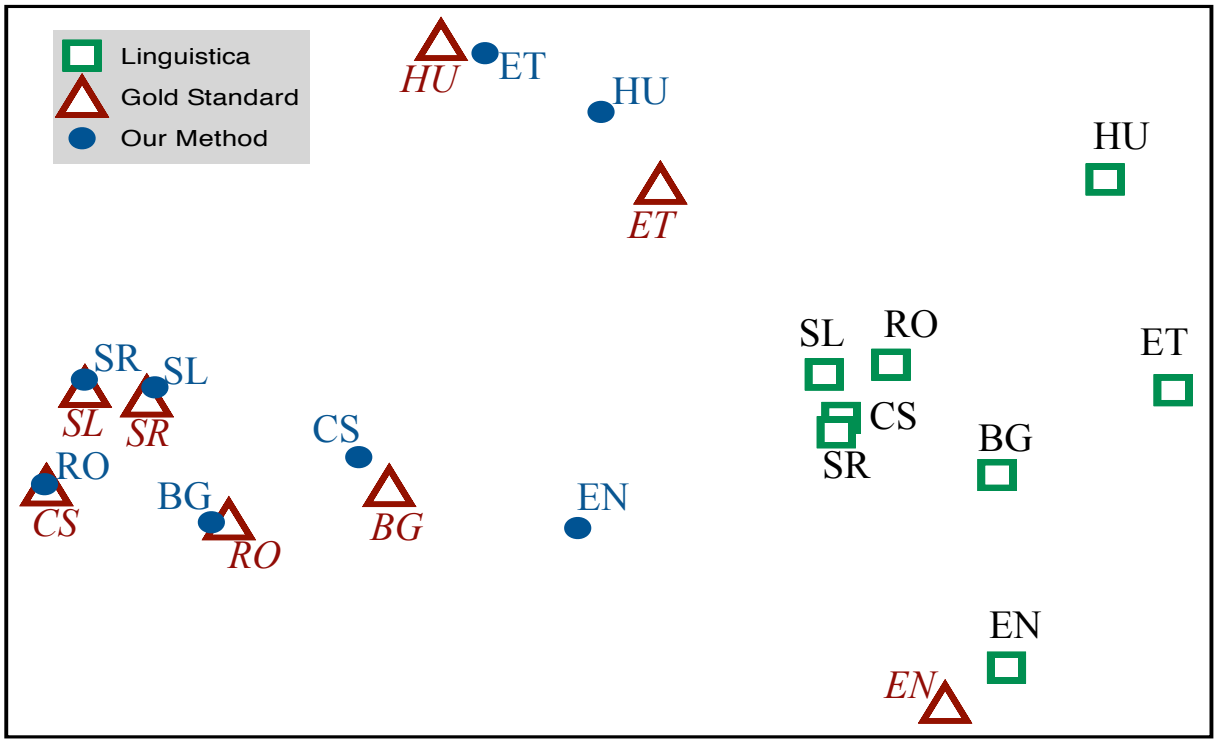

Figure 2.2: **Locations in Feature Space** of Linguistica predictions (green squares), gold standard analyses (red triangles), and our model's nearest neighbor predictions (blue circles). The original 8-dimensional feature space was reduced to two dimensions using Multidimensional Scaling.

model and (iii) the predictions of Linguistica. To do so, we reduce the 8-dimensional features space down to two dimensions while preserving the distances between the predicted and gold standard feature vectors, using Multidimensional Scaling (MDS). The results of this analysis are shown in Figure 2.2. With the exception of English, our model's analyses lie closer in feature space to their gold standard counterparts than those of the baseline. It is interesting to note that Serbian and Slovene, which are very similar languages, have essentially swapped places under our model's analysis, as have Estonian and Hungarian (both highly inflected Uralic languages). English has (unfortunately) been pulled towards Bulgarian, the second least inflecting language in our set.

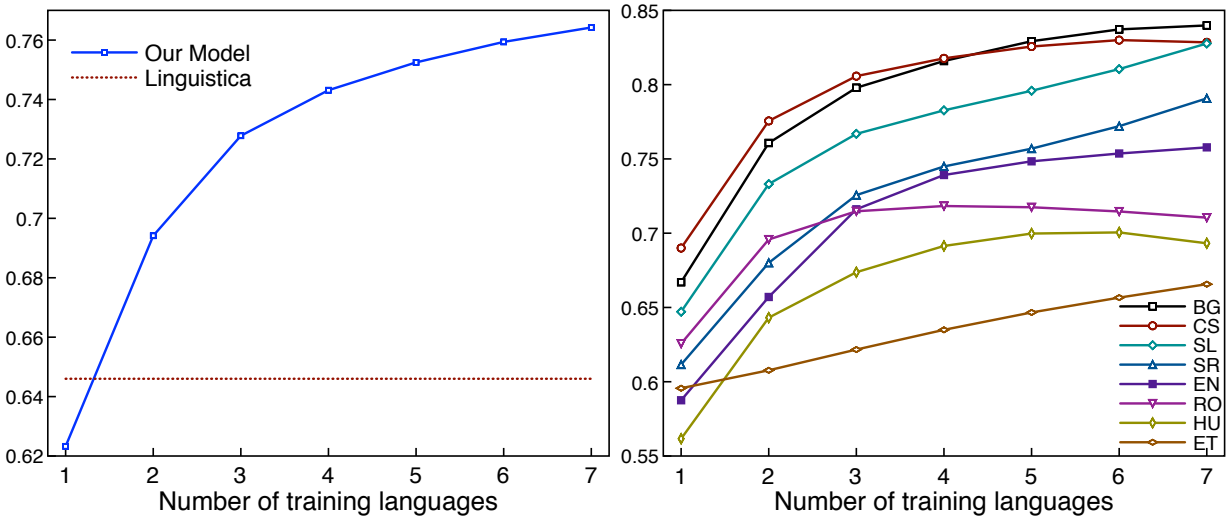

Figure 2.3: **Learning curves** for our model as the number of training languages increases.

2.4.4 Learning Curves

We also measured the performance of our method as a function of the number of languages in the training set. For each target language, we consider all possible training sets of sizes ranging from 1 to 7 and select the predictions which bring our test language closest in distance to one of the languages in the set. We then average the resulting accuracy over all training sets of each size. Figure 2.3 shows the resulting learning curves averaged over all test languages (left), as well as broken down by test language (right). The figure on the left shows the average accuracy of all eight languages for increasingly larger training sets (results are averaged over all training sets of size 1,2,3,...). The dotted line indicates the average performance of the baseline. The figure on the right shows similar learning curves, broken down individually for each test language (see Table 2.1 for language abbreviations). The overall trend is clear: as additional languages are added to the training set, test performance improves. In fact, with only one training language, our method performs worse (on average) than the Linguistica baseline. However, with two or more training languages available, our method achieves superior results.

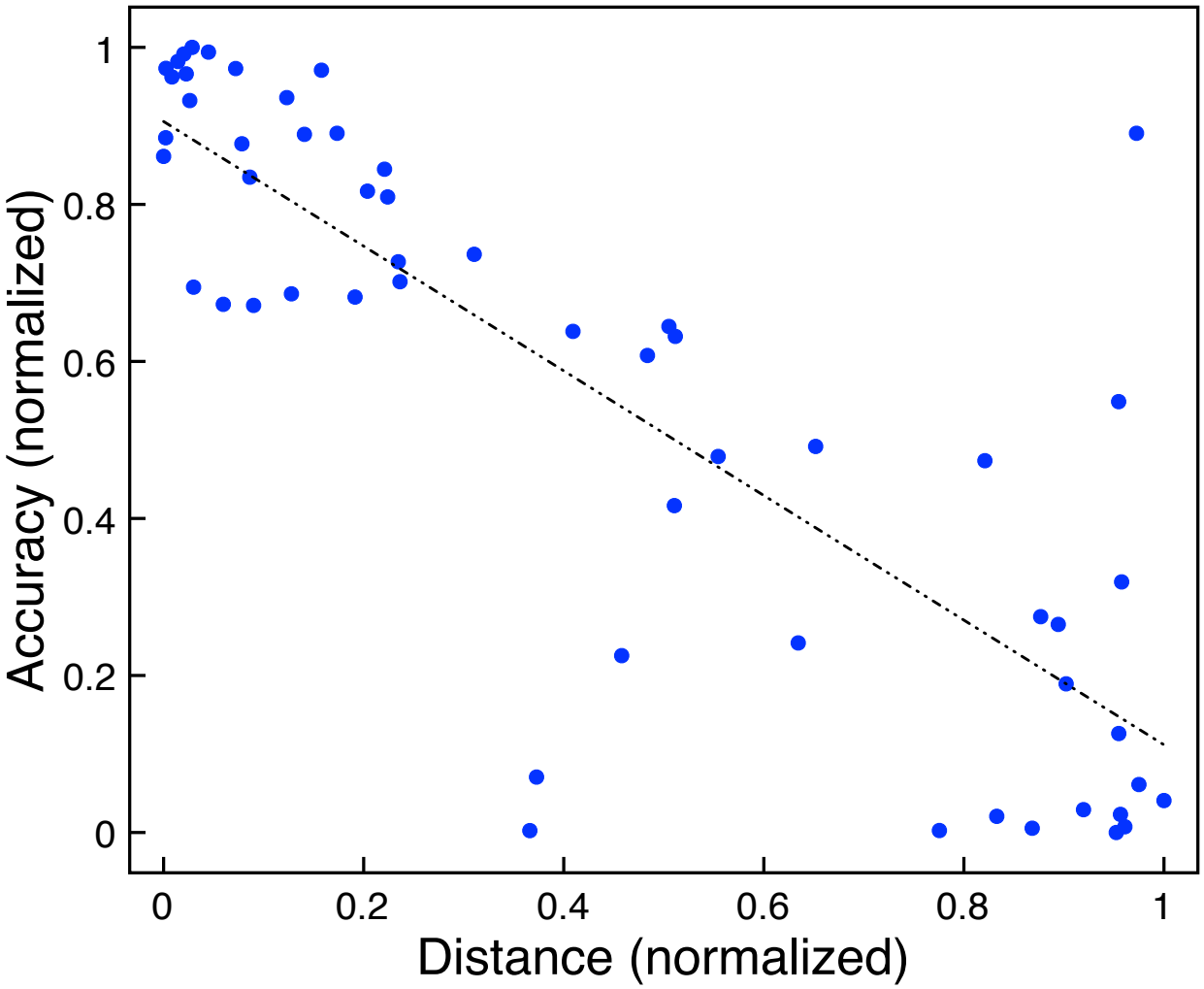

Figure 2.4: **Accuracy vs. Distance:** For all 56 possible test-train language pairs, we computed test accuracy along with resulting distance in universal feature space to the training language. Distance and accuracy are separately normalized to the unit interval for each test language, and all resulting points are plotted together. A line is fit to the points using least-squares regression.

2.4.5 Accuracy vs. Distance

In this final section of analysis, we can gain some insight into these learning curves if we consider the relationship between accuracy (of the test language analysis) and distance to the training language (of the same predicted analysis). The more training languages available, the greater

the chance that we can guide our test language into very close proximity to one of them. It thus stands to reason that a strong (negative) correlation between distance and accuracy would lead to increased accuracy with larger training sets. In order to assess this correlation, we considered all 56 test-train language pairs and collected the resulting accuracy and distance for each pair. We separately scaled accuracy and distance to the unit interval for each test language (as some test languages are inherently more difficult than others). The resulting plot, shown in Figure 2.4, shows the expected correlation: When our test language can be guided very closely to the training language, the resulting predictions are likely to be good. If not, the predictions are likely to be bad.

2.5 Conclusions

In this chapter, we proposed a novel approach to unsupervised morphological analysis. Traditional approaches focus on developing an appropriate hand-crafted inductive bias, whether explicitly as an MDL objective, or implicitly through the design of a model structure. However, incorporating declarative knowledge in a well-founded model is notoriously difficult. Beyond this difficulty lies a deeper problem: It is simply unrealistic to expect a hand-crafted inductive bias or model structure to effectively induce morphology across the full range of the world's languages.

In contrast, our approach treats morphological induction as a structured prediction task, where we assume the presence of morphologically labeled languages as *training examples* which guide the induction process for unlabeled test languages. We developed a novel structured nearest neighbor approach for this task, in which all languages and their morphological analyses lie in a universal feature space. The task of the learner is to search through the space of morphological analyses for the test language and return the result which lies closest in the universal feature space to one of the training languages. Our empirical findings validate this approach: across eight different languages, our method yields substantial accuracy gains in the task of nominal morphological induction, over a traditional MDL-based approach.

Besides gains in accuracy, a substantial advantage of this framework is that it could enable the automatic *interpretation* of morphological analyses. While current methods may segment words

into morphemes with some degree of accuracy, they have no hope of interpreting the grammatical function of the resulting units. In contrast, the approach advocated here may in the future allow us to automatically infer plausible interpretations of morphemes, by examining cross-lingual regularities of the various morphemic functions.

Chapter 3

Grapheme-to-Phoneme Analysis of Latin Alphabets

In the previous chapter, we considered the task of morphology analysis for Eastern European languages. In this chapter, we extend the idea of cross-lingual supervised learning to the task of grapheme-to-phoneme analysis of Latin alphabets as a test case for larger scale cross-lingual learning by using information from dozens of other languages. The goal is to automatically select a subset of phonemes for each language's graphemes. For example, depending on the language, the Latin alphabet grapheme "c" can represent any of the following phonemes:

$$\{/\text{k}/, /\text{c}/, /\text{s}/, /|/, /\text{d}\text{ʒ}/, /\text{t}\text{ʃ}/, /\text{ts}/\}$$

Among those phonemes, we would like to predict the set $\{/\text{s}/, /\text{k}/\}$ for the grapheme "c" in English, using only raw text as our input. This work was originally published in [59].

This chapter is organized as follows: Section 3.1 gives a broad introduction to the chapter. We argue that by harnessing dozens of languages the relationship between written symbol and spoken sound can be automatically inferred from textual patterns. We briefly describe our approach and summarize our experimental findings. Section 3.2 describes background and compares our approach to previous grapheme-to-phoneme analysis. Section 3.3 describes our features in great detail. Section 3.4 fully describes our model. Section 3.5 outlines experiments on 107 languages with Roman alphabets and reports our results. Section 3.5.4 analysis the predicted phoneme inventory from our model. Section 3.6 closes the chapter with some concluding remarks.

3.1 Introduction

Written languages are one of the defining technologies of human civilization. They have been independently invented at least three times through the course of history [25]. In many ways, written languages reflect their spoken counterparts. Written and spoken languages are subject to some of the same forces of change: migration, adaptation, borrowing, cultural influence, and imposition by empire and rulers. In other words, written languages harken further to the past, reflecting aspects of languages long gone from their spoken forms. In this chapter, we argue that this imperfect relationship between written symbols and spoken sounds can be automatically inferred from textual patterns. By examining data for over 100 languages, we train a statistical model to automatically relate graphemic patterns in texts to phonemic sequences for never-before-seen languages.

We focus here on the the alphabet, a writing system that has come down to us from the Sumerians, through the Greeks and Romans: the Latin alphabet. In a perfect alphabetic system, each phoneme in the language is unambiguously represented by a single grapheme. However, in practice of course, this ideal is never achieved. Besides, when existing alphabets are melded onto new languages, they must be imperfectly adapted to a new sound system. This ambiguity can pose difficulties for applications such as speech recognition and text-to-speech. In this chapter, we exploit the fact that a single alphabet, that of the Romans, has been adapted to a very large variety of languages.

Recent research has demonstrated the effectiveness of cross-lingual analysis. The joint analysis of several languages can increase model accuracy, and enable the development of computational tools for languages with minimal linguistic resources. Previous work has focused on settings where just a handful of languages are available. We treat the task of grapheme-to-phoneme analysis as a test case for larger scale multilingual learning, harnessing information from dozens of languages.

On a more practical note, accurately relating graphemes and phonemes to one another is crucial for tasks such as automatic speech recognition and text-to-speech generation. While pronunciation dictionaries and transcribed audio are available for some languages, these resources are entirely

lacking for the vast majority of the world's languages. Thus, automatic and generic methods for determining sound-symbol relationships are required.

Basically, this work is based on the following line of reasoning:

- First, character-level textual patterns mirror phonotactic regularities.
- Second, phonotactic regularities are shared across related languages and universally constrained.
- Finally, textual patterns for a newly observed language may thus reveal its underlying phonemics.

Our task can be viewed as an easy case of language decipherment (will be discussed in Chapter 5) – one where the underlying alphabetic system is widely known.

Nevertheless, the task of grapheme-to-phoneme analysis is still challenging. Characters in the Roman alphabet can take a wide range of phonemic values across the world's languages. For example, depending on the language, the grapheme "c" can represent the following phonemes:[1]

- /k/ (unvoiced velar plosive)
- /c/ (unvoiced palatal plosive)
- /s/ (unvoiced alveolar fricative)
- /|/ (dental click)
- /d͡ʒ/ (affricated voiced postalveolar fricative)
- /t͡ʃ/ (affricated unvoiced postalveolar fricative)
- /t͡s/ (affricated unvoiced alveolar fricative)

[1]For some brief background on phonetics, see Section 3.2. Note that we use the term "phoneme" throughout the chapter, though we also refer to "phonetic" properties. As we are dealing with texts (written in a roughly phonemic writing system), we have no access to the true contextual phonetic realizations, and even using IPA symbols to relate symbols across languages is somewhat theoretically suspect.

To make matters more challenging, the same language may use a single grapheme to ambiguously represent *multiple* phonemes. For example, English orthography uses the letter"c" to represent both the sounds /k/ and /s/. Our task is thus to select a *subset* of phonemes for each language's graphemes. We treat the subset selection problem as a set of related binary prediction problems, one for each possible grapheme-phoneme pair. Taken together, these predictions yield the grapheme-phoneme mapping for that language.

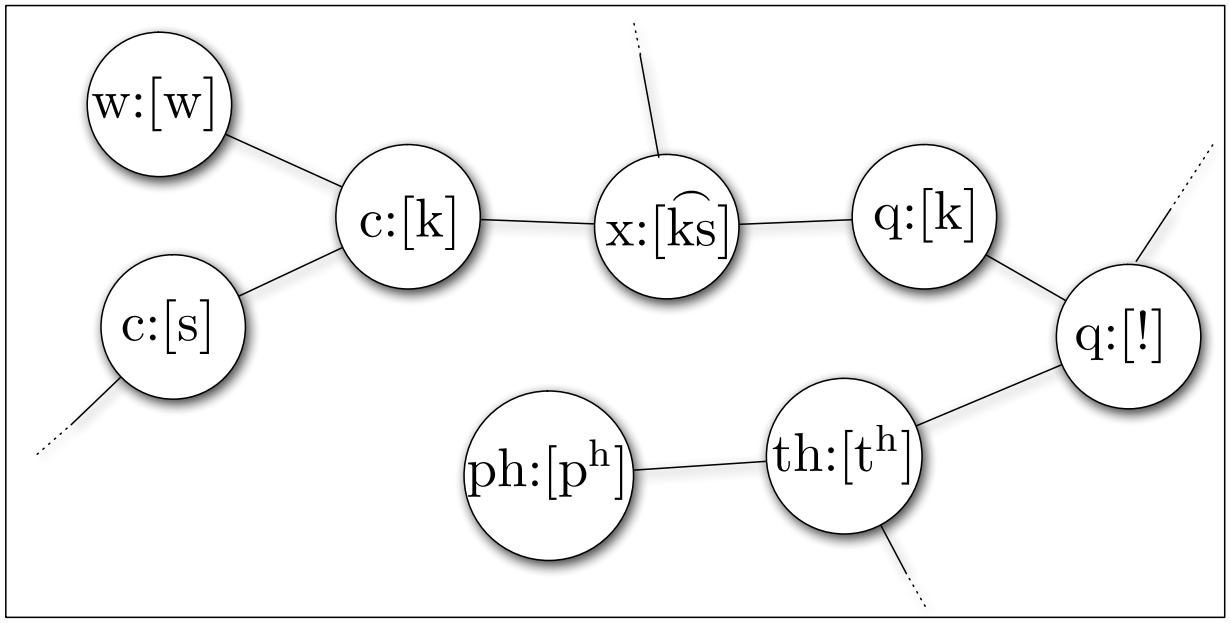

Figure 3.1: **A snippet of our undirected graphical model.** The binary-valued nodes represent whether a particular grapheme-phoneme pair is allowed by the language. Sparse edges are automatically induced to allow joint training and prediction over related inventory decisions.

We develop a probabilistic undirected graphical model for this prediction problem, where a large set of languages serve as a training data and a single held-out language serves as a test data. Each training and test language yields an instance of the graph, bound together through a shared set of features and parameter values to allow cross-lingual learning and generalization.

In the graph corresponding to a given language, each node represents a grapheme-phoneme pair $(g : p)$. The node is labeled with a binary value to indicate whether grapheme g can represent

phoneme p in that language. In order to allow coupled labelings across the various grapheme-phoneme pairs of the language, we employ a connected graph structure, with an automatically learned topology shared across different languages. The node and edge features are derived from textual co-occurrence statistics for the graphemes of each language, as well as general information about the language's family and region. Parameters are jointly optimized over the training languages to maximize the likelihood of the node labelings given the observed feature values. See Figure 3.1 for a snippet of the model.

We apply our model to a novel data-set consisting of grapheme-phoneme mappings for 107 languages with Roman alphabets and short texts. In this setting, we consider each language in turn as the test language, and train our model on the remaining 106 languages. Our highest performing model achieves an F1-measure of 88%, yielding perfect predictions for over 21% of languages. These results compare quite favorably to several baselines.

3.2 Background and Related Work

3.2.1 Phoneme Inventories

The sounds of the world's languages are produced through a wide variety of articulatory mechanisms. Consonants are sounds produced through a partial or complete stricture of the vocal tract. They are roughly categorized along three independent dimensions:

- *Voicing:* whether or not oscillation of the vocal folds accompanies the sound. For example, /t/ and /d/ differ only in that the latter is voiced.

- *Place of Articulation:* where in the anatomy of the vocal tract the stricture is made. For example, /p/ is a bilabial (the lips touching one another) while /k/ is a velar (tongue touching touching the soft palate).

- *Manner of Articulation:* the manner in which the airflow is regulated. For example, /m/ is a nasal (air flowing through the nostrils), while /p/ is a plosive (obstructed air suddenly released through the mouth).

In contrast, vowels are voiced sounds produced with an open vocal tract. They can be categorized primarily based on the position of the tongue and lips, along three dimensions:

- *Roundedness:* whether or not the lips are rounded during production of the sound.
- *Height:* the vertical position of the tongue.
- *Backness:* how far forward the tongue lies.

Linguists have noted several statistical regularities found in phoneme inventories throughout the world. First, *feature economy* refers to the idea that languages tend to minimize the number of differentiating characteristics (e.g. different kinds of voicing, manner, and place) that are used to distinguish consonant phonemes from one another [17]. In other words, once an articulatory feature is used to mark off one phoneme from another, it will likely be used again to differentiate other phoneme pairs in the same language. In addition, the principle of *Maximal perceptual contrast* refers to the idea that the set of vowels employed by a language will be located in phonetic space to maximize their perceptual distances from one another, thus relieving the perceptual burden of the listener [69]. An analysis of our results indicates that our model's predictions do not always follow these principles. In a preliminary experiment, we show how utilizing these principles can improve performance.

Finally, researchers have noted that languages exhibit set patterns in how they sequence their phonemes [56]. Certain sequences are forbidden outright by languages, while others are avoided or favored. While many of these patterns are language-specific, others seem more general, either reflecting anatomical constraints, common language ancestry, or universal aspects of the human language system. These phonotactic regularities and constraints are mirrored in graphemic patterns. As shown in our experiments, these can be explicitly modeled to achieve high accuracy in our task.

3.2.2 Grapheme-to-Phoneme Analysis

Much prior work has gone into developing methods for accurate grapheme-to-phoneme analysis. The common assumption underlying this research has been that some sort of knowledge,

usually in the form of a pronunciation dictionary or phonemically annotated text, is available for the language at hand. The focus has been instead on developing techniques for dealing with the phonemic ambiguity present both in annotated and unseen words. For example, Jiampojamarn and Kondrak [53] developed a method for aligning pairs of written and phonemically transcribed strings; Dwyer and Kondrak [34] developed a method for accurate letter-to-phoneme conversion while minimizing the number of training examples; Reddy and Goldsmith [85] develop an MDL-based approach to finding sub-word units that align well to phonemes.

A related line of work has grown around the task of machine transliteration. In this task, the goal is to automatically transliterate a name in one language into the written form of another language. Often this involves some level of phonetic analysis in one or both languages. Notable recent work in this vein includes research by Sproat et al [98] on transliteration between Chinese and English using comparable corpora, and Ravi and Knight [84] who take a decipherment approach to this problem.

Our work differs from all previous work on grapheme-to-phoneme analysis in that (i) we assume no knowledge for our target language beyond a small unannotated text, and possibly some region or language family information, and (ii) our goal is to construct the inventory of mappings between the language's letters and its phonemes (the latter of which we do not know ahead of time). When a grapheme maps to more than one phoneme, we do not attempt to disambiguate particular instances of that grapheme in words.

A final thread of related work is on quantitatively categorizing writing systems according to their levels of phonography and logography [99; 81]. As our data-set consists entirely of Latin-based writing systems, our work can be viewed as a more fine-grained computational exploration of the space of writing systems, with a focus on phonographic systems with a particular pedigree.

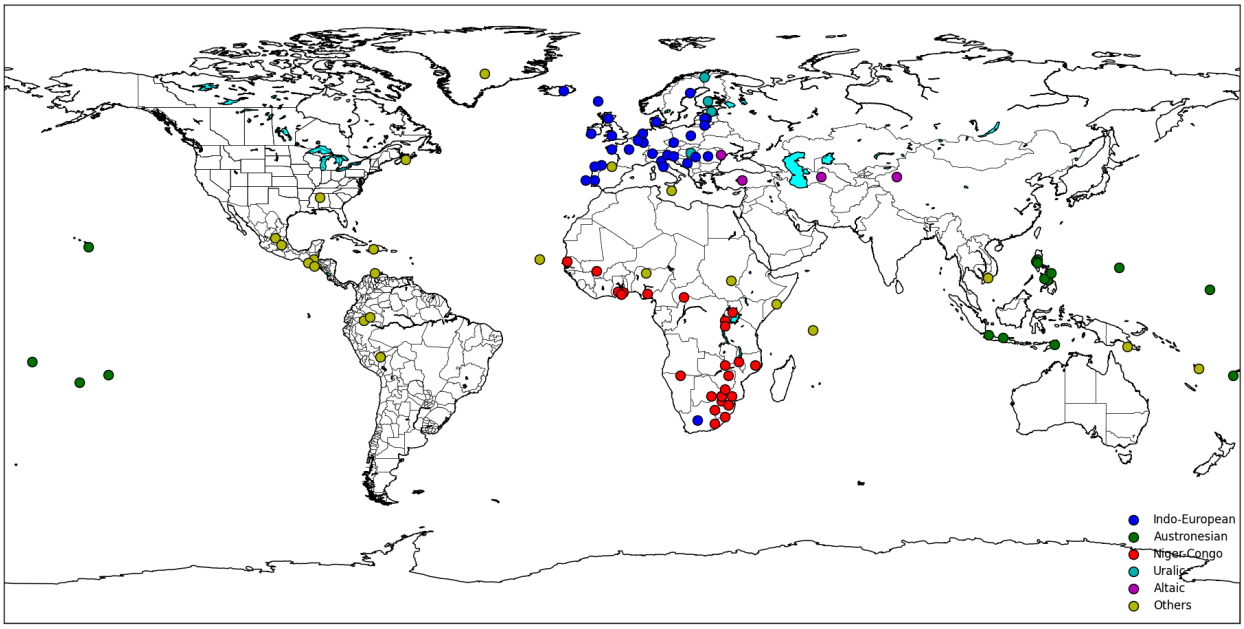

Figure 3.2: Map and language families of languages in our data-set.

3.2.3 Data

The data for our experiments comes from three sources: (i) grapheme-phoneme mappings from an online encyclopedia, (ii) translations of the Universal Declaration of Human Rights (UDHR)[2], and *(iii)* entries from the World Atlas of Language Structures (WALS) [32].

To start with, we downloaded and transcribed image files containing grapheme-phoneme mappings for several hundred languages from an online encyclopedia of writing systems[3]. We then cross-referenced the languages with the World Atlas of Language Structures (WALS) database [32] as well as the translations available for the Universal Declaration of Human Rights (UDHR). Our final set of 107 languages includes those which appeared consistently in all three sources and that employ a Latin alphabet. See Figure 3.2 for a world map annotated with the locations listed in the WALS database for these languages, as well as their language families. As seen from the figure, these languages cover a wide array of language families and regions.

[2]http://www.ohchr.org/en/udhr/pages/introduction.aspx
[3]http://www.omniglot.com/writing/langalph.htm#latin

We then analyzed the phoneme inventories for all the 107 languages. We decided to focus our attention on graphemes which are widely used across these languages with a diverse set of phonemic values. We measured the ambiguity of each grapheme by calculating the entropy of its phoneme sets across the languages. We found that 17 graphemes had entropy > 0.5 and appeared in at least 15 languages. Table 3.1 lists these graphemes, the set of phonemes that they can represent, the number of languages in our data-set which employ them, and the entropy of their phoneme-sets across these languages. The data, along with the feature vectors discussed below, are published as part of this work.

3.3 Features

The key intuition underlying this work is that graphemic patterns in text can reveal the phonemes which they represent. A crucial step in operationalizing this intuition lies in defining input features that have cross-lingual predictive value. We divide our feature set into three categories.

3.3.1 Character Context Features

These features represent the textual environment of each grapheme in a language. For each grapheme g, we consider counts of graphemes to the immediate left and right of g in the UDHR text. We define five feature templates, including counts of (1) single graphemes to the left of g, (2) single graphemes to the right of g, (3) pairs of graphemes to the left of g, (4) pairs of graphemes to the right of g, and (5) pairs of graphemes surrounding g. As our experiments below show, this set of features on its own performs poorly. It seems that these features are too language specific and not abstract enough to yield effective cross-lingual generalization. Our next set of features was designed to alleviate this problem.

3.3.2 Phonetic Context Features

A perfect feature-set would depend on the entire set of grapheme-to-phoneme predictions for a language. In other words, we would ideally map the graphemes in our text to phonemes, and then consider the plausibility of the resulting phoneme sequences. In practice, of course, this is

Graphemes	Phonemes	#langs	Entropy
a	/a/ /ɐ/ /ɑ/ /ə/ /ʌ/	106	1.25
c	/c/ /d͡ʒ/ /k/ /s/ /t͡s/ /t͡ʃ/ /ǀ/	62	2.33
ch	/k/ /t͡ʃ/ /x/ /ʃ/	39	1.35
e	/e/ /i/ /æ/ /ə/ /ɛ/	106	1.82
h	/-/ /h/ /x/ /ø/ /ɦ/	85	1.24
i	/i/ /j/ /ɪ/	106	0.92
j	/d͡ʒ/ /h/ /j/ /t͡ʃ/ /x/ /ɟ/ /ʒ/	79	2.05
o	/o/ /u/ /ɒ/ /ʉ/	103	1.47
ph	/f/ /pʰ/	15	0.64
q	/k/ /q/ /!/	32	1.04
r	/r/ /ɽ/ /ɾ/ /ʀ/ /ʁ/	95	1.50
th	/tʰ/ /θ/	15	0.64
u	/u/ /w/ /y/ /ɨ/ /ʊ/ /ʏ/	104	0.96
v	/b/ /f/ /v/ /w/ /β/	70	1.18
w	/u/ /v/ /w/	74	0.89
x	/k͡s/ /x/ /‖/ /ʃ/	44	1.31
z	/d͡z/ /s/ /t͡s/ /z/ /θ/	72	0.93

Table 3.1: Ambiguous graphemes and the set of phonemes that they may represent among our set of 107 languages.

impossible, as the set of possible grapheme-to-phoneme mappings is exponentially large. As an imperfect proxy for this idea, we made the following observation: for most Latin graphemes, the most common phonemic value across languages is the identical IPA symbol of that grapheme (e.g. the most common phoneme for *g* is /g/, the most common phoneme for *t* is /t/, etc). Using this observation, we again consider all contexts in which a grapheme g appears, but this time map the surrounding graphemes to their IPA phoneme equivalents. We then consider various linguistic properties of these surrounding "phonemes" – whether they are vowels or consonants, whether they

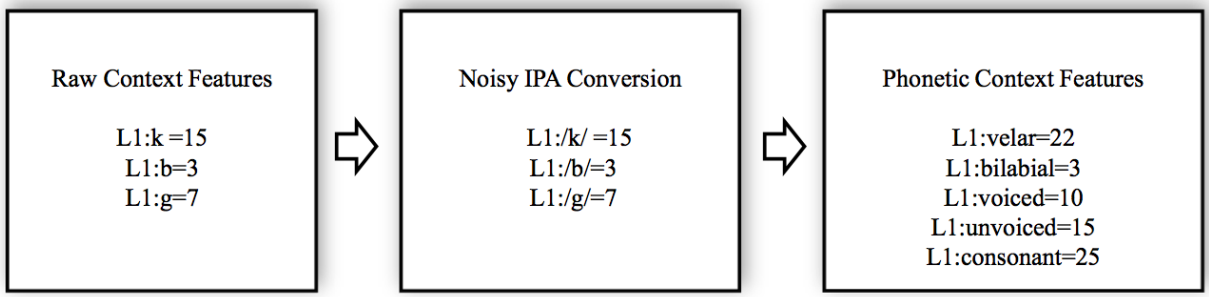

Figure 3.3: **Generating phonetic context features.** First, character context features are extracted for each grapheme. The features drawn here give the counts of the character to the immediate left of the grapheme. Next, the contextual characters are noisily converted to phonemes using their IPA notation. Finally, phonetic context features are extracted. In this case, phonemes /k/ and /g/ combine to give a "velar" count of 22, while /g/ and /b/ combine to give a "voiced" count of 10.

are voiced or not, their manner and places of articulation – and create phonetic context features. The process is illustrated in Figure 3.3. The intuition here is that these features can (noisily) capture the *phonotactic context* of a grapheme, allowing our model to learn general phonotactic constraints. As our experiments below demonstrate, these features proved to be quite powerful.

3.3.3 Language Family Features

Finally, we consider features drawn from the WALS database which capture general information about the language – specifically, its region (e.g. Europe), its small language family (e.g. Germanic), and its large language family (e.g. Indo-European). These features allow our model to capture family and region specific phonetic biases. For example, African languages are more likely to use *c* and *q* to represents clicks in comparison to European languages. As we mention below, we also consider conjunctions of all features. Thus, a language family feature can combine with a phonetic context feature to represent a family specific phonotactic constraint. Interestingly, our experiments below show that these features are not needed for highly accurate prediction.

3.3.4 Feature Discretization and Filtering

It is well known that many learning techniques perform best when continuous features are binned and converted to binary values [31]. As a preprocessing step, we therefore discretize and filter the count-based features outlined above. We adopt the technique of Recursive Minimal Entropy Partitioning [52]. This technique recursively partitions feature values so as to minimize the conditional entropy of the labels. Partitioning stops when the gain in label entropy falls below the number of additional bits in overhead needed to describe the new feature split. This leads to a (local) minimum description length discretization.

Number of Features	Raw	Filtered
# Text Features	28,474	1,848
# Phonemic Features	28,948	7,799
# Family Features	66	32
Total	57,488	9,679

Table 3.2: **Number of features** in each category before and after discretization/filtering. Note that the pair-wise conjunction features are not included in these counts.

We noticed that most of our raw features (especially the text features) could not achieve even a single split point without increasing description length, as they were not well correlated with the labels. We decided to use this heuristic as a feature selection technique, discarding such features. After this discretization and filtering, we took the resulting binary features and added their pairwise conjunctions to the set. This process was conducted separately for each leave-one-out scenario, without observation of the test language labels. Table 3.2 shows the total number of features before the discretization/filtering as well as the typical numbers of features obtained after filtering (the exact numbers depend on the training/test split).

3.4 Model

Using the features described in 3.3, we develop an undirected graphical model approach to our prediction task. Corresponding to each training language is an instance of our undirected graph, labeled with its true grapheme-phoneme mapping. We learn weights over our features which optimally relate the input features of the training languages to their observed labels. At test-time, the learned weights are used to predict the labeling of the held-out test language.

$$\begin{aligned}
\log P\left(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}\right) = & \sum_i \lambda_i \cdot [f_i(\mathbf{x}^{(l)})\delta(y_i^{(l)} = 1)] \\
& + \sum_{j,k} \lambda_{jk1} \cdot [g_{jk}(\mathbf{x}^{(l)})\delta(y_j^{(l)} = 1 \wedge y_k^{(l)} = 1)] \\
& + \sum_{j,k} \lambda_{jk2} \cdot [g_{jk}(\mathbf{x}^{(l)})\delta(y_j^{(l)} = 1 \wedge y_k^{(l)} = 0)] \\
& + \sum_{j,k} \lambda_{jk3} \cdot [g_{jk}(\mathbf{x}^{(l)})\delta(y_j^{(l)} = 0 \wedge y_k^{(l)} = 1)] \\
& - \log Z(\mathbf{x}^{(l)}, \lambda)
\end{aligned}$$

More formally, we assume a set of graph nodes 1, ..., m with edges between some pairs of nodes (i,j). Each node corresponds to a grapheme-phoneme pair ($g : p$) and can be labeled with a binary value. For each training language l, we observe a text $\mathbf{x}^{(l)}$ and a binary labeling of the graph nodes $\mathbf{y}^{(l)}$. For each node i, we also obtain a feature vector $f_i(\mathbf{x}^{(l)})$, by examining the language's text and extracting textual and noisy phonetic patterns. We obtain similar feature vectors for edges (i,j): $g_{jk}(\mathbf{x}^{(l)})$. We then parameterize the probability of each labeling using a log-linear from over node and edge factors:[4]

The first sum ranges over the nodes i in the graph. For each i, we extract a feature vector $f_i(\mathbf{x}^{(l)})$. If the label of node i is 1, we take dot product of the feature vector and corresponding parameters, otherwise the term is zeroed out. Likewise for the graph edges j, k: we extract a feature vector, and depending on the labels of the two vertices y_i and y_k, take a dot product with

[4]The delta function $\delta(p)$ evaluates to 1 when predicate p is true, and to 0 when p is otherwise.

the relevant parameters. The final term is a normalization constant to ensure that the probabilities sum to one over all possible labelings of the graph.

Before learning our parameters, we first automatically induce the set of edges in our graph, using the PC graph-structure learning algorithm [97]. This procedure starts with a fully connected undirected graph structure, and iteratively removes edges between nodes that are conditionally independent given other neighboring nodes in the graph according to a statistical independence test. In our graphs we have 75 nodes, and thus 2,775 potential edges. Running the structure learning algorithm on our data yields sparse graphs, typically consisting of about 50 edges.

Once the graph structure has been induced, we learn parameter values by maximizing the L2-penalized log-likelihood over all training languages:[5]

$$L(\lambda) = \sum_{l} \log P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}) - C||\lambda||^2$$

The gradient takes the standard form of a difference between expected and observed feature counts. Expected counts, as well as predicted assignments at test-time, are computed using loopy belief propagation. Numerical optimization is performed using L-BFGS.

3.5 Experiments

We describe a set of experiments performed to evaluate the performance of our model. Besides our primary undirected graphical model, we also consider several baselines and variants, in order to assess the contribution of our model's graph structure as well as the features used. In all cases, we perform leave-one-out cross-validation over the 107 languages in our data-set.

3.5.1 Baselines

Our baselines include:

1. A majority baseline, where the most common binary value is chosen for each grapheme-phoneme pair,

[5] In our experiments, we used an L2 penalty weight of .5 for node features and .1 for edge features. Similar results are observed for a wide range of values.

	Phoneme			Grapheme	Language
	Precision	Recall	F1	Accuracy	Accuracy
MAJORITY	80.47	57.47	67.06	55.54	2.8
SVM CONTINUOUS	79.87	64.48	79.87	59.07	3.74
SVM DISCRETE	90.55	78.27	83.97	70.78	8.41
NEAREST NEIGHBOR	85.35	79.43	82.28	67.97	2.8
MODEL: NO EDGES	89.35	82.05	85.54	73.96	10.28
FULL MODEL	91.06	83.98	87.37	78.58	**21.5**
MODEL: NO FAMILY	**92.43**	**84.67**	**88.38**	**80.04**	19.63
MODEL: NO TEXT	89.58	81.43	85.31	75.86	15.89
MODEL: NO PHONETIC	86.52	74.19	79.88	69.6	9.35

Table 3.3: The performance of baselines and variants of our model, evaluated at the phoneme-level (binary predictions), whole-grapheme accuracy, and whole-language accuracy.

2. Two linear SVM's, one trained using the discretized and filtered features described in Section 3.3, and the other using the raw continuous features,

3. A nearest Neighbor classifier, which chooses the closest training language for each grapheme-phoneme pair in the discretized feature space, and predicts its label, and

4. A variant of our model with no edges between nodes (essentially reducing to a set of independent log-linear classifiers).

3.5.2 Evaluation

We report our results using three evaluation metrics of increasing coarseness.

1. **Phoneme-level**: For individual grapheme-phoneme pairs (e.g. a:/ɐ/, a:/ʌ/, c:/k/, c:/tʃ/) our task consists of a set of binary predictions, and can thus be evaluated in terms of precision, recall, and F1-measure. We report micro-averages of these quantities across all grapheme-phoneme pairs in all leave-one-out test languages.

2. **Grapheme-level**: We also report grapheme-level accuracy. For this metric, we consider each grapheme g and examine its predicted labels over all its possible phonemes: $(g : p_{(i)})_{i=1}^{k}$

   If all k binary predictions are correct, then the grapheme's phoneme-set has been correctly predicted. We report the percentage of all graphemes with such correct predictions (micro-averaged over all graphemes in all test language scenarios).

3. **Language-level**: Finally, we assess language-wide performance. For this metric, we report the percentage of test languages for which our model achieves perfect predictions on all grapheme-phoneme pairs, yielding a perfect mapping.

3.5.3 Results

The results for the baselines and our model are shown in Table 3.3. The majority baseline yields 67% F1-measure on the phoneme-level binary prediction task, with 56% grapheme accuracy, and about 3% language accuracy.

Using undiscretized raw count features, the SVM improves phoneme-level performance to about 80% F1, but fails to provide any improvement on grapheme or language performance. In contrast, the SVM using discretized and filtered features achieves performance gains in all three categories, achieving 71% grapheme accuracy and 8% language accuracy. The nearest neighbor baseline achieves performance somewhere in between the two SVM variants.

The unconnected version of our model achieves similar, though slightly improved performance over the discretized SVM. Adding the automatically induced edges into our model leads to significant gains across all three categories. Phoneme-level F1 reaches 87%, grapheme accuracy hits 79%, and language accuracy more than doubles, achieving 22%. It is perhaps not surprising that the biggest relative gains are seen at the language level: by jointly learning and predicting an entire language's grapheme-phoneme inventory, our model ensures that language-level coherence is maintained.

Recall that three sets of features are used by our models. (1) language family and region features, (2) textual context features, and (3) phonetic context features. We now assess the relative

merits of each set by considering our model's performance when the set has been removed. Table 3.3 shows several striking results from our experiment. First, it appears that dropping the region and language family features actually improves performance. This result is somewhat surprising, as we expected these features to be quite informative. However, it appears that whatever information they convey is redundant when considering the text-based feature sets. We next observe that dropping the textual context features leads to a small drop in performance. Finally, we see that dropping the phonetic context features seriously degrades our model's accuracy. Achieving robust cross-linguistic generalization seems to require a level of feature abstraction not achieved by character-level context features alone.

3.5.4 Global Inventory Analysis

We analyze the predicted phoneme inventories and ask whether they display the statistical properties observed in the gold-standard mappings. As outlined before, consonant phonemes can be represented by the three articulatory features of voicing, manner, and place. The principle of feature economy states that phoneme inventories will be organized to minimize the number of distinct articulatory features used in the language, while maximizing the number of resulting phonemes. This principle has several implications. First, we can measure the *economy index* of a consonant system by computing the ratio of the number of consonantal phonemes to the number of articulatory features used in their production: $\frac{\#\text{consonants}}{\#\text{features}}$ [17].

Second, for each articulatory dimension we can calculate an empirical distribution over feature values observed across the consonants of the language. Since consonants are produced as combinations of the three articulatory dimensions, the greatest number of consonants (for a given set of feature values) will be produced when the distributions over feature values are close to uniform. Thus, we can measure how economical each feature dimension is by computing the entropy of its distribution over consonants. For example, in an economical system, we would expect roughly half the consonants to be voiced, and half to be unvoiced.

Table 3.4 shows the results of this analysis. First, we notice that the average entropy of voiced vs. unvoiced consonants is nearly identical in both cases, close to the optimal value. However,

	H(voice)	H(place)	H(manner)	Economy Index
True	0.9739	2.7355	2.4725	1.6536
Predicted	0.9733	2.6715	2.4163	1.6337

Table 3.4: Measures of feature economy applied to the predicted and true consonant inventories (averaged over all 107 languages).

when we examine the dimensions of place and manner, we notice that the entropy induced by our model is not as high as that of the true consonant inventories, implying a suboptimal allocation of consonants. In fact, when we examine the economy index (ratio of consonants to features), we indeed find that – on average – our model's predictions are not as economical as the gold standard. This analysis suggests that we might obtain a more powerful predictive model by taking the principle of feature economy into account.

3.6 Conclusions

In this chapter, we considered a novel problem: that of automatically relating written symbols to spoken sounds for an unknown language using a known writing system – the Latin alphabet. We constructed a data-set consisting of grapheme-phoneme mappings and a short text for over 100 languages. This data allows us to cast our problem in the supervised learning framework, where each observed language serves as a training example, and predictions are made on a new language. Our model automatically learns how to relate textual patterns of the unknown language to plausible phonemic interpretations using induced phonotactic regularities.

From our experiments, we draw several conclusions.

1. Character co-occurrence features alone are not sufficient for cross-lingual predictive accuracy in this task. Instead, we map raw contextual counts to more linguistically meaningful generalizations to learn effective cross-lingual patterns.

2. A connected graph topology is crucial for learning linguistically coherent grapheme-to-phoneme mappings. Without any edges, our model yields perfect mappings for only 10% of

test languages. By employing structure learning and including the induced edges, we more than double the number of test languages with perfect predictions.

3. Finally, an analysis of our grapheme-phoneme predictions shows that they do not achieve certain global characteristics observed across true phoneme inventories. In particular, the level of "feature economy" in our predictions is too low, suggesting an avenue for future research.

Chapter 4

Interlude: Tracing the History of the Roman Alphabet

As a short interlude, we pause at this moment to show the deep relationship between cross-lingual phonotactic regularities and the history of the Latin alphabet.

In this chapter, we propose a data-set and a method for the automatic reconstruction of the history of the Roman alphabet as it has spread across time and space. Our data-set consists of grapheme-phone mappings for close to 300 languages, collated with meta-data including language family and geographical location. Using only two simple distance features, our clustering results are remarkably consistent with what is known from historical knowledge.

This chapter is organized as follows: Section 4.1 gives a broad introduction to the chapter. We argue that cross-lingual grapheme-phone ambiguity can serve as a means for the automatic analysis of the *history* of the alphabet. Section 4.2 describes our data-set for history reconstruction. Section 4.3 fully describes two distance measures for clustering. Section 4.4 outlines qualitatively assessment results on our clustering compared to what we know of the history of the alphabet. Section 4.5 completes the chapter with some concluding remarks.

4.1 Introduction

In this chapter, we focus on the Latin alphabet as in Chapter 3. In a idealized alphabetic system, each phoneme in the language is unambiguously represented by a single grapheme, but this ideal is never achieved in practice because they must be re-purposed for a new sound system when existing alphabets are melded onto a new language. However, we argue that cross-lingual grapheme-phone ambiguity can serve as a means for the automatic analysis of the *history* of the alphabet

Our key premise is that as an alphabet gets passed from one language to another, the idiosyncratic writing features of the source language get passed on to the target language alphabet, allowing us to retrace the history of the writing system over time and place.

As a motivating example, consider the familiar letter j. In classical Latin, this was an orthographic variation of the letter i, and both symbols represented the palatal approximant /j/ (the "y" sound in English). Indeed, the majority of European Roman alphabets – including those used by most Germanic, Uralic, Slavic, and Baltic languages – continue to employ the j:/j/ mapping to this day.

In French and Portuguese, however, the Latin /j/ phone was fronted to become /ʒ/, resulting in the grapheme-phone mapping of j:/ʒ/. In the middle ages, English adapted this French mapping to its similar sounding affricated phone /dʒ/, resulting in our modern-day English letter j. In a parallel line of development, Spanish devoiced and backed Latin /j/ to become /h $\sim$ x/ (a range between the soft English "h" and the raspier sound found in German and Hebrew).

Interestingly, these differences become even more apparent when we survey Roman alphabets from around the globe. For example, about a dozen Turkic languages, following the lead of Ataturk in Turkey, adopted the Latin alphabet in the first half of the 20th century. These languages uniformly employ the j:/ʒ/ mapping of French, a reflection of that language's cultural prestige at the time. In contrast, when we examine the Latin writing systems of African languages, we find that the English j:/dʒ/ mapping predominates.[1] This tendency reflects the fact that many Latin-based African writing systems were developed by 19th century American missionaries and local governmental initiatives undertaken in the latter half of the 20th century, when English was on the rise as the global language.

Finally, when we examine the writing system of native American languages we find a similar divide. Mayan, Aztec, and most other south and central American languages have adopted variants of the Spanish mapping j:/h $\sim$ x/. In contrast, most North American languages have adapted the English mapping, sometimes adding the unvoiced variant j:/tʃ/. The alphabetic writing systems

[1] The /ʒ/ phone is typically represented by the diphone zh.

for these languages were developed through a combination of American missionaries, government initiatives, and north American linguists.

The contributions of this chapter are several-fold. First, we propose a new task: the automatic reconstruction and visualization of the history of the alphabet by means of unsupervised machine learning. Furthermore, we have collected and provide a novel data-set for this task, consisting of the grapheme-to-phone mappings for almost 300 languages with Latin alphabets. The language entries have been additionally collated with their large and small language families (e.g. Indo-European, Slavic), world regions (e.g. Europe, Africa) as well as latitude and longitude coordinates.

As an initial experiment, we applied unsupervised hierarchical clustering to our data-set, using a very simple distance function. This function measures two aspects of writing system similarity: The fraction of *shared phonemes* which are represented in the two writing systems by the *same graphemes*, and the overlap in non-standard graphemes (including digraphs, accent marks, and unusual symbols). We provide a qualitative analysis of the clustering results, showing that the induced cluster hierarchy lines up very well with our current historical knowledge. Finally, we provide a stand-alone software tool to allow readers to browse the clustering results along with our annotated analysis.

4.2 Data

In this section, we describe the data-set collected for and released with this thesis. Our data consists primarily of annotated mappings between the graphemes of a language, and their corresponding phonemic values. These mappings have been collected from and cross-checked against two main sources, (1) Omniglot: the online encyclopedia of writing systems and languages[2], and (2) the Wikipedia articles corresponding to specific languages.

We have collated these mappings with additional meta-data for each language, consisting of a large language family (e.g. Indo-European, Uralic), a smaller language family (e.g. Slavic, Finnic), a region (e.g. Africa, Europe), and latitude and longitude coordinates for the language.

[2]http://www.omniglot.com/writing/langalph.htm#latin

Large Families		Small Families		
Indo-European	68	Romance	24	Indo-European
Austronesian	34	Germanic	17	
Niger-Congo	32	Slavic	10	
Uralic	17	Celtic	6	
Altaic	14	Iranian	4	
Australian	11	Indic	2	
Algic	11	Baltic	2	
Afro-Asiatic	8	Albanian	2	
Uto-Aztecan	7	Tosk	1	
Nilo-Saharan	6	E. Cushitic	3	Afro-Asiatic
Na-Dene	6	Berber	3	
Muskogean	6	W. Chadic	1	
Mayan	6	Semitic	1	
Iroquoian	4	Finnic	10	Uralic
Siouan	3	Sami	5	
Sino-Tibetan	3	Ugric	1	
Eskimo-Aleut	3	Samoyedic	1	

Table 4.1: Large language families with at least three languages in our data-set, and small language families for three of the large language families.

These meta-data were collected from a variety of secondary sources, including the World Atlas of Language Structures[3], Ethnologue[4], and Wikipedia.

Our data consists of 296 languages with Latin-based alphabetic writing systems, including 84 European languages, 79 Native American languages, 53 African languages, 47 Asian languages, and 33 Pacific island languages. Besides geographical diversity, our languages also span a diverse

[3]http://wals.info/

[4]http://www.ethnologue.com/

set of language families. Table 4.1 shows the most common large language families; given the history of the Latin alphabet, Indo-European languages naturally dominate this category.

When we examine the grapheme-phone mappings in our data, we find 97 distinct graphemes that appear in at least 5% of our languages. These consist of the 26 standard letters of the Latin alphabet, 47 digraphs (pairs of letters that together represent a single sound), 20 accented letters, and four additional symbols apparently drawn from IPA notation. Table 4.2 lists the graphemes with the highest entropy in terms of their cross-lingual phoneme distributions. We note that it is not uncommon for these graphemes to represent multiple phonemes in a single language as well.

c		j		x		ch		q	
/k/	25	/j/	71	/x/	39	/tʃ/	53	/k/	37
/ts/	38	/dʒ/	51	/ks/	38	/x/	16	/q/	34
tʃ	32	/ʒ/	32	/ʃ/	21	/ʃ/	12	/ʔ/	5
/s/	29	/tʃ/	12	/χ/	6	/k/	12	/!/	5
/c/	9	/h/	12	/‖/	5	/tʃ^{h}/	8	/c/	3
/ʃ/	6	/x/	11	/z/	4	/c/	4	/k^{w}/	3

Table 4.2: The most ambiguous graphemes and their most frequent sounds.

4.3 Clustering of languages

In this initial stab at reconstructing the history of the alphabet, we have used the standard unsupervised technique of hierarchical agglomerative clustering. This method begins with each language in its own cluster, and recursively merges clusters in a greedy fashion, so as to minimize within-cluster pairwise distances. The end result is a complete dendrogram of the data, allowing the user to easily visualize the clustering at multiple layers of granularity.

The key challenge in our case was to design a distance function that will cluster languages, not based on how similar or related the languages are to one another, but only based only on the manner in which they employ the symbols of the Latin script. To achieve this effect, we decompose our

distance function into two linear components: $d(\ell_1, \ell_2) = d_1(\ell_1, \ell_2) + d_2(\ell_1, \ell_2)$, which we now describe in turn.

The first component measures the degree to which the languages represent their common phonemes using the same graphemes. Formally, let $g(ph, \ell)$ be a function which returns the set of graphemes which represent phoneme ph in langauge ℓ. we then compute:

$$d_1(\ell_1, \ell_2) = 1 - \frac{1}{N} \sum_{ph} JC\big(g(ph, \ell_1), g(ph, \ell_2)\big),$$

where the sum ranges over phonemes ph that are common to both languages, N is the number of such phonemes, and $JC(A, B) = \frac{|A \cap B|}{|A \cup B|}$ is the Jaccard coefficient, measuring the similarity of two sets. The advantage of this definition is that it is neutral with respect to the number of shared phonemes between the languages, only measuring their graphemic overlap.

Our second distance component measures the degree of overlap between "non-standard" graphemes employed by the two languages, i.e. accented letters, digraphs, and non-Latin characters. We simply take the Jaccard coefficient between the two sets of such graphemes:

$$d_2(\ell_1, \ell_2) = 1 - JC(U_1, U_2),$$

where U_1 is the set of non-standard graphemes in ℓ_1, and U_2 is the corresponding set for ℓ_2. As we shall see in the next section, this simple distance function yields results which line up remarkably well with the historical facts.

4.4 Analysis

Unfortunately, no gold-standard exists as of yet that would allow us to quantify the success of our clustering. Instead, we examine the clusters and qualitatively assess how well they match up with what we know of the history of the alphabet.

We start by noting that coherent clusters can be found at various levels of the tree. Interestingly, the height of the clusters seems to correspond quite nicely with the time at which the languages adopted the Latin alphabet. The highest clusters are dominated by European languages. A few steps down, we find a cluster consisting of Turkic and other central Asian languages. Near the

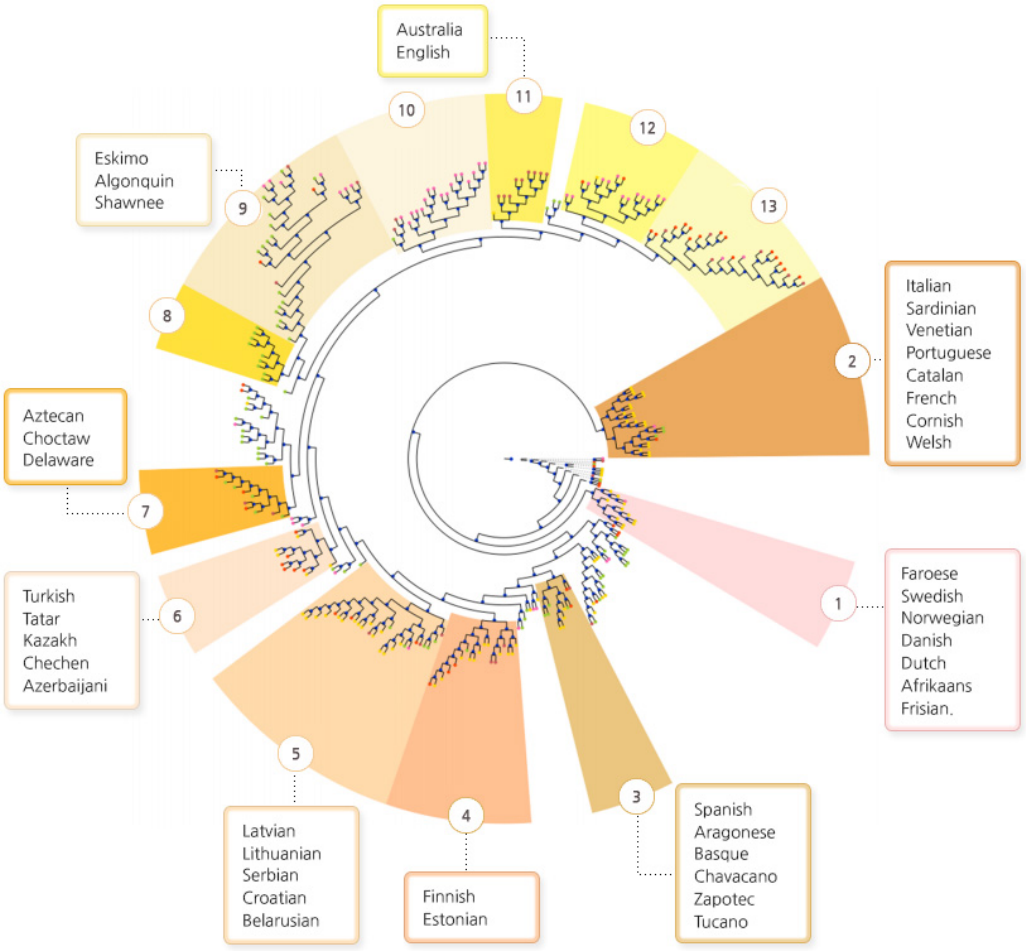

Figure 4.1: Hierarchical clustering of languages, based on alphabetic variations.

bottom of the tree, we see clusters of native American and African languages, and at the very bottom of the tree, we find a cluster consisting of southeast Asian and Pacific island languages.

We begin our guided tour at the top of the tree and work our way down. Figure 4.1 shows the induced tree structure with the various clusters that we discuss highlighted and labelled. We start with cluster **1**. Besides a few stray languages, this is the highest cluster in the tree and consists entirely of northern european languages from both the Indo-European and Uralic language

families. These include Norse languages such as Faroese, Swedish, Norwegian, and Danish, as well as northern Germanic languages such as Dutch, Afrikaans, and Frisian. Also included are two Uralic languages from the north.

Going down the tree, we encounter cluster **2**, which consists almost entirely of southern Indo-European languages. These include Italian, Sardinian, Venetian, Portuguese, Catalan, French, and three African and Caribbean French creoles. Cornish and Welsh belong to this cluster as well. The next cluster, number **3**, contains several more Romance languages including Spanish and Aragonese. Interestingly, this cluster also includes Basque (a non-IE language whose writing system has been influenced by Spanish), Chavacano (a Philippine creole based on Spanish), and several Latin American indigenous languages, such as Zapotec and Tucano.

Continuing down the tree, we find cluster **4**, which holds our primogenitor, Latin. It also includes some Germanic languages, several Romance languages, as well as some Uralic languages like Finnish and Estonian. Cluster **5** consists mostly of eastern European languages of several families, including Baltic languages (Latvian, Lithuanian), Slavic languages (Serbian, Croatian, and Belarusian), and two Albanian languages. Alongside these we have the northern European Sami languages (Uralic languages like Finnish and Estonian in the previous cluster).

In cluster **6**, we encounter our first big cluster of non-European languages. These consist entirely of central Asian languages, spanning several unrelated families, including Turkic languages (Turkish, Tatar, Kazakh), Chechen (a Caucasian language), and Azerbaijani (an Indo-European language of the Iranian branch). Although these three language families are unrelated, the languages in this cluster are in close proximity to one another, suggesting alphabetic influence.

The next clusters consist almost entirely of native American languages. Some clusters seem to mix North and South (Aztecan, Choctaw, and Delaware are all in cluster **7**), whereas others are demarcated geographically. Cluster **8** consists entirely of South American native languages, mostly of the Mayan family, whereas cluster **9** mixes North American indigenous languages (Eskimo, Algonquin, Shawnee) with some Bantoid and Eastern Cushitic languages of Africa. This grouping may reflect the joint influence of English, which we encounter on our next step down the tree.

Clusters **10** and **12** yields African languages, mostly from the Bantoid family (Zulu, Sotho). Sandwiched between these clusters is the interesting cluster **11**, containing native aboriginal languages of Australia, along with English. As Australia was a colony of Great Britain, it stands to reason that English would have an outsized influence here. Finally, the lowest cluster, number **13**, almost entirely consists of southeast Asian and Pacific island languages. These include the languages of the Philippines, Indonesia, Malaysia, and various other Austronesian languages and creoles.

4.5 Conclusion

We presented our work on the automatic reconstruction of the history of the Latin alphabet using unsupervised clustering. To begin this enterprise, collected a data-set consisting of nearly 300 languages, which we include as a resource for other researchers. Our clustering results, using only two simple distance features between alphabets, mirrors much of what we know about the history of the alphabet. As a resource for other researchers, we also include stand-alone visualization software to allow browsing of the clustering results.

Chapter 5

Phonetic Decipherment

In Chapter 2, we have developed the idea that *supervised* knowledge of some number of languages can help guide the unsupervised induction of linguistic structures, even in the absence of parallel texts. In Chapter 3, we also tackled the problem of unsupervised phonemic prediction for unknown languages by using textual regularities of known languages. However, we assumed that the target language was written in a known (Latin) alphabet, greatly reducing the difficulty of the prediction task. In this chapter, we assume no knowledge of any relationship between the writing system of the target language and known languages, other than that they are all alphabetic in nature. Specifically, we here focus on one aspect of language decipherment tasks: automatically identifying basic phonetic properties of letters in an unknown alphabetic writing system. For example, if a character is vowel or consonant and if it is non-nasal consonant or nasal. This work was originally published in [61].

This chapter is organized as follows: Section 5.1 gives a broad introduction to the chapter. Our argument is that we can successfully decode new languages by harnessing knowledge of the phonetic regularities encoded in known language vocabularies. We briefly describe our approach and summarize our experimental findings. Section 5.2 compares our approach to previous phonetic decipherment. Section 5.3.1 describes our model in great detail. Section 5.4 fully describes our inference procedures. Section 5.5.2 outlines our experiments on 503 world language in different alphabets and reports our results. Section 5.6 analysis our results. Section 5.7 completes the chapter with some concluding remarks.

5.1 Introduction

Over the past centuries, dozens of lost languages have been deciphered through the painstaking work of scholars, often after decades of slow progress and dead ends. Several major writing systems and languages remain undeciphered to this day.

In this chapter, we present a successful solution to one aspect of the decipherment puzzle: automatically identifying the basic phonetic properties of letters in an unknown alphabetic writing system. Our key idea is to use knowledge of the phonetic regularities encoded in known language vocabularies to automatically build a probabilistic model that can successfully decode new languages.

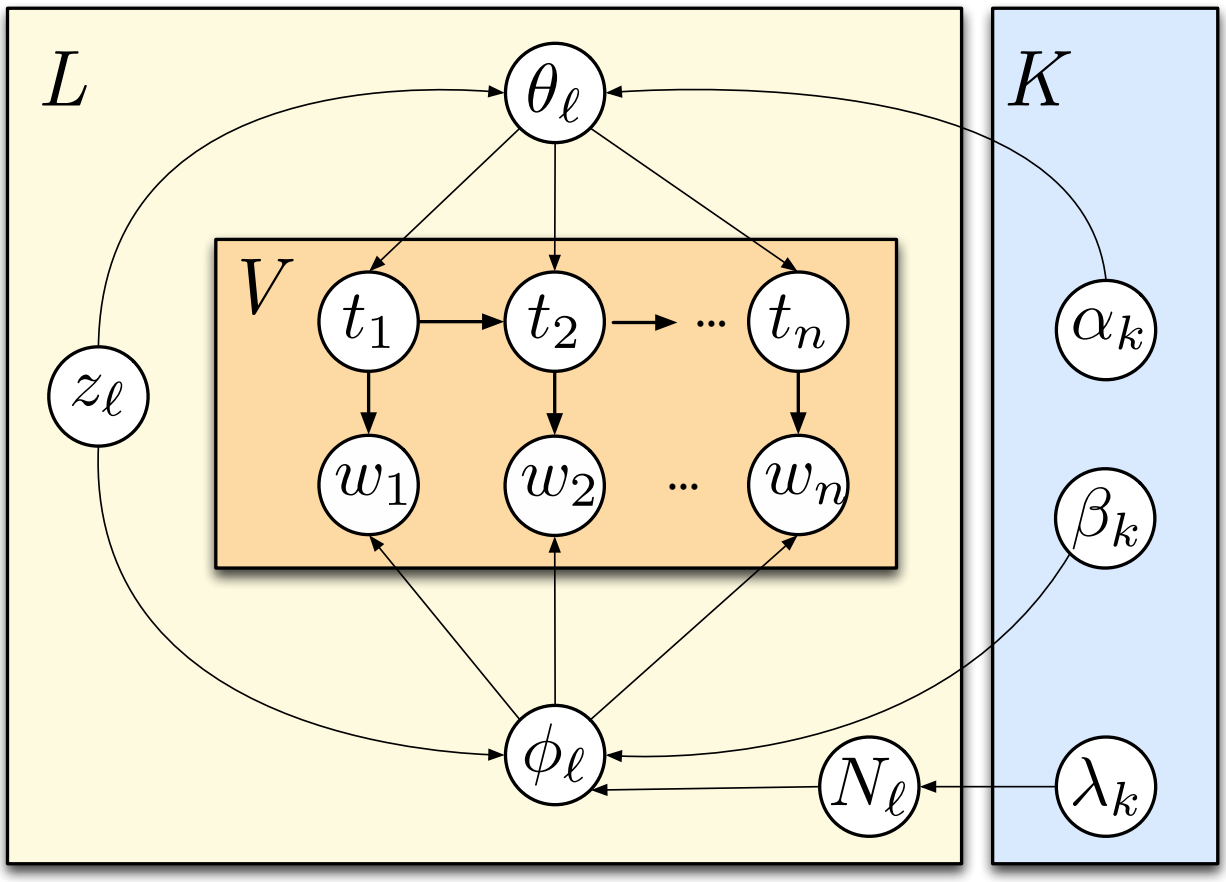

Figure 5.1: Graphical representation of our model. We have K language clusters, L languages, and V words in each language.

See Figure 5.1. Our approach adopts a classical Bayesian perspective. We assume that each language has an unobserved set of parameters explaining its observed vocabulary. We further assume that each language-specific set of parameters was itself drawn from an unobserved common prior, shared across a cluster of typologically related languages. In turn, each cluster derives its parameters from a universal prior common to all language groups. This approach allows us to mix together data from languages with various levels of observations and perform joint posterior inference over unobserved variables of interest.

At the bottom layer, our model assumes a language-specific data generating hidden Markov model (HMM) over the characters appearing in the language vocabulary. As letters typically represent just one major phonetic category such as consonant or vowel, we assume that each character type is constrained to be emitted by only one underlying tag category. Going one layer up, we posit that the HMM parameters are themselves drawn from biased priors representing a typologically coherent language grouping. By applying the model to a mix of observed and unobserved languages, these priors can be revealed and essentially guide the prediction for our target language.

We apply this approach to two decipherment tasks:

1. predicting whether individual characters in an unknown alphabet and language represent vowels or consonants, and

2. predicting whether individual characters in an unknown alphabet and language represent vowels, nasals, or non-nasal consonants.

For both tasks, our approach yields considerable success. We experiment with a data set consisting of vocabularies of 503 languages, written in a mix of Latin, Cyrillic, and Greek alphabets. In turn, we consider the writing system of each of these languages "unobserved" (i.e. we pretend to not know any phonetic properties of the characters) while treating the vocabularies of the remaining languages as fully observed with Consonant, Vowel, and Nasal tags on each of the letters.

On average, over these 503 leave-one-language-out scenarios, our model predicts consonant/vowel distinctions with 99% accuracy. In the more challenging task of vowel/nasal/non-nasal prediction, our model achieves average accuracy over 89%.

5.2 Related Work

The most direct precedent to the present work is a section in Knight et al. [62] on universal phonetic decipherment. They build a trigram HMM with three hidden states, corresponding to consonants, vowels, and spaces. As in our model, individual characters are treated as the observed emissions of the hidden states. In contrast to the present work, they allow letters to be emitted by multiple states.

Their experiments show that the HMM trained with expectation-maximization (EM) successfully clusters Spanish letters into consonants and vowels. They further design a more sophisticated finite-state model, based on linguistic universals regarding syllable structure and sonority. Experiments with the second model indicate that it can distinguish sonorous consonants (such as n, m, l, r) from non-sonorous consonants in Spanish. An advantage of the linguistically structured model is that its predictions do not require an additional mapping step from uninterpreted hidden states to linguistic categories, as they do with the HMM.

Both our model and experiments can be viewed as complementary to the work of Knight et al., while also extending it to hundreds of languages. We use the simple HMM with EM as our baseline. Instead of a linguistically designed model structure, we choose an empirical data-driven approach, allowing posterior inference over hundreds of known languages to guide the model's decisions for the unknown script and language.

In this sense, our model bears some similarity to the decipherment model of Snyder et al. [93], which used knowledge of a related language (Hebrew) in an elaborate Bayesian framework in order to decipher the ancient language of Ugaritic. While the aim of the present work is more modest (discovering very basic phonetic properties of letters) it is also more widely applicable, as we do not require the detailed analysis of a known related language.

Other recent work has employed a similar perspective for tying learning across languages. Naseem et al. [77] use a non-parametric Bayesian model over parallel text to jointly learn part-of-speech taggers across 8 languages, while Cohen and Smith [18] develop a shared logistic normal

prior to couple multilingual learning even in the absence of parallel text. In similar veins, Berg-Kirkpatrick and Klein [7] develop hierarchically tied grammar priors over languages within the same family, and Bouchard-Côté et al. [11] develop a probabilistic model of sound change using data from 637 Austronesian languages.

Finally, we note some similarities of our model to some ideas proposed in other contexts. We make the assumption that each observation type (letter) occurs with only one hidden state (consonant or vowel). Similar constraints have been developed for part-of-speech tagging [64; 16], and the power of type-based sampling has been demonstrated, even in the absence of explicit model constraints [67].

5.3 Model

Our generative Bayesian model over the observed vocabularies of hundreds of languages is presented in Figure 5.1 and its generative process is shown in Algorithms 1, 2, and 3. We present a running commentary on the generative process from the bottom up, starting with Algorithm 1.

5.3.1 Data Generation

Algorithm 1: Data Generation

for each language ℓ **do**

for each position i **do**

// transition to new tag token

$t_i|t_{i-1} \sim \text{Mult}(\theta_{\ell,t_{i-1},1}...\theta_{\ell,t_{i-1},T})$

// emit observation index token

$j|t_i \sim \text{Mult}(\phi_{\ell,t_i,1}...\phi_{\ell,t_i,N_{\ell,t_i}})$

// transcribe index token as character

$w_i \leftarrow \texttt{orth}(\ell, j, t_i)$

See Algorithm 1. At the data generation stage, our model resembles an HMM. At each time step i, a tag t_i is selected according to a language-specific transition distribution ϕ, indexed by the previous tag t_{i-1}. Note that in practice, we implemented a trigram version of the model, where the transition distribution is indexed by the previous *two* tags. However we here present the bigram version here for notational clarity. We assume that our tagset includes phonetic categories of interest (such as consonant, vowel, nasal, etc) as well as a special tag to represent the boundaries between words.

An observation index $j \in 1...N_{\ell,t_i}$ is then drawn from the language-specific emission distribution ϕ, indexed by the current tag t_i. N_{ℓ,t_i} denotes the number of observation types associated with tag t_i in language ℓ. Finally, we assume the existence of a deterministic function `orth` which maps each tag's observation indices to unique orthographic character symbols. This ensures that each observed character type corresponds to an observation index in exactly one tag category.

5.3.2 Language Generation

Algorithm 2: Language Generation

for each language ℓ **do**

// draw cluster assignment

cluster $z_\ell \sim \mathrm{Unif}[1...K]$

for each tag t **do**

// generate tag type-count

$N_{\ell,t} \sim \mathrm{Poiss}(\lambda_{z_\ell,t})$

// generate emission multinomial

$\phi_{\ell,t,1}...\phi_{\ell,t,N_{\ell,t}} \sim \mathrm{SymmDir}(\beta_{z_\ell,t})$

// generate transition multinomial

$\theta_{\ell,t,1}...\theta_{\ell,t,T} \sim \mathrm{Dir}(\alpha_{z_\ell,t,1}...\alpha_{z_\ell,t,T})$

See Algorithm 2. At the next stage up, we consider the generation of all language-specific parameters. This process begins by selecting a language cluster assignment z_ℓ uniformly. The language cluster provides priors over the HMM parameters. These priors include:

1. Poisson distributions over the number of observation types $N_{\ell,t}$ associated with tag t,
2. Dirichlet priors over transition distributions θ, and
3. Dirichlet priors over emission distributions ϕ.

For example, the cluster Poisson parameter over vowel observation types might be $\lambda = 9$ (indicating 9 vowel letters on average for the cluster), while the parameter over consonant observation types might be $\lambda = 20$ (indicating 20 consonant letters on average). These priors will be distinct for each language cluster and serve to characterize its general linguistic and typological properties.

We pause at this point to review the Dirichlet distribution in more detail. A k-dimensional Dirichlet with parameters $\alpha_1 ... \alpha_k$ defines a distribution over the $k-1$ simplex with the following density:

$$f(\theta_1 ... \theta_k | \alpha_1 ... \alpha_k) \propto \prod_i \theta_i^{\alpha_i - 1}$$

where $\alpha_i > 0$, $\theta_i > 0$, and $\sum_i \theta_i = 1$. The Dirichlet serves as the *conjugate prior* for the Multinomial, meaning that the posterior $\theta_1 ... \theta_k | X_1 ... X_n$ is again distributed as a Dirichlet (with updated parameters). It is instructive to reparameterize the Dirichlet with $k+1$ parameters:

$$f(\theta_1 ... \theta_k | \alpha_0, \alpha'_1 ... \alpha'_k) \propto \prod_i \theta_i^{\alpha_0 \alpha'_i - 1}$$

where $\alpha_0 = \sum_i \alpha_i$, and $\alpha'_i = \alpha_i / \alpha_0$. In this parameterization, we have $\mathbb{E}[\theta_i] = \alpha'_i$. In other words, the parameters α'_i give the mean of the distribution, and α_0 gives the *precision* of the distribution. For large $\alpha_0 \gg k$, the distribution is highly peaked around the mean (conversely, when $\alpha_0 \ll k$, the mean lies in a valley).

Thus, the Dirichlet parameters of a language cluster characterize both the average HMMs of individual languages within the cluster, as well as how much we expect the HMMs to vary from the mean. In the case of emission distributions, we assume symmetric Dirichlet priors — i.e.

one-parameter Dirichlets with densities given by $f(\theta_1...\theta_k|\beta) \propto \prod \theta_i^{(\beta-1)}$. This assumption is necessary, as we have no way to identify characters across languages in the decipherment scenario, and even the number of consonants and vowels (and thus multinomial/Dirichlet dimensions) can vary across the languages of a cluster. Thus, the mean of these Dirichlets will always be a uniform emission distribution. The single Dirichlet emission parameter per cluster will specify whether this mean is on a peak (large β) or in a valley (small β). In other words, it will control the expected *sparsity* of the resulting per-language emission multinomials.

In contrast, the transition Dirichlet parameters may be asymmetric, and thus very specific and informative. For example, one cluster may have the property that CCC consonant clusters are exceedingly rare across all its languages. This property would be expressed by a very small mean $\alpha'_{\text{CCC}} \ll 1$ but large precision α_0. Later we shall see examples of learned transition Dirichlet parameters.

5.3.3 Cluster Generation

Algorithm 3: Cluster Generation

for each cluster $k \in 1...K$ **do**

  for each tag $t \in 1...T$ **do**

    // emission Dirichlet parameter

    $\beta_{k,t} \sim \text{Unif}[0, 500]$

    // type-count Poisson parameter

    $\lambda_{k,t} \sim \text{Gamma}(g_1, g_2)$

    // transition Dirichlet parameters

    for each tag t' **do**

      $\alpha_{k,t,t'} \sim \text{Unif}[0, 500]$

See Algorithm 3. The generation of the cluster parameters defines the highest layer of priors for our model. As Dirichlets lack a standard conjugate prior, we simply use uniform priors over

the interval $[0, 500]$. For the cluster Poisson parameters, we use conjugate Gamma distributions with vague priors.[1]

5.4 Inference

We detail the inference procedure we followed to make predictions under our model. We run the procedure over data from 503 languages, assuming that all languages but one have observed character and tag sequences: $w_1, w_2, \ldots, t_1, t_2, \ldots$ Since each character type w is assumed to have a single tag category, this is equivalent to observing the character token sequence along with a character-type-to-tag mapping t_w. For the target language, we observe only character token sequence $w_1, w_2, \ldots$

We assume fixed and known parameter values only at the cluster generation level. Unobserved variables include (i) the cluster parameters α, β, λ, (ii) the cluster assignments $\mathbf{z}$, (iii) the per-language HMM parameters θ, ϕ for all languages, and (iv) for the target language, the tag tokens $t_1, t_2, \ldots$ — or equivalently the character-type-to-tag mappings t_w — along with the observation type-counts N_t.

5.4.1 Monte Carlo Approximation

Our goal in inference is to predict the most likely tag $t_{w,\ell}$ for each character type w in our target language ℓ according to the posterior:

$$f\left(t_{w,\ell}|\mathbf{w}, \mathbf{t}_{-\ell}\right) = \int f\left(\mathbf{t}_\ell, \mathbf{z}, \alpha, \beta|\mathbf{w}, \mathbf{t}_{-\ell}\right) d\,\Theta \tag{5.1}$$

where $\Theta = (\mathbf{t}_{-w,\ell}, \mathbf{z}, \alpha, \beta)$, $\mathbf{w}$ are the observed character sequences for all languages, $\mathbf{t}_{-\ell}$ are the character-to-tag mappings for the observed languages, $\mathbf{z}$ are the language-to-cluster assignments, and α and β are all the cluster-level transition and emission Dirichlet parameters.

[1] (1,19) for consonants, (1,10) for vowels, (0.2, 15) for nasals, and (1,16) for non-nasal consonants.

Sampling values $(\mathbf{t}_\ell, \mathbf{z}, \alpha, \beta)_{n=1}^{N}$ from the integrand in Equation 5.1 allows us to perform the standard Monte Carlo approximation:

$$f\left(t_{w,\ell} = t|\mathbf{w}, \mathbf{t}_{-\ell}\right) \approx N^{-1} \sum_{n=1}^{N} \mathbb{I}\left(t_{w,\ell} = t \text{ in sample } n\right) \tag{5.2}$$

To maximize the Monte Carlo posterior, we simply take the most commonly sampled tag value for character type w in language ℓ. Note that we leave out the language-level HMM parameters (θ, ϕ) as well as the cluster-level Poisson parameters λ from Equation 5.1 (and thus our sample space), as we can analytically integrate them out in our sampling equations.

5.4.2 Gibbs Sampling

To sample values $(\mathbf{t}_\ell, \mathbf{z}, \alpha, \beta)$ from their posterior (the integrand of Equation 5.1), we use Gibbs sampling, a Monte Carlo technique that constructs a Markov chain over a high-dimensional sample space by iteratively sampling each variable conditioned on the currently drawn sample values for the others, starting from a random initialization. The Markov chain converges to an equilibrium distribution which is in fact the desired joint density [41]. We now sketch the sampling equations for each of our sampled variables.

5.4.2.1 Sampling $t_{w,\ell}$

To sample the tag assignment to character w in language ℓ, we need to compute:

$$f\left(t_{w,\ell}|\mathbf{w}, \mathbf{t}_{-w,\ell}, \mathbf{t}_{-\ell}, \mathbf{z}, \alpha, \beta\right) \tag{5.3}$$

$$\propto f\left(\mathbf{w}_\ell, \mathbf{t}_\ell, N_\ell|\alpha_k, \beta_k, \mathbf{N}_{k-\ell}\right) \tag{5.4}$$

where N_ℓ are the types-per-tag counts implied by the mapping $\mathbf{t}_\ell$, k is the current cluster assignment for the target language ($z_\ell = k$), α_k and β_k are the cluster parameters, and $\mathbf{N}_{k-\ell}$ are the types-per-tag counts for all languages currently assigned to the cluster, *other* than language ℓ.

Applying the chain rule along with our model's conditional independence structure, we can further re-write Equation 5.4 as a product of three terms:

$$f(N_\ell|\mathbf{N}_{k-\ell}) \tag{5.5}$$

$$f(t_1, t_2, \ldots|\alpha_k) \tag{5.6}$$

$$f(w_1, w_2, \ldots|N_\ell, t_1, t_2, \ldots, \beta_k) \tag{5.7}$$

The first term is the posterior predictive distribution for the Poisson-Gamma compound distribution and is easy to derive. The second term is the tag transition predictive distribution given Dirichlet hyperparameters, yielding a familiar Polya urn scheme form. Removing terms that don't depend on the tag assignment $t_{\ell,w}$ gives us:

$$\frac{\prod_{t,t'}\left(\alpha_{k,t,t'} + n(t,t')\right)^{[n'(t,t')]}}{\prod_t\left(\sum_{t'}\alpha_{k,t,t'} + n(t)\right)^{[n'(t)]}}$$

where $n(t)$ and $n(t,t')$ are, respectively, unigram and bigram tag counts *excluding* those containing character w. Conversely, $n'(t)$ and $n'(t,t')$ are, respectively, unigram and bigram tag counts *only including* those containing character w. The notation $a^{[n]}$ denotes the ascending factorial: $a(a+1)\cdots(a+n-1)$. Finally, we tackle the third term, Equation 5.7, corresponding to the predictive distribution of emission observations given Dirichlet hyperparameters. Again, removing constant terms gives us:

$$\frac{\beta_{k,t}^{[n(w)]}}{\prod_{t'} N_{\ell,t'}\beta_{k,t'}^{[n(t')]}}$$

where $n(w)$ is the unigram count of character w, and $n(t')$ is the unigram count of tag t, over all characters tokens (including w).

5.4.2.2 Sampling $\alpha_{k,t,t'}$

To sample the Dirichlet hyperparameter for cluster k and transition $t \to t'$, we need to compute:

$$f(\alpha_{k,t,t'}|\mathbf{t},\mathbf{z}) \propto f(\mathbf{t},\mathbf{z}|\alpha_{z,t,t'}) = f(\mathbf{t}_k|\alpha_{z,t,t'})$$

where $\mathbf{t}_k$ are the tag sequences for all languages currently assigned to cluster k. This term is a predictive distribution of the multinomial-Dirichlet compound when the observations are grouped

into *multiple* multinomials all with the same prior. Rather than inefficiently computing a product of Polya urn schemes (with many repeated ascending factorials with the same base), we group common terms together and calculate:

$$\frac{\prod_{j=1}(\alpha_{k,t,t'}+k)^{n(j,k,t,t')}}{\prod_{j=1}(\sum_{t''}\alpha_{k,t,t''}+k)^{n(j,k,t)}}$$

where $n(j,k,t)$ and $n(j,k,t,t')$ are the numbers of languages currently assigned to cluster k which have *more than* j occurrences of unigram (t) and bigram (t,t'), respectively.

This gives us an efficient way to compute unnormalized posterior densities for α. However, we need to sample from these distributions, not just compute them. To do so, we turn to slice sampling [78], a simple yet effective auxiliary variable scheme for sampling values from unnormalized but otherwise computable densities.

The key idea is to supplement the variable x, distributed according to unnormalized density $\tilde{p}(x)$, with a second variable u with joint density defined as $p(x,u) \propto \mathbb{I}(u < \tilde{p}(x))$. It is easy to see that $\tilde{p}(x) \propto \int p(x,u)du$. We then iteratively sample $u|x$ and $x|u$, both of which are distributed uniformly across appropriately bounded intervals. Our implementation follows the pseudo-code given in Mackay [70].

5.4.2.3 Sampling $\beta_{k,t}$

To sample the Dirichlet hyperparameter for cluster k and tag t we need to compute:

$$f(\beta_{k,t}|\mathbf{t},\mathbf{w},\mathbf{z},\mathbf{N}) \propto f(\mathbf{w}|\mathbf{t},\mathbf{z},\beta_{k,t},\mathbf{N}) \propto f(\mathbf{w}_k|\mathbf{t}_k,\beta_{k,t},\mathbf{N}_k)$$

where, as before, $\mathbf{t}_k$ are the tag sequences for languages assigned to cluster k, $\mathbf{N}_k$ are the tag observation type-counts for languages assigned to the cluster, and likewise $\mathbf{w}_k$ are the character sequences of all languages in the cluster. Again, we have the predictive distribution of the multinomial-Dirichlet compound with multiple grouped observations. We can apply the same trick as above to group terms in the ascending factorials for efficient computation. As before, we use slice sampling for obtaining samples.

5.4.2.4 Sampling z_ℓ

Finally, we consider sampling the cluster assignment z_ℓ for each language ℓ. We calculate:

$$\begin{aligned} f(z_\ell = k|\mathbf{w}, \mathbf{t}, \mathbf{N}, \mathbf{z}_{-\ell}, \alpha, \beta) &\propto f(\mathbf{w}_\ell, \mathbf{t}_\ell, N_\ell|\alpha_k, \beta_k, \mathbf{N}_{k-\ell}) \\ &= f(N_\ell|\mathbf{N}_{k-\ell}) f(\mathbf{t}_\ell|\alpha_k) f(\mathbf{w}_\ell|\mathbf{t}_\ell, N_\ell, \beta_k) \end{aligned}$$

The three terms correspond to (1) a standard predictive distributions for the Poisson-gamma compound and (2) the standard predictive distributions for the transition and emission multinomial-Dirichlet compounds.

5.5 Experiments

To test the effectiveness of our model, we apply it to a corpus of 503 languages for two decipherment tasks. In both cases, we will assume no knowledge of our target language or its writing system, other than that it is alphabetic in nature. At the same time, we will assume basic phonetic knowledge of the writing systems of the other 502 languages. For our first task, we will predict whether each character type is a consonant or a vowel. In the second task, we further subdivide the consonants into two major categories: the nasal consonants, and the non-nasal consonants. Nasal consonants are known to be perceptually very salient and are unique in being high frequency consonants in all known languages.

5.5.1 Data

Our data is drawn from online electronic translations of the Bible (`http://www.bible.is`, `http://www.crosswire.org/index.jsp`, and `http://www.biblegateway.com`). We have identified translations covering 503 distinct languages employing alphabetic writing systems. Most of these languages (476) use variants of the Latin alphabet, a few (26) use Cyrillic, and one uses the Greek alphabet. As Table 5.1 indicates, the languages cover a very diverse set of families and geographic regions, with Niger-Congo languages being the largest represented family.[2] Of these

[2] In fact, the Niger-Congo grouping is often considered the largest language family in the world in terms of distinct member languages.

Language Family	#langs
Niger-Congo	114
Austronesian	67
Oto-Manguean	41
Indo-European	39
Mayan	34
Afro-Asiatic, Quechuan	17
Altaic, Uto-Aztecan	16
Trans-New Guinea	15
Nilo-Saharan	14
Sino-Tibetan	13
Tucanoan	9
Isolate	9
Creole	8
Chibchan	6
Maipurean, Tupian	5
Cariban, Nakh-Daghestanian, Totonacan, Uralic	4
Choco, Jivaroan, Mixe-Zoque	3
Austro-Asiatic, Guajiboan, Huavean, Jean, Paezan, Witotoan	2
Mapudungu, Puinavean, Uru-Chipaya, East Geelvink Bay, South-Central Papuan	1
Northwest Caucasian, Algic, Cahuapanan, Arauan, Barbacoan	1
Panoan, Eskimo-Aleut, West Papuan, South Bougainville, Nambiquaran	1
Jicaquean, Tequistlatecan, Aymaran, Lower Sepik-Ramu, Tor-Kwerba, Yanomam	1

Table 5.1: Language families in our data set.

languages, 30 are either language isolates, or sole members of their language family in our data set.

For our experiments, we extracted unique word types occurring at least 5 times from the downloaded Bible texts. We manually identified vowel, nasal, and non-nasal character types. Since the

letter "y" can frequently represent both a consonant and vowel, we exclude it from our evaluation. On average, the resulting vocabularies contain 2,388 unique words, with 19 consonant characters, two 2 nasal characters, and 9 vowels. We include the data as part of the work.

5.5.2 Baselines and Model Variants

As our baseline, we consider the trigram HMM model of Knight et al. [62], trained with EM. In all experiments, we run 10 random restarts of EM, and pick the prediction with highest likelihood. We map the induced tags to the gold-standard tag categories (1-1 mapping) in the way that maximizes accuracy.

We then consider three variants of our model. The simplest version, SYMM, disregards all information from other languages, using simple symmetric hyperparameters on the transition and emission Dirichlet priors (all hyperparameters set to 1). This allows us to assess the performance of our Gibbs sampling inference method for the type-based HMM, even in the absence of multilingual priors.

We next consider a variant of our model, MERGE, that assumes that *all* languages reside in a single cluster. This allows knowledge from the other languages to affect our tag posteriors in a generic, language-neutral way.

Finally, we consider the full version of our model, CLUST, with 20 language clusters. By allowing for the division of languages into smaller groupings, we hope to learn more specific parameters tailored for typologically coherent clusters of languages.

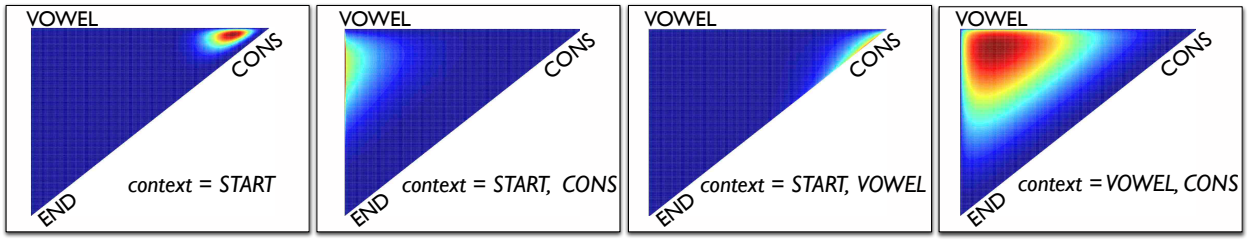

Figure 5.2: Inferred transition Dirichlet distributions for trigram MERGE model. Heat plots indicate Dirichlet densities over the 2-simplex.

Language	Model	Cons vs Vowel	Cons vs Vowel vs Nasal
All	EM	93.37	74.59
	SYMM	95.99	80.72
	MERGE	97.14	86.13
	CLUST	**98.85**	**89.37**
Isolates	EM	94.50	74.53
	SYMM	96.18	78.13
	MERGE	97.66	86.47
	CLUST	**98.55**	**89.07**
Non-Latin	EM	92.93	78.26
	SYMM	95.90	79.04
	MERGE	96.06	83.78
	CLUST	**97.03**	**85.79**

Table 5.2: Average accuracy for EM baseline and model variants across 503 languages.

5.5.3 Results

The results of our experiments are shown in Table 5.2. First panel (All) gives results on all languages. Second panel (Isolates) gives results for 30 isolate and singleton languages. Third panel (Non-Latin) gives results for 27 non-Latin alphabet languages such Cyrillic and Greek. In all cases, we report token-level accuracy (i.e. frequent characters count more than infrequent characters), and results are macro-averaged over the 503 languages. Variance across languages is quite low: the standard deviations are about 2 percentage points.

For the consonant vs. vowel prediction task, all tested models perform well. Our baseline, the EM-based HMM, achieves 93.4% accuracy. Simply using our Gibbs sampler with symmetric priors boosts the performance up to 96%. Performance increases again when we condition on other languages (MERGE), and we observe nearly 99% accuracy when allowing languages to cluster.

In the three-way nasal vs. non-nasal consonant vs. vowel prediction task, EM does not fare particularly well, only achieving 75% accuracy. As before, we see increasing performance gains for our model variants, culminating in almost 90% accuracy when the language clustering is used. The relatively weak performance of EM in this case should not be surprising: there is no *a priori* reason to expect any particular three-way classification to be the most salient clustering of letters from the perspective of EM. In contrast, our empirical multilingual approach allows the language-specific tag predictions to be guided by whatever values are set for the other, observed, languages.

We note that although a post-hoc mapping from inferred tags to true tags is necessary for both EM and SYMM, this is not the case for the final two variants of our model. Both MERGE and CLUST break symmetries over tags by way of the asymmetric posterior over transition Dirichlet parameters. Thus the reported accuracies are obtained without the need for any additional tag mappings.

Table 5.2 further breaks down results for languages without any other related language in our collection. These include 9 language isolates and 21 singleton languages acting as sole representatives of their families. In addition, we show results for the 27 languages which employ non-Latin alphabets (26 Cyrillic and one Greek). Both of these scenarios are likely to occur in cases of lost language decipherment. We see similar results and trends, with somewhat lower performance in both cases.

5.6 Analysis

To further compare our model to the EM baseline, we show confusion matrices for the three-way classification task in Figure 5.3. We can immediately see that EM had considerable difficulty making nasal predictions. Most true nasals (third row) are assigned to the regular consonant category, and apparently EM mostly used the additional tag as a way to further subcategorize vowels. In contrast, our model does fairly well with nasals: most actual nasals are assigned to the nasal category (third row), while the plurality of nasal predictions are indeed true nasals (third column).

Next we examine the transition Dirichlet hyperparameters learned by our model. For the MERGE model, we infer a posterior over parameters shared by all 503 languages in our data set.

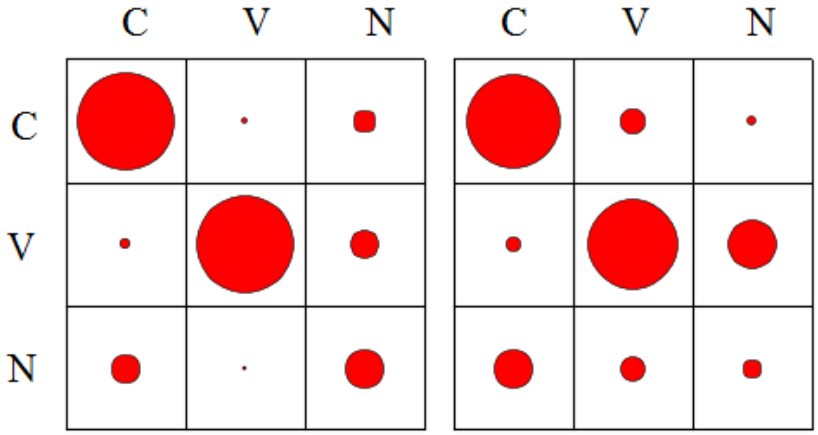

Figure 5.3: Confusion matrix for CLUST (left) and EM (right). Rows show true values, columns show predicted values. Size of blobs are proportional to counts.

Figure 5.2 shows MAP estimates of four of the Dirichlets governing transition probabilities from various contexts. As we can see, the learned hyperparameters yield highly asymmetric priors over transition distributions. Most languages like to start words with consonants, and after an initial consonant or vowel prefer to switch to the opposite category. In contrast, after a vowel-consonant sequence, languages can vary significantly in terms of the category favored next.

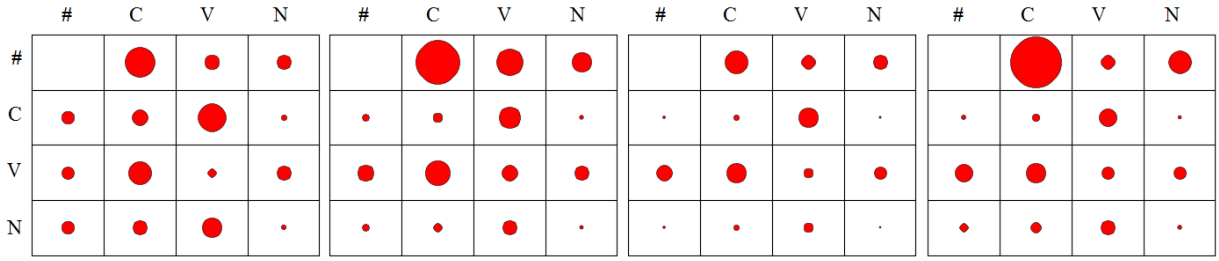

Figure 5.4: Inferred Dirichlet transition hyperparameters for bigram CLUST on three-way classification task with four latent clusters. Row gives starting state, column gives target state. Size of red blobs are proportional to magnitude of corresponding hyperparameters.

Figure 5.4 shows MAP transition Dirichlet hyperparameters of the CLUST model, when trained with a bigram HMM with four language clusters. Examining just the first row, we see that the

Language Family	Portion	#langs	Entropy
Indo-European	0.38	26	2.26
	0.24	41	3.19
	0.21	38	3.77
Quechuan	0.89	18	0.61
Mayan	0.64	33	1.70
Oto-Manguean	0.55	31	1.99
Maipurean	0.25	8	2.75
Tucanoan	0.2	45	3.98
Uto-Aztecan	0.4	25	2.85
Altaic	0.44	27	2.76
Niger-Congo	1	2	0.00
	0.78	23	1.26
	0.74	27	1.05
	0.68	22	1.22
	0.67	33	1.62
	0.5	18	2.21
	0.24	25	3.27
Austronesian	0.91	22	0.53
	0.71	21	1.51
	0.24	17	3.06

Table 5.3: Plurality language families across 20 clusters. The columns indicate portion of languages in the plurality family, number of languages, and entropy over families.

languages are partially grouped by their preference for the initial tag of words. All clusters favor languages which prefer initial consonants, though this preference is most weakly expressed in

cluster 3. In contrast, both clusters 2 and 4 have very dominant tendencies towards consonant-initial languages, but differ in the relative weight given to languages preferring either vowels or nasals initially.

Finally, we examine the relationship between the induced clusters and language families in Table 5.3, for the trigram consonant vs. vowel CLUST model with 20 clusters. We see that for about half the clusters, there is a majority language family, most often Niger-Congo. We also observe distinctive clusters devoted to Austronesian and Quechuan languages. The largest two clusters are rather indistinct, without any single language family achieving more than 24% of the total.

5.7 Conclusions

We presented a successful solution to one aspect of the decipherment task: the prediction of consonants and vowels for an unknown language's alphabet. Adopting a classical Bayesian perspective, we developed a model that performs posterior inference over hundreds of languages, thereby using knowledge of known languages to uncover general linguistic patterns of typologically coherent language clusters. We achieved average accuracy in the unsupervised consonant/vowel prediction task of over 99% across 476 languages. We further experimented with the task of distinguishing nasal from non-nasal consonants and report overall accuracy of over 90%.

Chapter 6

Part-of-speech Tagging for Low-resource Languages

So far in the previous chapters, we have examined the applications of cross-lingual supervised learning to several longstanding problems in NLP, including morphological analysis, grapheme-to-phoneme analysis, and phonetic decipherment. We have showed that cross-lingual supervised learning leads to significant performance gains over monolingual models. The previous tasks are word-level structural analyses, where the input, $x \in \mathcal{X}$, is a word and output, $y \in \mathcal{Y}$, is its label. In this chapter, we apply our methods to part-of-speech tagging, a sentence-level structural task, for hundreds of languages without ancillary resources such as tag dictionaries. We propose a method that leverages existing parallel data between the target language and a large set of resource-rich languages. Crucially, we use canonical correlation analysis (CCA) to induce latent representations to induce latent word representation that incorporate cross-genre distributional cues as well as projected tags from a full array of resource-rich languages.

This chapter is organized as follows: Section 6.1 gives a broad introduction to part-of-speech (POS) tagging for low resource languages . We argue that vastly multilingual nature of our parallel data can be used to build POS taggers for low resource languages without ancillary sources of information. Section 6.2 compares our approach to previous multilingual tagging approach. Section 6.3 describes our approach in great detail. Section 6.4 outlines setup and results of experiments. Section 6.5 completes the chapter with some concluding remarks.

6.1 Introduction

In this chapter, we address the challenge of creating accurate and robust part-of-speech taggers for resource-poor languages. We aim to apply our methods of cross-lingual supervised learning to hundreds, and potentially thousands, of languages with meager electronic resources. We do not assume the existence of a tag dictionary, or any prior knowledge of the target language. Instead, we base our methods entirely on the existence of parallel data between the target language and a set of resource-rich languages.

Fortunately, such parallel data exists for just about every written language, in the form of Bible translations. Around 2,500 languages have at least partial Bible translations, and somewhere between 500 and 1,000 languages have complete translations. We have collected such electronic Bible translations for 650 languages. Figure 6.1 breaks down the number of languages in our collection according to their token counts. The majority of our languages have at least 200,000 tokens of Bible translations.

While previous studies have addressed this general setting, they have typically assumed the existence of a partial tag dictionary as well as large quantities of non-parallel data in the target language. These assumptions are quite reasonable for the dozen most popular languages in the world, but are inadequate for the creation of a truly world-wide repository of NLP tools and linguistic data.

In fact, we argue that such ancillary sources of information are not really necessary once we take into account the vastly multilingual nature of our parallel data. Annotations projected from individual resource-rich languages are often noisy and unreliable, due to systematic differences between the languages in question as well as word alignment errors. We can thus think of these languages as very lazy and unreliable annotators of our target language. Despite their incompetence, as the number of such annotators increases, their combined efforts converge upon the truth, as idiosyncratic biases and random noises are washed away.

Our assumption throughout will be that we have in our possession a single multilingual corpus (the Bible) consisting of about 200,000 tokens for several hundred languages languages as well as

reasonably accurate POS taggers for about ten "resource-rich" languages. We will tag the Bible data for the resource-rich languages, word-align them to one another, and also word-align them to the remaining several hundred target languages.

Of course, our goal is not to produce a tagger restricted to the Biblical lexicon. We therefore assume a small unannotated monolingual sample of the target language in an entirely unrelated genre (e.g. newswire). We use this sample transductively to adapt our learned taggers from the Biblical genre. In our experiments, we use the CoNLL 2006 and 2007 shared-task test data for this purpose. Of course tagged data does not exist for truly resource-poor languages, so we evaluate our methodology on the resource-rich languages. Each such language takes a turn playing the role of the target language for testing purposes.

The goal of this chapter is to introduce a general "recipe" for successful cross-lingual induction of accurate taggers using meager resources. We face three major technical challenges:

- First, word alignments across languages are incomplete, and often do not preserve part-of-speech due to language differences.
- Second, when using multiple resource-rich languages, we need to resolve conflicting projections.
- Third, the parallel data at our disposal is of an idiosyncratic genre (the Bible), and we wish to induce a general-purpose tagger.

To address these challenges, we forgo the typical sequence-based learning technique of HMM's and CRF's and instead adopt an instance-learning approach using latent distributional features. To induce these features, we employ a new method using Canonical Correlation Analysis (CCA) to adapt tags of words of known languages to words of the target language. The method tackles each word from three views: (1) the token view (the word's context), (2) the type view (the word identity), and (3) the projected tags of neighboring words. We run CCA to induce latent continuous vector representations of each view that maximizes their correlations to one another. On the test data, a simple multi-class classifier then suffices to predict accurate tags, even for novel words.

This approach outperforms a state-of-the-art baseline [101] to achieve average tagging accuracy of 85% on newswire texts.

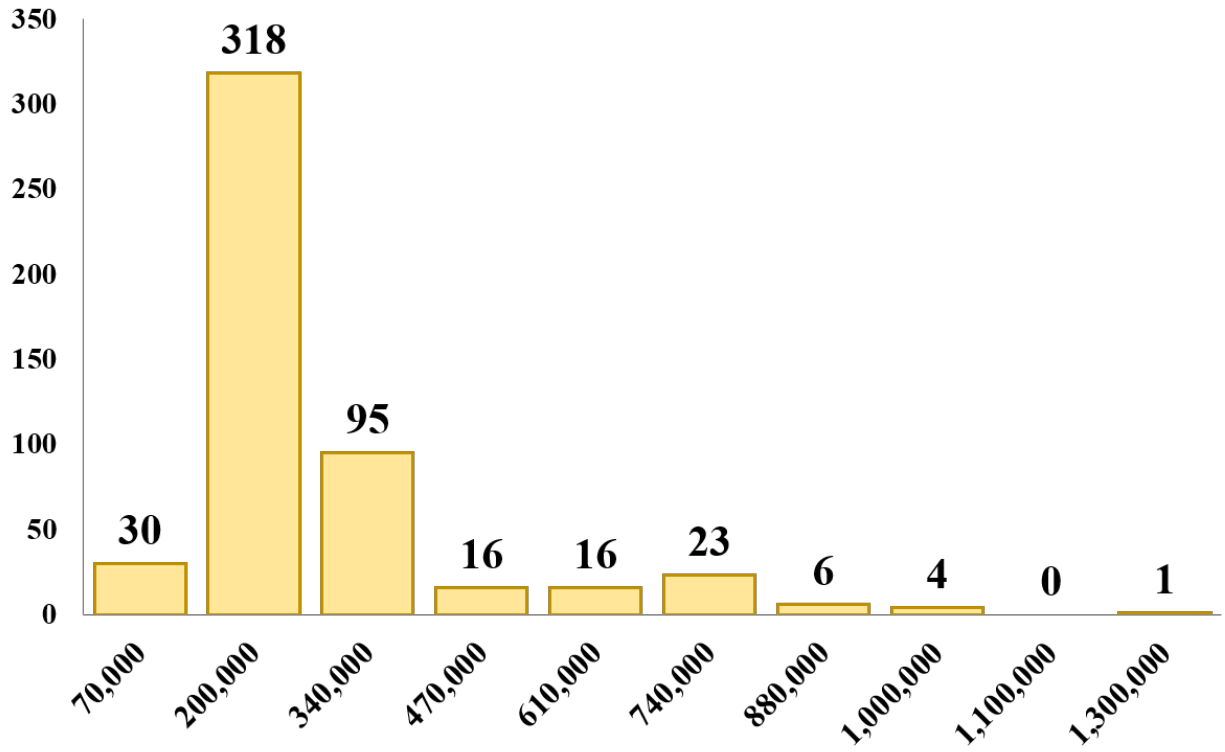

Figure 6.1: The breakdown of languages by the number of tokens in their available Bible translations. The horizontal axis gives the number of tokens, and the vertical axis gives the number of languages in each token range.

6.2 Related Work

We divide our survey of related work into several topics.

6.2.1 Multilingual Projection

The idea of projecting annotated resources across languages using parallel data was first proposed by Yarowsky et al. [106]. This early work recognized the noisy nature of automatic word

alignments and engineered smoothing and filtering methods to mitigate the effects of cross- lingual variation and alignment errors. More recent work in this vein has dealt with this by instead transferring information at the word type or model structure level, rather than on a token-by-token basis [26; 33]. Current state-of-the- art results for indirectly supervised POS performance use a combination of token constraints as well as type constraints mined from Wiktionary [65; 101]. As we argued in 6.1, the only widely available source of information for most low-resource languages is in fact their Bible translation. Perhaps surprisingly, our experiments show that this data source suffices to achieve state-of-the-art results.

Several previous authors have considered the advantage of using more than one resource-rich language to alleviate alignment noise. Fossum and Abney [39] found that using two source languages project-sources gave better results than simply using more data from one language. McDonald et al. [74] also found advantages to using multiple language sources for projecting parsing constraints. In more of an unsupervised context (but using small tag dictionaries), adding more languages to the mix has been shown to improve part-of-speech performance across all component languages [77].

6.2.2 Word Alignment

Most of the papers surveyed above rely on automatic word alignments to guide the cross-lingual transfer of information. Given our desire to use highly multilingual information to improve projection accuracy, the question of word alignment performance becomes crucial. Our hypothesis is that multiple- language projections are beneficial not only in weeding out random errors and idiosyncratic variations, but also in improving the linguistic consistency of the alignments themselves. Instead of simply aligning each source language to the target language in isolation, we will instead use a confidence model to synthesize information from multiple sources.

While we are not aware of any paper that has explored word alignment on a multilingual scale, there have been related efforts to *symmetrize* bilingual alignment models, using a variety of techniques ranging from modifications of EM [68], posterior-regularized objective function [40], and by considering relaxations of the hard combinatorial assignment problem [28].

6.2.3 Canonical Correlation Analysis (CCA)

Our method for generalizing the projections to unseen words and contexts is based on Canonical Correlation Analysis (CCA), a dimensionality reduction technique first introduced by Hotelling [50]. The key idea is to consider two groups of random variables with corresponding observations and to find linear subspaces with highest correlation between the two views. This can be seen as a kind of supervised version of Principal Components Analysis (PCA), where each view is providing supervision for the other. In fact, it can be shown that CCA directly generalizes both multiple linear regression and Fisher's Latent Discriminative Analysis (LDA) [42].

From a learning theory perspective, CCA is interesting in that it allows us to prove regret-based learning bounds that depend on the "intrinsic" dimensionality of the problem rather than the apparent dimensionality [54]. This seems especially relevant to natural language processing scenarios, where the ambient dimension is extremely large and sparse, but reductions to dense lower-dimensional spaces may preserve nearly all the relevant semantic and syntactic information. In fact, CCA has recently been adapted to learning latent word representations in an interesting way: by dividing each word position into a token view (which only sees surrounding context) and a type view (which only sees the word itself) and performing a CCA between these two views [29].

Our technique will extend this idea by additionally considering a third *projected tag* view. Crucially, it is this view which pushes the latent representations into coherent part-of-speech categories, allowing us to simply apply multi-class SVM for unseen words in our test set.

6.3 Tag projection from resource-rich languages

In this section, we describe two methods for incorporating transferred tags from resource-rich languages: sequence-based learning [101] and instance-based learning. In the former, the transferred tags are used to train a partially-observed CRF (PO-CRF) by maximizing the probability of a constrained lattice. In contrast, instance-based learning views each word token as an independent classification task, but uses latent distributional information gleaned from surrounding words as features.

6.3.1 A Sequence Learning Example of Partially Observed CRF (PO-CRF)

A first-order CRF parametrized by $\theta \in \mathbb{R}^d$ defines a conditional probability of a label sequence $y = y_1 \dots y_n$ given an observation sequence $x = x_1 \dots x_n$ as follows:

$$p_\theta(y|x) = \frac{\exp(\theta^\top \Phi(x, y))}{\sum_{y' \in \mathcal{Y}(x)} \exp(\theta^\top \Phi(x, y'))}$$

where $\mathcal{Y}(x)$ is the set of all possible label sequences for x and $\Phi(x, y) \in \mathbb{R}^d$ is a global feature function that decomposes into local feature functions $\Phi(x, y) = \sum_{j=1}^{n} \phi(x, j, y_{j-1}, y_j)$ by the first-order Markovian assumption. Given fully labeled sequences $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$, the standard training method is to find θ that maximizes the log likelihood of the label sequences under the model with l_2-regularization:

$$\theta^* = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_\theta(y^{(i)}|x^{(i)}) - \frac{\lambda}{2} ||\theta||^2$$

Unfortunately, in our problem we do not have fully labeled sequences. Instead, for each token x_j in sequence $x_1 \dots x_n$ we have transfered labels information from resource rich languages.

Täckström et al. [101] propose a different objective that allows training a CRF in this scenario. They define a constrained *lattice* $\mathcal{Y}(x, \tilde{y}) = \mathcal{Y}(x_1, \tilde{y}_1) \times \ldots \times \mathcal{Y}(x_n, \tilde{y}_n)$ where at each position j a set of allowed label types is given as:

$$\mathcal{Y}(x_j, \tilde{y}_j) = \begin{cases} \{\tilde{y}_j\} & \text{if } \tilde{y}_j \text{ is given} \\ \mathcal{Y}(x_j) & \text{otherwise} \end{cases}$$

Täckström et al. [101] define a conditional probability over label lattices for a given observation sequence x:

$$p_\theta(\mathcal{Y}(x, \tilde{y})|x) = \sum_{y \in \mathcal{Y}(x, \tilde{y})} p_\theta(y|x)$$

Given a label dictionary $\mathcal{Y}(x_j)$ for every token type x_j and training sequences $\{(x^{(i)}, \tilde{y}^{(i)})\}_{i=1}^{N}$ where $\tilde{y}^{(i)}$ is (possibly non-existent) transferred labels for $x^{(i)}$ and, the new training method is to find θ that maximizes the log likelihood of the label lattices:

$$\theta^* = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmax}} \sum_{i=1}^{N} \log p_\theta(\mathcal{Y}(x^{(i)}, \tilde{y}^{(i)})|x^{(i)}) - \frac{\lambda}{2} ||\theta||^2$$

Since this objective is non-convex, we find a local optimum with a gradient-based algorithm. The gradient of this objective at each example $(x^{(i)}, \tilde{y}^{(i)})$ takes an intuitive form:

$$\begin{aligned}
&\frac{\partial}{\partial\theta}\log p_\theta(\mathcal{Y}(x^{(i)},\tilde{y}^{(i)})|x^{(i)}) - \frac{\lambda}{2}\,||\theta||^2 \\
&= \sum_{y\in\mathcal{Y}(x^{(i)},\tilde{y})} p_\theta(y|x^{(i)})\Phi(x^{(i)},y) \\
&- \sum_{y\in\mathcal{Y}(x^{(i)})} p_\theta(y|x^{(i)})\Phi(x^{(i)},y) - \lambda\theta
\end{aligned}$$

This is the same as the standard CRF training except the first term where the gold features $\Phi(x^{(i)}, y^{(i)})$ are replaced by the expected value of features in the constrained lattice $\mathcal{Y}(x^{(i)}, \tilde{y})$.

An important distinction in our setting is that our token and type constraints are generated by only using the transferred tags whereas Täckström et al. [101] generate type constraints induced from Wiktionary. Our setting is more realistic for several reasons; 1) Wiktionary is not always available. 2) transferable information is not limited, but Wiktionary is (e.g., semantic role and named entity). 3) the imposed constraints are arguably more robust.

6.3.2 Cross-lingual Instance-based Learning

The proposed method for cross-lingual instance-based learning has three steps:

1. Select training tokens based on the confidence of the projected tag information.
2. Induce distributional features over these words that incorporates all projected tags.
3. Train a multi-class classifier with these induced features to make local predictions for individual tokens.

We will describe each step below.

6.3.2.1 Selecting training words

Since transferred tags are not always reliable, all words in the parallel data are not necessary helpful in training. Since this method trains on words instead of sequences, it is easy to discard

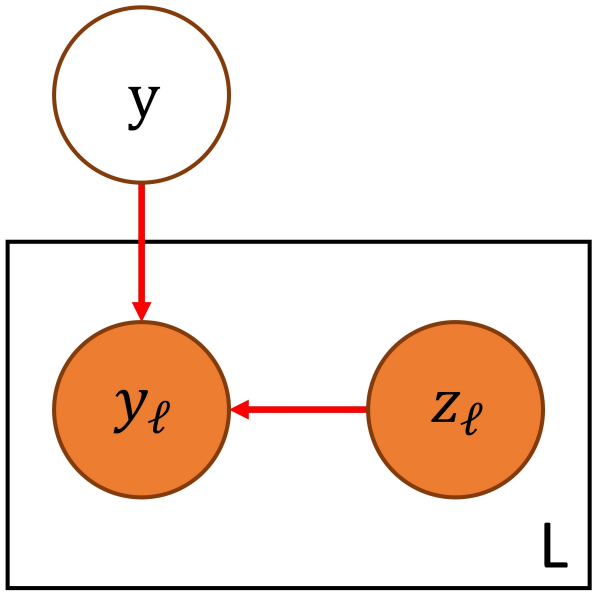

Figure 6.2: Graphical representation of the confidence model. Unobserved variable y denotes the true target-language tag for a token. Each of the L resource-rich languages displays a project of y, as y_ℓ, with an indicator variable z_ℓ determining the fidelity of the projection.

words which have unreliable or highly conflicting projections from different resource-rich languages.

To select our set of training tokens, we define a simple probability-based *confidence* model, illustrated in Figure 6.2. Suppose we have L resource-rich languages with alignments to the word in question. If the true tag is y, we assume that the projected tag for language ℓ will be identical to y with probability $1 - \epsilon_\ell$, where ϵ_ℓ is a language-specific corruption probability. With probability ϵ_ℓ, the projection will instead be chosen randomly (uniformly).

To make this explicit, we introduce a corruption indicator variable z_ℓ with:

$$P(z_\ell = 1) = \epsilon_\ell$$

Given z_ℓ, the probability of the projected tag y_ℓ is given by:

$$P(y_\ell|y, z_\ell) = \begin{cases} 1 & \text{if } z = 0 \text{ and } y = y_\ell, \\ \frac{1}{m} & \text{if } z = 1, \\ 0 & \text{otherwise.} \end{cases}$$

where m is the total number of possible tags. We can now compute a conditional distribution over the unknown tag y, marginalizing out the unknown corruption variables for each language:

$$p(y|y_1, \ldots, y_n) = \frac{\prod_{\ell=1}^{n} \left[\frac{\epsilon_\ell}{m} + (1 - \epsilon_\ell)\delta(y, y_\ell)\right]}{\frac{1}{m^{n-1}} \sum_{y'} \prod_{\ell=1}^{n} \left[\frac{\epsilon_\ell}{m} + (1 - \epsilon_\ell)\delta(y', y_\ell)\right]}$$

For simplicity, we simply set all ϵ_ℓ to 0.1 and use y as a training label when the conditional probability of the most likely value is greater than 0.9.

6.3.2.2 Inducing distributional features

In this section we discuss our approach for deriving latent distributional features. Canonical Correlation Analysis is a general method for inducing new representations for a pair of variables X and Y [49]. To derive word embeddings using CCA, a natural approach is to define X to represent a word and Y to represent the relevant information about a word, typically context words [29]. When they are defined as one-hot encodings, the CCA computation reduces to performing an SVD of the matrix Ω where each entry is

$$\Omega_{w,c} = \frac{\textbf{count}(w, c)}{\sqrt{\textbf{count}(w)\textbf{count}(c)}}$$

The resulting word representation is given by $U^\top X$ where U is a matrix of the scaled left singular vectors of Ω.

In our work, we use a slightly modified version of this definition by taking square-root of each count:

$$\sqrt{\Omega}_{w,c} = \frac{\textbf{count}(w, c)^{1/2}}{\sqrt{\textbf{count}(w)^{1/2}\textbf{count}(c)^{1/2}}}$$

This has an effect of stabilizing the variance of each term in the matrix, leading to a more efficient estimator. The square-root transformation also transforms the distribution of the count data to look more Gaussian [5]: since an interpretation of CCA is a latent-variable with normal distributions [4], it makes the data more suitable for CCA. It has been observed in past works (e.g., [30]) to significantly improve the quality of the resulting representations.

Feature Induction Algorithm We now describe our algorithm for inducing latent distributional features both on the multilingual parallel corpus, as well as the monolingual, newswire test data. This algorithm is described in detail in Figure 6.3. The key idea is to perform two CCA steps. The first step incorporates word-distributional information over both the multilingual corpus (the Bible) as well as the external domain monolingual corpus (CONLL data). This provides us with word representations that are general, and not overly specific to any single genre. However, it does not incorporate any projected tag information. We truncate this first SVD to the first 100 dimensions.

After this CCA step is performed, we then replace the words in the multilingual Bible data with their latent representations. We then perform a second CCA between these word representations and vectors representing the projected tags from all resource-rich languages. This step effectively *adapts* the first latent representation to the information contained in the tag projections. We truncate this second SVD to the first 50 dimensions.

We now have word embeddings that can be applied to any corpus, and are designed to maximize correlation both with typical surrounding word context, as well as typical projected tag context. These embeddings serve as our primary feature vectors for training the POS classifier (described in the next section). We concatenate this primary feature vector with the embeddings of the previous and subsequent words, in order to provide context-sensitive POS predictions.

6.3.2.3 Multi-class classifier

To train our POS tagger, we use a linear multi-class SVM [21]. It has a parameter $w_y \in \mathbb{R}^d$ for every tag $y \in \mathcal{T}$ and defines a linear score function $s(\mathbf{x}, j, y) := w_y^\top \Phi(\mathbf{x}, j)$. Given any sentence

Input:

- N "labeled" tokens in the Bible domain: word $w^{(i)} \in \mathcal{V}$, corresponding context $C(w^{(i)}) \subset \mathcal{V}$ and (projected) tag set $P^{(i)} \subset \mathcal{T}$ for $i = 1 \dots N$
- N' tokens in the test domain: word $v^{(i)} \in \mathcal{V}'$ and corresponding context $C(v^{(i)}) \subset \mathcal{V}'$ for $i = 1 \dots N'$
- CCA dimensions k_1, k_2

Output: embedding $\mathbf{e}(w) \in \mathbb{R}^{k_2}$ for each word $w \in \mathcal{V} \cup \mathcal{V}'$

1. Combine the observed tokens and their context from the Bible and test datasets:

$$\mathcal{W}_1 := \left(w : w \in (w^{(i)})_{i=1}^{N} \cup (v^{(i)})_{i=1}^{N'}\right)$$
$$\mathcal{C}_1 := \left(C(w) : w \in (w^{(i)})_{i=1}^{N} \cup (v^{(i)})_{i=1}^{N'}\right)$$

2. Perform rank-k_1 CCA on $(\mathcal{W}_1, \mathcal{C}_1)$ to derive a word projection matrix $\Phi_{\mathcal{W}_1}$ and a context projection matrix $\Phi_{\mathcal{C}_1}$.

3. Project all word examples in the Bible domain using $\Phi_{\mathcal{W}_1}$. Denote these projected words and the corresponding projected tag sets by

$$\mathcal{W}_2 := \left(\Phi_{\mathcal{W}_1}(w^{(i)}) : i = 1 \dots N\right)$$
$$\mathcal{P}_2 := \left(P^{(i)} : i = 1 \dots N\right)$$

4. Perform rank-k_2 CCA on $(\mathcal{W}_2, \mathcal{P}_2)$ to derive a word projection matrix $\Phi_{\mathcal{W}_2}$ and a tag projection matrix $\Phi_{\mathcal{P}_2}$.

5. Set the embedding $\mathbf{e}(w)$ for each word $w \in \mathcal{V} \cup \mathcal{V}'$

$$\mathbf{e}(w) = \Phi_{\mathcal{W}_2}(\Phi_{\mathcal{W}_1}(w))$$

Figure 6.3: Algorithm for deriving word vectors for the (unannotated) test data that use the projected tags in the Bible data.

$\mathbf{x}$ and a position j, it predicts $\mathrm{argmax}_{y \in \mathcal{T}} s(\mathbf{x}, j, y)$ as the tag of x_j. We use the implementation of [38]. We use the default hyperparameter configurations for training.

6.4 Experiments

6.4.1 Training Data

There are more than 4,000 living languages in the world, and one of the most prevalently translated books is the Bible. We now describe the Bible dataset we collected.

lang	percent of shared tokens	unseen words
BG	0.6386	15085
CS	0.5223	22730
DA	0.6675	11823
DE	0.6675	10052
ES	0.6868	13066
IT	0.6484	13040
NL	0.5676	7952
PT	0.6251	15210
AVG	0.6280	13619.75

Table 6.1: Percentage of shared tokens and the number of unseen words.

We first collect 893 bible volumes spanning several hundred languages that are freely available from three resources [1] and changed to UTF-8 format. The distribution of token in each bible in the unit of a language is in Figure 6.1.

Note that the Bible scripts are not exactly translated by sentences but by verses. We thus assume that each verse in a chapter has the same meaning if the number of verses is exactly same in a same chapter. In addition, we also assume that the whole chapters have the same meaning if the number of chapters in a book are exactly the same. In the same manner, we also assume the volumes that

[1] http://www.bible.is, http://www.crosswire.org, http://www.biblegateway.com

have the same number of chapters are the same. That is, their volume size should be as similar as possible with the respect to the number of verses, chapters, and books.

Based upon these assumptions, we choose the best translation in a language based on a comparison to a reference Bible, the Modern King James Version (MKJV) in English. We choose the translation for each language that best matches this reference version in terms of chapter and verse numbering.

There are other factors considered if there are more than one candidates satisfying this matching. We focus on the contents of the bible such as the publication time. For instance, 1599 Geneva Bible in English contains old vocabulary with different spelling systems, causing unexpected errors when tagged by POS annotation tools. Also, some of volumes such as Amplified Bible (AMP) contains extraneous comments on verses themselves, causing errors for word alignments.

After the choice of the best volume, we finally select the 10 resource rich languages. These languages are Bulgarian, Czech, Danish, German, English, French, Spanish, Italian, Dutch, and Portuguese. The two criteria to select resource rich languages are having i) the matched bible scripts both on the Old and New testament and ii) reliable parts-of-speech annotation tools. If these two requirements are satisfied, we can freely add more languages as resource rich languages in the future research. We use Hunpos tagger for Czech, Danish, German, English, and Portuguese, Treetagger for Bulgarian, Spanish, Italian, and Dutch, and Meltparser for French. The selected bible volume are POS-annotated by these taggers and the token accuracy of these taggers will be introduced with test data.

6.4.2 Test Data

We use CoNLL parts-of-speech tagged data as our test data. It consists of 5,000-6,000 hand-labeled tokens. The accuracy of each supervised tagger on this data is tested and reported in Table 6.2. Since there is no French tagged CoNLL data, we exclude French on testing but still use it in Training.

The tag definitions used in CoNLL data are not exactly matched the ones used in the taggers when converted to universal POS tags. For instance in Spanish, we initially follow mapping of

Lang	Tagger	Accuracy
BG	Treetagger	0.9909
CS	Hunpos	0.8969
DE	Hunpos	0.9855
EN	Hunpos	0.9854
ES	Treetagger	0.8785
IT	Treetagger	0.9059
NL	Treetagger	0.8781
PT	Hunpos	0.9770
AVG	-	0.9373

Table 6.2: Tagger accuracy on CoNLL data.

[82] for CoNLL data. The 'dp' tag for words *sus, su, mi* are mapped to DET but they are mapped to PRON in the bible data because of the Treetagger definitions. Whenever we find this kind of issues, we analyze them and choose the one of mappings for compatibility. For the 'dp' tag, we choose to map PRON.

6.4.3 Alignments

For experiments, we perform two kinds of alignments in our data sets; (i) the verse alignment and (ii) the word alignment. When the tagged bible volumes are prepared, we align verses across all resource rich languages. For verse alignments, we pre-process to remove extraneous information such as in-line reference (e.g. [REV 4:16]) and HTML tags. These alignments between two languages occurred only when volumes have the exact same number of chapters and verses. For instance, Mark must have 16 chapters and the first chapter of the Mark must have 45 verses in our criteria. The correct number of chapters and verses are pre-defined on MKJV volume, and the number of matched verses on each volume is greater than 30,500.

After performing verse alignments, we then perform word alignments. The quality of tags in resource poor languages is highly dependent on the quality of word alignments because parts-of-speech tags will be projected through this alignment path. First, we use GIZA++ for initial one-to-many alignments and we symmetrize by taking their intersection. This ensures that the resulting alignments are of high quality.

6.4.4 Results

In first experiment, we consider the state-of-the-art PO-CRF baseline. This model trains a partially observed CRF based on a single projected tag for each token. We experiment with different methods of choosing the projected tags. The result are shown in Table 6.3. The majority method is to choose the most common tag from the projected tags of the current token. We then experiment with taking the union of all projected tags (i.e. only constraining the lattice based on unanimity of the resource-rich languages). Finally, we considered choosing the high confidence tags, based on our confidence model. The confident tags are defined by a method described in Section 6.3.2.1 If this ratio is greater than 0.9, we assume that this token has high confidence. As the results indicate, this final method yielded the best tagging performance on the CONLL test data, achieving average accuracy of 83%.

In the remaining experiments we will adopt the confidence-based selection criterion for both the baseline as well as our method.

In order to isolate the errors due to projection mismatch versus domain variation, we first test both models on the Bible data itself. To do so, we assume that the tags produced by the test-language's supervised tagger are in fact the ground truth. This experiment allows us to compare to tag projection models using (1) PO-CRF and (2) CCA+SVM. Results are given in Table 6.4. Unsurprisingly, PO-CRF performs better on the multilingual corpus than on the CONLL data, due to the beneficial constraint of the projected tags. Perhaps surprisingly, the CCA+SVM method, which is a simple instance-based classifier using cleverly constructed features, outperforms the sequence labeller, achieving accuracy of nearly 87%.

	majority	union	confident
BG	0.8123	0.8167	0.8235
CS	0.8013	0.8094	0.8142
DA	0.8412	0.8497	0.8492
DE	0.8532	0.8611	0.8721
ES	0.8278	0.8345	0.8385
IT	0.8486	0.8445	0.8481
NL	0.7864	0.7876	0.7884
PT	0.8022	0.8081	0.8110
AVG	0.8216	0.8264	0.8306

Table 6.3: Baseline model CONLL performance depending on criterion for selecting tag projection.

	PO-CRF	CCA+SVM
BG	0.8450	0.8686
CS	0.8359	0.8442
DA	0.8727	0.8826
DE	0.8862	0.9025
ES	0.8523	0.8816
IT	0.8705	0.8911
NL	0.8115	0.8345
PT	0.8346	0.8410
AVG	0.8511	0.8683

Table 6.4: Accuracy on multilingual Bible data

In third experiment we use CoNLL test data and compare the PO-CRF models with different settings. This experiment is to show the effects of suffix and Brown cluster features on PO-CRF to

	1 lang (A)	9 lang (W)	9 langs (no S/C)	9 langs (A)
BG	0.7883	0.7144	0.8094	0.8478
CS	0.6601	0.5589	0.6535	0.7168
DA	0.7820	0.7765	0.8016	0.8227
DE	0.8323	0.6956	0.7589	0.8500
ES	0.7893	0.7608	0.8279	0.8665
IT	0.8444	0.7588	0.8136	0.8921
NL	0.7887	0.6825	0.7751	0.8214
PT	0.8476	0.7797	0.8464	0.9056
AVG	0.7916	0.7159	0.7858	0.8403

Table 6.5: Accuracy of the PO-CRF models on CoNLL data. A, W, no S/C means: all, word, all but no suffix and cluster features are used, respectively. Especially, all features include brown clustering IDs collected from more than 2 million line documents, making the setting unrealistic for resource-poor language.

relieve the unseen words issue. Additionally, we also show that the more projecting languages are included the better the results gets.

With just the word features, the averaged performance is 0.7159 and other indicator features (hyphen, digit, capitalized) increase the performance to 0.7858. Also note that the suffix and Brown clustering ID features increase the performance from 0.7858 to 0.8403. As reported, PO-CRF mitigates the adverse effects of the unseen word issues and almost meets the performance in the previous experiment (0.8511) of [101] by using these features.

In fourth and final experiment, we used the same features for PO-CRF, with Brown clusters induced on a more realistically sized corpus for a low resource language. We compare directly to our CCA+SVM model (which does not use Brown clustering features at all). We achieved 0.8093 on PO-CRF with all features and our corresponding model on CCA achieved about 0.8482,

shown in Table 6.6. As reported, our model outperforms the PO-CRF with the realistic settings for resource poor languages.

	PO-CRF 9 lang, 3k Brown	CCA+SVM
BG	0.8318	0.8815
CS	0.7635	0.7632
DA	0.7335	0.8911
DE	0.8296	0.8343
ES	0.8319	0.8713
IT	0.8451	0.8474
NL	0.7626	0.8145
PT	0.8768	0.8823
AVG	0.8093	0.8482

Table 6.6: Performances on our test data, CoNLL document.

6.5 Conclusions

We addressed the challenge of POS tagging low-resource languages. Our key idea is to use a massively multilingual corpus. Instead of relying on a single resource-rich language, we leverage the full array of currently available POS taggers. This removes alignment-mismatch noise and identifies a subset of words with highly confident tags. We then use a CCA procedure to induce latent feature representations across domains, incorporating word contexts as well as projected tags. We then train an SVM to predict tags.

Experimentally, we show that this procedure yields accuracy of about 85% for languages with nearly no resources available, beating a state-of-the-art partially observed CRF formulation. In the near future, this technique will enable us to release a suite of POS taggers for hundreds of low-resource languages.

Chapter 7

Optimal Data Selection

Until Chapter 6, we examined applications of cross-lingual supervised learning to NLP tasks of morphological analysis, grapheme-to-phoneme analysis, phonetic decipherment and part-of-speech tagging. We have showed that the suggested framework leads to significant performance gains over monolingual models. In all cases, we assumed that no annotated resources are available for the target language. However, for some tasks and languages, we can have the ability to create labeled data to train a model, but with limited time and budget. Parts of this work were originally published in [60].

Under this assumption, we introduce a novel method for rapid proto-typing and efficient construction of natural language understanding systems. Our key insight is that even small amounts of annotated data can yield powerful results when the examples to be labeled are chosen carefully. We develop two novel methods to achieve this goal, one based on matrix factorizations and the other based on a notion of feature coverage. We apply our techniques to four natural language tasks of pronunciation dictionary induction, part-of-speech prediction, named entity recognition, and semantic tagging of spoken queries. In all cases, our methods yield considerable performance improvements over randomly selected labeled examples. These results are robust across multiple languages, domains, and tasks, demonstrating that powerful natural language understanding systems can be built with far less annotation than previously thought.

This chapter is organized as follows: Section 7.1 gives a broad introduction to optimal data selection. We argue that optimal data set can yield a high performance supervised model. Section 7.2 describes background and previous approaches. Section 7.3 describe our two suggested methods

in great details. Section 7.4 describe results and analysis of experiments. Section 7.5 completes the chapter with some concluding remarks.

7.1 Introduction

Over the last 15 years, supervised statistical learning has become the dominant paradigm for building natural language technologies. While the accuracy of supervised models can be high, expertly annotated data sets exist for a small fraction of possible tasks, genres, and languages. The would-be tool builder is thus often faced with the prospect of annotating data, using crowd-sourcing or domain experts. With limited time and budget, the amount of data to be annotated might be small, especially in the prototyping stage, when the exact specification of the prediction task may still be in flux, and rapid prototypes are desired. When multilingual and multi-domain models are desired, the problem of annotation scarcity becomes even more severe.

In this chapter, we propose the idea that when the examples to be labeled are chosen carefully and intelligently, the amount of annotation required to achieve performance goals can be drastically reduced. To achieve this goal, we introduce the novel task of unsupervised optimal data set selection. Formally, given a large set $\mathcal{X}$ of n unlabeled examples, we must select a subset $\mathcal{S} \subset \mathcal{X}$ of size $k \ll n$ to be labeled. Our goal is to select such a subset which, when labeled, will yield a high performance supervised model over the entire data set $\mathcal{X}$. This task can be thought of as a zero-stage version of active learning: we must choose a single batch of examples to label, without the benefit of any prior labelled data points. This problem definition avoids the practical complexity of the active learning set-up (many iterations of learning and labeling), and ensures that the labeled examples are not tied to one particular model class or task, a well-known danger of active learning [90]. Alternatively, our methods may be used to create the initial seed set for the active learner.

Our initial testbed for optimal data set selection consists of four natural language understanding tasks: pronunciation dictionary induction, part-of-speech prediction, named entity recognition, and semantic tagging.

Pronunciation Dictionary Induction: In this task, we are given an out-of-vocabulary word, with the goal of predicting a sequence of phonemes corresponding to its pronunciation. For example, given a spelling of *calcium*, a model should predict its pronunciation, /kælsiəm/. As training data, we are given a pronunciation dictionary listing words alongside corresponding sequences of phones, representing canonical pronunciations of those words. Such dictionaries are used as the final bridge between written and spoken language technologies that span this divide, such as speech recognition, text-to-speech generation, and speech-to-speech language translation. These dictionaries are necessary: the pronunciation of words continues to evolve after their written form has been fixed, leading to a large number of rules and irregularities. While large pronunciation dictionaries of over 100,000 words exist for several major languages, these resources are entirely lacking for the majority of the world's languages. Our goal is to automatically select a small but optimal subset of words to be annotated with pronunciation data.

Part-of-speech Prediction: The goal is to predict the sequence of parts-of-speech tags of words forming a sentence. For example, given the sentence

Can you show me a map of Madison?

a model should determine that the word *Can* is a verb, rather than a noun, and the word *map* is a noun, rather than a verb. It is often a first step in the NLP pipeline, but part-of-speech taggers are only available for a handful of languages and domains. As training data, we are given a large pool of tagged example sentences, and we want to select the optimal subset of sentences to label.

Named Entity Recognition: The task of name entity recognition is to extract mentions of entities in texts and speech data. From the sentence

Show me a list of movies directed by Steven Spielberg.

a named entity recognizer should output that the words *Steven Spielberg* refer to the famous film director of that name. As for other NLP tasks, the problem with building such systems is the lack of annotated data across a range of languages and domains. Thus, we will intelligently sample a subset of example sentences to label with named entity mentions.

Semantic Tagging: In our final task, we consider the task of semantic tagging of sentences, also known as slot-filling. This task is an essential component of natural language understanding systems, allowing the machine to understand the desires of the speaker. For example, in the sentence

Give me a list of the 5 best selling books printed by Pearson in 2014.

the goal is return the products with the following characteristics:

- **type:** *book*
- **publisher:** *Pearson*
- **popularity:** $\geq$ *Rank 5*
- **year:** *2014*

Clearly, achieving high performance on this task is crucial for developing high performing interactive voice systems. As with the other tasks, the bottleneck is often obtaining labeled data for specific languages and domains.

The main intuition behind our approach is that the subset of selected data points should efficiently cover the range of phenomena most commonly observed across the pool of unlabeled examples. We consider two methods. The first comes from a line of research initiated by the numerical linear algebra community [45] and taken up by computer science theoreticians [13], with the name COLUMN SUBSET SELECTION PROBLEM (CSSP). Given a matrix A, the goal of CSSP is to select a subset of k columns whose span most closely captures the range of the full matrix. In particular, the matrix $\tilde{A}$ formed by orthogonally projecting A onto the k-dimensional space spanned by the selected columns should be a good approximation to A. By defining A^T to be our data matrix, whose rows correspond to words and whose columns correspond to features, we can apply the CSSP randomized algorithm of [13] on A to obtain a subset of k examples which best span the entire data space.

Our second approach is based on a notion of *feature coverage*. We assume that the benefit of seeing a feature f in a selected example bears some positive relationship to the frequency of f in

		Pronunciation Dictionary Induction	POS Tagging	Named Entity Recognition	Semantic Tagging
Domains	Fiction	n/a	✓		
	Wikipedia	n/a		✓	
	Apps	n/a			✓
	Games	n/a			✓
	Music	n/a			✓
	Movies	n/a			✓
Langs	Bulgarian		✓		
	Czech		✓	✓	
	English	✓	✓	✓	✓
	Estonian		✓		
	Dutch	✓			
	Farsi		✓		
	French	✓		✓	✓
	Frisian	✓			
	German	✓		✓	✓
	Hungarian		✓		
	Italian	✓		✓	
	Macedonian		✓		
	Norwegian	✓			
	Polish		✓	✓	
	Portuguese			✓	
	Romanian		✓		
	Russian			✓	
	Slovene		✓		
	Serbian		✓		
	Spanish	✓		✓	

Table 7.1: Summary of tasks, domains, and languages in our experimental evaluation.

the unlabeled pool. However, we further assume that the lion's share of benefit accrues the first few times that we label an example with feature f, with the marginal utility quickly tapering off as more such examples have been labeled. We formalize this notion and provide an exact greedy algorithm for selecting the k data points with maximal feature coverage.

To assess the benefit of these methods, we apply them to a suite of four natural language understanding tasks, spanning 20 languages and five language domains. See Table 7.1 for an overview of the tasks, domains, and languages. In all cases, we apply our methods to select optimal examples for labeling, and then train a state-of-the-art supervised model on the selected labeled examples. We assess the performance of the models as a function of the number of training examples. In all scenarios, our data set selection methods lead to significant increases in performance over randomly labeled examples. Average reductions in error range from 20% to 31% across the various tasks.

7.2 Background and Related Work

7.2.1 Data Set Selection and Active Learning

Eck et al [35] developed a method for training compact Machine Translation systems by selecting a subset of sentences with high n-gram coverage. Their selection criterion essentially corresponds to our feature coverage selection method using coverage function cov_2 (see Section 7.3.2). As our results will show, the use of a geometric feature discount (cov_3) provided better results in our task.

Otherwise, we are not aware of previous work proposing optimal data set selection as a general research problem. Of course, active learning strategies can be employed for this task by starting with a small random seed of examples and incrementally adding small batches. Unfortunately, this can lead to data-sets that are biased to work well for one particular class of models and task, but may otherwise perform worse than a random set of examples [90, Section 6.6]. Furthermore the active learning set-up can be prohibitively tedious and slow. To illustrate, Dwyer and Kondrak [34] used 190 iterations of active learning to arrive at 2,000 words. Each iteration involves bootstrapping 10 different samples, and training 10 corresponding learners. Thus, in total, the underlying

prediction model is trained 1,900 times. In contrast, our selection methods are fast, can select any number of data points in a single step, and are not tied to a particular prediction task or model. Furthermore, these methods can be combined with active learning in selecting the initial seed set.

7.2.2 Unsupervised Feature Selection

Finally, we note that CSSP and related spectral methods have been applied to the problem of *unsupervised feature selection* [100; 71; 103; 108; 12]. These methods are related to dimensionality reduction techniques such as Principal Components Analysis (PCA), but instead of truncating features in the eigenbasis representation (where each feature is a linear combination of all the original features), the goal is to remove dimensions in the standard basis, leading to a compact set of interpretable features. As long as the discarded features can be well approximated by a (linear) function of the selected features, the loss of information will be minimal.

Our first method for optimal data-set creation applies a randomized CSSP approach to the transpose of the data matrix, A^T. Equivalently, it selects the optimal k *rows* of A for embedding the full set of unlabeled examples. We use a recently developed randomized algorithm [13], and an underlying rank-revealing QR factorization [45].

7.2.3 Pronunciation Dictionary Induction

The task of pronunciation dictionary induction has been considered in a variety of frameworks, including neural networks [89], rule-based FSA's [55], and pronunciation by analogy [72]. Our goal here is not to compare these methods, so we focus on the probabilistic joint-sequence model of Bisani and Ney [8]. This model defines a joint distribution over a grapheme sequence $\boldsymbol{g} \in G^*$ and a phoneme sequence $\boldsymbol{\phi} \in \Phi^*$, by way of an unobserved *co-segmentation* sequence $\boldsymbol{q}$. Each co-segmentation unit q_i is called a *graphone* and consists of an aligned pair of zero or one graphemes and zero or one phonemes: $q_i \in G \cup \{\epsilon\} \times \Phi \cup \{\epsilon\}$.[1] The probability of a joint grapheme-phoneme

[1] The model generalizes easily to graphones consisting of more than one grapheme or phoneme, but in both [8] and our initial experiments we found that the 01-to-01 model always performed best.

sequence is then obtained by summing over all possible co-segmentations:

$$P(\boldsymbol{g}, \boldsymbol{\phi}) = \sum_{\boldsymbol{q} \in S(\boldsymbol{g}, \boldsymbol{\phi})} P(\boldsymbol{q})$$

where $S(\boldsymbol{g}, \boldsymbol{\phi})$ denotes the set of all graphone sequences which yield $\boldsymbol{g}$ and $\boldsymbol{\phi}$. The probability of a graphone sequence of length K is defined using an h-order Markov model with multinomial transitions:

$$P(\boldsymbol{q}) = \prod_{i=1}^{k+1} P(q_i | q_{i-h}, \ldots, q_{i-1})$$

where special start and end symbols are assumed for $q_{j<1}$ and q_{k+1}, respectively.

To deal with the unobserved co-segmentation sequences, the authors develop an EM training regime that avoids overfitting using a variety of smoothing and initialization techniques. Their model produces state-of-the-art or comparable accuracies across a wide range of languages and data sets.[2] We use the publicly available code provided by the authors.[3] In all our experiments we set $h = 4$ (i.e. a 5-gram model), as we found that accuracy tended to be flat for $h > 4$.

7.2.4 Active Learning for Pronunciation Dictionary Induction

Perhaps most closely related to our work are the papers of Kominek and Black [63] and Dwyer and Kondrak [34], both of which use active learning to efficiently bootstrap pronunciation dictionaries. In the former, the authors develop an active learning word selection strategy for inducing pronunciation rules. In fact, their greedy n-gram selection strategy shares some of the some intuition as our second data set selection method, but they were unable to achieve any accuracy gains over randomly selected words without active learning.

Dwyer and Kondrak use a Query-by-Bagging active learning strategy over decision tree learners. They find that their active learning strategy produces higher accuracy across 5 of the 6 languages that they explored (English being the exception). They extract further performance gains through various refinements to their model. Even so, we found that the Bisani and Ney grapheme-to-phoneme (G2P) model [8] always achieved higher accuracy, even when trained on random

[2] We note that the discriminative model of Jiampojamarn and Kondrak [53] outperforms the Bisani and Ney model by an average of about 0.75 percentage points across five data sets.

[3] `http://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html`

words. Furthermore, the relative gains that we observe using our optimal data set selection strategies (without any active learning) are much larger than the relative gains of active learning found in their study.

7.3 Two Methods for Optimal Data Set Selection

In this section we detail our two proposed methods for optimal data set selection. The key intuition is that we would like to pick a subset of data points which broadly and efficiently cover the features of the full range of data points. We assume a large pool $\mathcal{X}$ of n unlabeled examples, and our goal is to select a subset $\mathcal{S} \subset \mathcal{X}$ of size $k \ll n$ for labeling. We assume that each data point $x \in \mathcal{X}$ is a vector of m feature values. Our first method applies to any real or complex feature space, while our second method is specialized for binary features. We will use the $(n \times m)$ matrix A to denote our unlabeled data: each row is a data point and each column is a feature. In all our experiments, we used the presence (1) or absence (0) of each character 4-gram as our set of features.

7.3.1 Method 1: Row Subset Selection

To motivate this method, first consider the task of finding a rank k approximation to the data matrix A. The SVD decomposition yields:

$$A = U\Sigma V^T$$

- U is $(n \times n)$ orthogonal and its columns form the eigenvectors of AA^T
- V is $(m \times m)$ orthogonal and its columns form the eigenvectors of A^TA
- Σ is $(n \times m)$ diagonal, and its diagonal entries are the singular values of A (the square roots of the eigenvalues of both AA^T and A^TA).

To obtain a rank k approximation to A, we start by rewriting the SVD decomposition as a sum:

$$A = \sum_{i=1}^{\rho} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \tag{7.1}$$

where $\rho = \min(m, n)$, σ_i is the ith diagonal entry of Σ, $\mathbf{u}_i$ is the ith column of U, and $\mathbf{v}_i$ is the ith column of V. To obtain a rank k approximation to A, we simply truncate the sum in equation 7.1 to its first k terms, yielding A_k. To evaluate the quality of this approximation, we can measure the Frobenius norm of the residual matrix $||A - A_k||_F$.[4] The Eckart-Young theorem [36] states that A_k is optimal in the following sense:

$$A_k = \underset{\tilde{A} \ s.t. \ \text{rank}(\tilde{A})=k}{\text{argmin}} ||A - \tilde{A}||_F \tag{7.2}$$

In other words, truncated SVD gives the best rank k approximation to A in terms of minimizing the Frobenius norm of the residual matrix. In CSSP, the goal is similar, with the added constraint that the approximation to A must be obtained by projecting onto the subspace spanned by a k-subset of the original rows of A.[5] Formally, the goal is to produce a $(k \times m)$ matrix S formed from rows of A, such that

$$||A - AS^+S||_F \tag{7.3}$$

is minimized over all $\binom{n}{k}$ possible choices for S. Here S^+ is the $(m \times k)$ Moore-Penrose pseudo-inverse of S, and S^+S gives the orthogonal projector onto the rowspace of S. In other words, our goal is to select k data points which serve as a good approximate basis for *all* the data points. Since AS^+S can be at most rank k, the constraint considered here is stricter than that of Equation 7.1, so the truncated SVD A_k gives a lower bound on the residual.

Boutsidis et al [13] develop a randomized algorithm that produces a submatrix S (consisting of k rows of A) which, with high probability, achieves a residual bound of:

$$||A - AS^+S||_F \leq O(k\sqrt{\log k})||A - A_k||_F \tag{7.4}$$

in running time $O(\min\{mn^2, m^2n\})$. The algorithm proceeds in three steps: first by computing the SVD of A, then by randomly sampling $O(k \log k)$ rows of A with importance weights carefully computed from the SVD, and then applying a deterministic rank-revealing QR factorization [45]

[4]The Frobenius norm $||M||_F$ is defined as the entry-wise L_2 norm: $\sqrt{\sum_{i,j} m_{ij}^2}$

[5]Though usually framed in terms of column selection, we switch to row selection here as our goal is to select data points rather than features.

to select k of the sampled rows. To give some intuition, we now provide some background on rank revealing factorizations.

7.3.1.1 Rank revealing QR / LQ (RRQR)

Every real $(n \times m)$ matrix can be factored as $A = LQ$, where Q is $(m \times m)$ orthogonal and L is $(n \times m)$ lower triangular.[6] It is important to notice that in this triangular factorization, each successive row of A introduces exactly one new basis vector from Q. We can thus represent row i as a linear combination of the first $i-1$ rows along with the ith row of Q.

A *rank-revealing* factorization is one which displays the numerical rank of the matrix – defined to be the singular value index r such that

$$\sigma_r \gg \sigma_{r+1} = O(\epsilon)$$

for machine precision ϵ. In the case of the LQ factorization, our goal is to order the rows of A such that each successive row has decreasing representational importance as a basis for the future rows. More formally, If there exists a row permutation Π such that ΠA has a triangular factorization $\Pi A = LQ$ with $L = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}$, where the smallest singular value of L_{11} is much greater than the spectral norm of L_{22}, which is itself almost zero:

$$\sigma_{min}(L_{11}) \gg ||L_{22}||_2 = O(\epsilon)$$

then we say that $\Pi A = LQ$ is a *rank-revealing LQ factorization*. Both L_{11} and L_{22} will be lower triangular matrices and if L_{11} is $(r \times r)$ then A has numerical rank r [48].

7.3.1.2 Implementation

In our implementation of the CSSP algorithm, we first prune away features that appear in fewer than 3 examples, then compute the SVD of the pruned data matrix using the PROPACK package,[7] which efficiently handles sparse matrixes. After sampling $k \log k$ examples from A (with sampling

[6] We replace the standard upper triangular QR factorization with an equivalent lower triangular factorization LQ to focus intuition on the rowspace of A.

[7] `http://soi.stanford.edu/~rmunk/PROPACK/`

weights calculated from the top-k singular vectors), we form a submatrix B consisting of the sampled examples. We then use the RRQR implementation from ACM Algorithm 782 [9] (routine DGEQPX) to compute $\Pi B = LQ$. We finally select the first k rows of ΠB as our optimal data set. Even for our largest data sets this entire procedure runs in less than an hour on a 3.4Ghz quad-core i7 desktop with 32 GB of RAM.

7.3.2 Method 2: Feature Coverage Maximization

In our previous approach, we adopted a general method for approximating a matrix with a subset of rows (or columns). Here we develop a novel objective function with the specific aim of optimal data set selection. Our key assumption is that the benefit of seeing a new feature f in a selected data point bears a positive relationship to the frequency of f in the unlabeled pool of words. However, we further assume that the lion's share of benefit accrues quickly, with the marginal utility quickly tapering off as we label more and more examples with feature f. Note that for this method, we assume a boolean feature space.

To formalize this intuition, we will define the *coverage* of a selected $(k \times m)$ submatrix S consisting of rows of A, with respect to a feature index j. For illustration purposes, we will list three alternative definitions:

$$\text{cov}_1(S; j) = ||\mathbf{s}_j||_1 \tag{7.5}$$

$$\text{cov}_2(S; j) = ||\mathbf{a}_j||_1 \mathbb{I}\big(||\mathbf{s}_j||_1 > 0\big) \tag{7.6}$$

$$\text{cov}_3(S; j) = ||\mathbf{a}_j||_1 - \frac{||\mathbf{a}_j||_1}{\eta^{||\mathbf{s}_j||_1}} \mathbb{I}\big(||\mathbf{s}_j||_1 < ||\mathbf{a}_j||_1\big) \tag{7.7}$$

In all cases, $\mathbf{s}_j$ refers the j^{th} column of S, $\mathbf{a}_j$ refers the j^{th} column of A, $\mathbb{I}(\cdot)$ is a 0-1 indicator function, and η is a scalar discount factor.[8]

Figure 7.1 provides an intuitive explanation of these functions: cov_1 simply counts the number of selected data points with boolean feature j. Thus, full coverage ($||\mathbf{a}_j||$: the entire number of data

[8] Chosen to be 5 in all our experiments. We experimented with several values between 2 and 10, without significant differences in results.

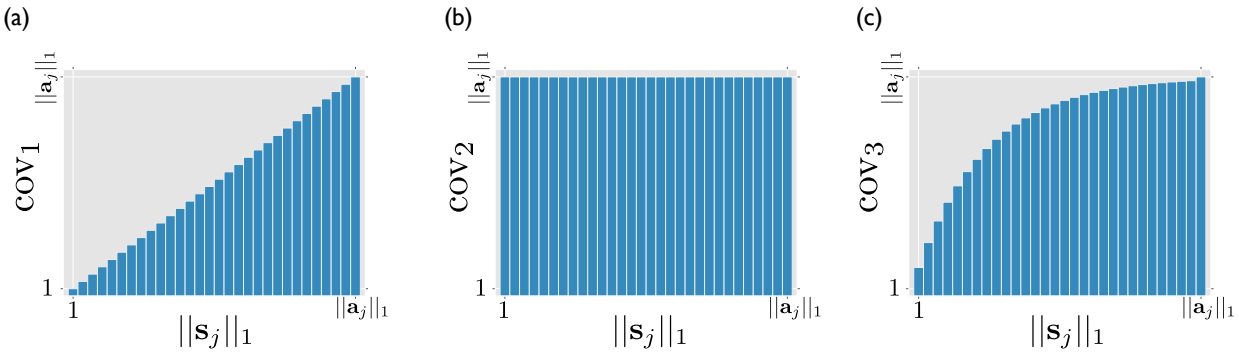

Figure 7.1: Various versions of the feature coverage function. Panel (a) shows cov_1 (Equation 7.5). Panel (b) shows cov_2 (Equation 7.6). Panel (c) shows cov_3 (Equation 7.7) with discount factor $\eta = 1.2$.

points with the feature) is only achieved when *all* data points with the feature are selected. cov_2 lies at the opposite extreme. Even a single selected data point with feature j triggers coverage of the entire feature. Finally, cov_3 is designed so that the coverage scales monotonically as additional data points with feature j are selected. The first selected data point will capture all but $\frac{1}{\eta}$ of the total coverage, and each further selected data point will capture all but $\frac{1}{\eta}$ of whatever coverage remains. Essentially, the coverage for a feature scales as a geometric series in the number of selected examples having that feature.

To ensure that the total coverage ($||\mathbf{a}_j||_1$) is achieved when all the data points are selected, we add an indicator function for the case of $||\mathbf{c}_j||_1 = ||\mathbf{a}_j||_1$.[9]

Setting our feature coverage function to cov_3, we can now define the overall feature coverage of the selected points as:

$$\textbf{coverage}(S) = \frac{1}{||A||_1} \sum_j \text{cov}_3(S; j) \tag{7.8}$$

where $||A||_1$ is the L_1 entrywise matrix norm, $\sum_{i,j} |A_{ij}|$, which ensures that $0 \leq \textbf{coverage}(S) \leq 1$ with equality only achieved when $S = A$, i.e. when all data points have been selected.

We provide a brief sketch of our optimization algorithm: To pick the subset S of k examples which optimizes Equation 7.8, we incrementally build optimal subsets $S' \subset S$ of size $k' < k$. At

[9]Otherwise, the geometric coverage function would converge to $||\mathbf{a}_j||$ only as $||\mathbf{c}_j|| \to \infty$.

each stage, we keep track of the unclaimed coverage associated with each feature j:

$$\text{unclaimed}(j) = ||\mathbf{a}_j||_1 - \text{cov}_3(S'; j)$$

To add a new example, we scan through the pool of remaining words, and calculate the additional coverage that selecting word w would achieve:

$$\Delta(w) = \sum_{\text{feature } j \text{ in } w} \text{unclaimed}(j) \left(\frac{\eta - 1}{\eta}\right)$$

We greedily select the example which adds the most coverage, remove it from the pool, and update the unclaimed feature coverages. It is easy to show that this greedy algorithm is globally optimal.

7.4 Experiments and Analysis

To test the effectiveness of the two proposed data set selection methods, we conduct experiments on a suite of four natural language understanding tasks, ranging over 20 languages and six language domains. See Table 7.1 for an overview of the tasks, domains, and languages. In all cases, we use our methods to select a subset of examples for labeling, and then train a state-of-the-art supervised model over the selected examples. We assess the accuracy of the models as a function of the number of training examples, and compare the performance to models trained over an equal number of randomly selected data points.

7.4.1 Task 1: Pronunciation Dictionary Induction

We conduct pronunciation dictionary induction experiments across 8 languages: Dutch, English, French, Frisian, German, Italian, Norwegian, and Spanish. The data was obtained from the PASCAL Letter-to-Phoneme Conversion Challenge,[10] and was processed to match the setup of Dwyer and Kondrak [34]. The data comes from a range of sources, including CELEX for Dutch and German [3], BRULEX for French [75], CMUDict for English,[11] the Italian Festival Dictionary [20], as well as pronunciation dictionaries for Spanish, Norwegian, and Frisian (original provenance not clear).

[10] `http://pascallin.ecs.soton.ac.uk/Challenges/PRONALSYL/`

[11] `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

As Table 7.2 shows, the size of the dictionaries ranges from 31,491 words (Spanish) up to 116,211 words (Dutch). We follow the PASCAL challenge training and test folds, treating the training set as our pool of words to be selected for labeling.

Language	Training	Test	Total
Dutch	11,622	104,589	116,211
English	11209	100891	112100
French	2,748	24,721	27,469
Frisian	6,198	55,778	61,976
German	4,942	44,460	49,402
Italian	7,529	79,133	86,662
Norwegian	4,172	37,541	41,713
Spanish	3,150	28,341	31,491

Table 7.2: Pronunciation dictionary size for each of the languages.

7.4.1.1 Results

We consider training subsets of sizes 500, 1000, 1500, and 2000. For our baseline, we train the grapheme-to-phoneme model [8] on randomly selected words of each size, and average the results over 10 runs. We follow the same procedure for our two data set selection methods. Figure 7.2 plots the word prediction accuracy for all three methods across the eight languages with varying training sizes, while Table 7.3 provides corresponding numerical results. We see that in all scenarios the two data set selection strategies fare better than random subsets of words.

In all but one case, the feature coverage method yields the best performance (with the exception of Spanish trained with 500 words, where the CSSP yields the best results). Feature coverage achieves average error reduction of 20% over the randomly selected training words across the different languages and training set sizes.

	500 Words			2000 Words		
	RAND	CSSP	FEAT	RAND	CSSP	FEAT
Dut	48.2	50.8	**59.3**	69.8	75.0	**77.8**
Eng	25.4	26.5	**29.5**	40.3	40.1	**42.8**
Fra	66.9	69.2	**72.1**	81.2	82.0	**84.8**
Fri	42.7	48.0	**53.6**	62.2	65.3	**68.5**
Ger	55.2	58.6	**65.0**	74.2	78.6	**80.8**
Ita	80.6	82.8	**82.8**	85.3	86.1	**86.8**
Nor	48.1	49.5	**55.0**	66.1	69.9	**71.6**
Spa	90.7	**96.8**	95.0	98.1	98.4	**99.0**
avg	57.2	60.3	**64.0**	72.2	74.4	**76.5**

Table 7.3: Test word accuracy across the 8 languages for randomly selected words (RAND), CSSP matrix subset selection (CSSP), and Feature Coverage Maximization (FEAT). We show results for 500 and 2000 word training sets.

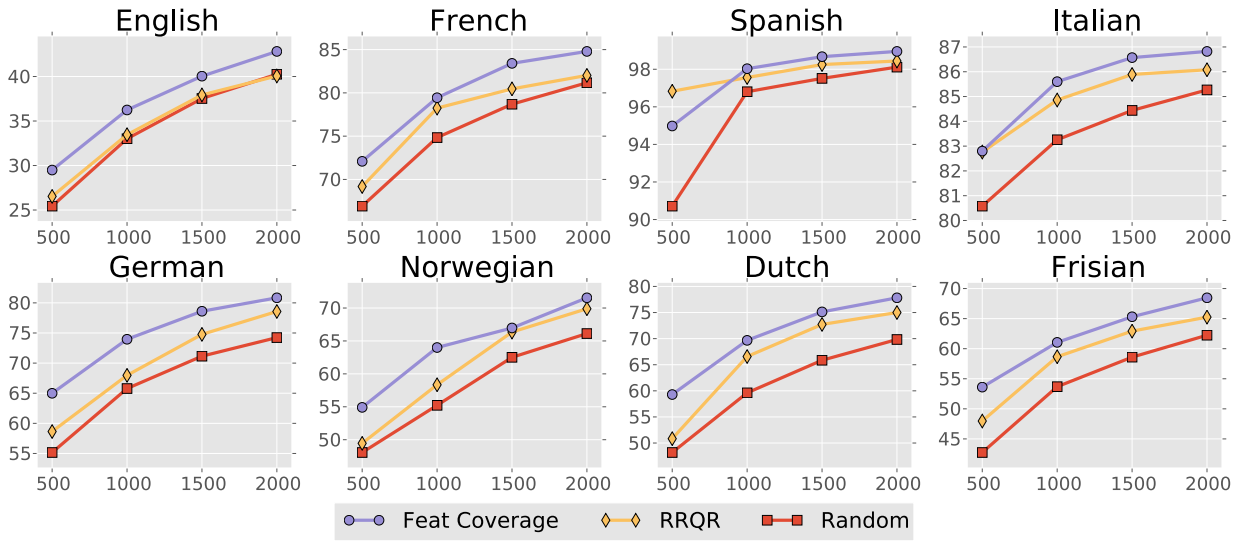

Figure 7.2: Test word accuracy across the 8 languages for (1) feature coverage, (2) CSSP matrix subset selection, (3) and randomly selected words.

7.4.1.2 Coverage variants

We also experimented with the other versions of the feature coverage function discussed in Section 7.3.2 (see Figure 7.1). While cov_1 tended to perform quite poorly (usually worse than random), cov_2 – yields results just slightly worse than the CSSP matrix method on average, and always better than random. In the 2000 word scenario, for example, cov_2 achieves average accuracy of 74.0, just a bit below the 74.4 accuracy of the CSSP method. It is also possible that more careful tuning of the discount factor η of cov_3 would yield further gains.

	RAND	CSSP	FEAT	SVD
Fra	0.66	0.62	0.65	0.51
Fry	0.75	0.72	0.75	0.6
Ger	0.71	0.67	0.71	0.55
Ita	0.64	0.61	0.67	0.49
Nor	0.7	0.61	0.64	0.5
Spa	0.65	0.67	0.68	0.53
avg	0.69	0.65	0.68	0.53

Table 7.4: Residual matrix norm across 6 languages for randomly selected words (RAND), CSSP matrix subset selection (CSSP), feature coverage maximization (FEAT), and the rank k SVD (SVD). Lower is better.

7.4.1.3 Optimization Analysis

Both the CSSP and feature coverage methods have clearly defined objective functions – formulated in Equations 7.3 and 7.8, respectively. We can therefore ask how well each methods fares in optimizing either one of the two objectives.

First we consider the objective of the CSSP algorithm: to find k data points which can accurately embed the entire data matrix. Once the data points are selected, we compute the orthogonal projection of the data matrix onto the submatrix, obtaining an approximation matrix $\tilde{A}$. We can

	RAND	CSSP	FEAT
Dut	0.66	0.72	0.81
Eng	0.52	0.58	0.69
Fra	0.68	0.74	0.81
Fry	0.7	0.79	0.84
Ger	0.68	0.74	0.81
Ita	0.79	0.84	0.9
Nor	0.7	0.79	0.84
Spa	0.67	0.75	0.8
avg	0.68	0.74	0.81

Table 7.5: Feature coverage across the 8 languages for randomly selected words (RAND), CSSP matrix subset selection (CSSP), and feature coverage maximization (FEAT). Higher is better.

then measure the residual norm as a fraction of the original matrix norm:

$$\frac{||A - \tilde{A}||_F}{||A||_F} \tag{7.9}$$

As noted in Section 7.3.1, truncated SVD minimizes the residual over all rank k matrices, so we can compare our three methods – all of which select k examples as a basis, against the lower bound given by SVD. Table 7.4 shows the result of this analysis for $k = 2000$ (Note that we were unable to compute the projection matrices for English and Dutch due to the size of the data and memory limitations). As expected, SVD fares the best, with CSSP as a somewhat distant second. On average, feature coverage seems to do a bit better than random.

A similar analysis for the feature coverage objective function is shown in Table 7.5. Unsurprisingly, this objective is best optimized by the feature coverage method. Interestingly though, CSSP seems to perform about halfway between random and the feature coverage method. This makes some sense, as good basis data points will tend to have frequent features, while at the same time being maximally spread out from one another. We also note that the poor coverage result for English in Table 7.5 mirrors its overall poor performance in the pronunciation dictionary induction

CSSP	FEAT	FEAT-SLS
fettered	internationalization	rating
exceptionally	underestimating	overs
gellert	schellinger	nation
daughtry	barristers	scherman
blowed	constellations	olinger
harmonium	complementing	anderson
cassini	bergerman	inter
rupees	characteristically	stated
tewksbury	heatherington	press
ley	overstated	conner

Table 7.6: Top 10 words selected by CSSP, feature coverage (FEAT), and feature coverage with stratified length sampling (FEAT-SLS)

task – not only are the phoneme labels unpredictable, but the input data itself is wild and hard to compress.

7.4.1.4 Stratified length sampling

As Table 7.6 shows, the top 10 words selected by the feature coverage method are mostly long and unusual, averaging 13.3 characters in length. In light of the potential annotation burden, we developed a stratified sampling strategy to ensure typical word lengths. Before selecting each new word, we first sample a word length according to the empirical word length distribution. We then choose among words of the sampled length according to the feature coverage criterion. This results in more typical words of average length, with only a very small drop in performance.

7.4.2 Task 2: Part-of-Speech Tagging

Next, we consider the task of part-of-speech tagging. The goal is to obtain a model for a given language that can accurately predict the part-of-speech (noun, verb, etc) of each word in the

context of a sentence. This is considered one of the most fundamental tasks in natural language processing, and is viewed as a prerequisite in the NLP pipeline for more complex tasks of language understanding. Although taggers now exist for a variety of languages, they are still absent for hundreds of languages and their accuracy degrades on new domains.

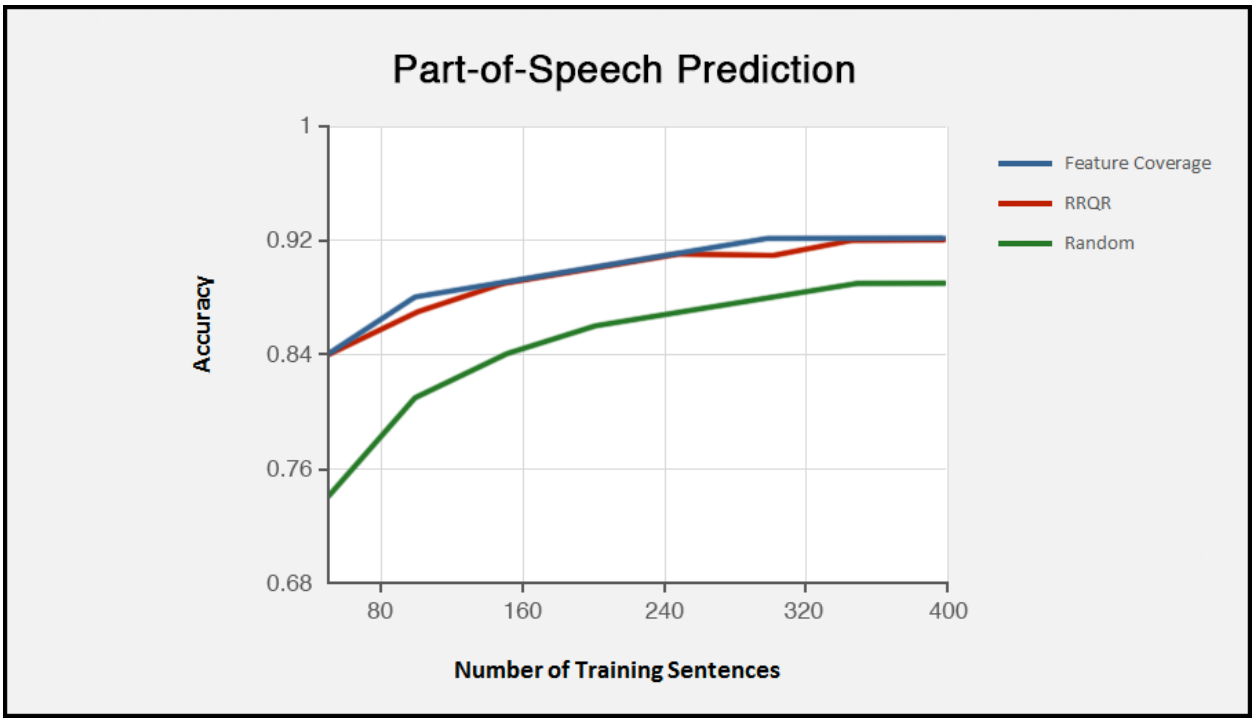

Figure 7.3: Average test sentence accuracy across the 11 languages for (1) feature coverage, (2) CSSP matrix subset selection, (3) and randomly selected training sentences.

In this section, we consider a dataset of 11 languages, using data obtained from the Multext east V4 corpus [37]. Because we are now dealing with *sentences* as our datapoints instead of just words, we first perform clustering over the words in order to reduce the dimensionality of our data matrix. We run a state-of-the-art unsupervised tagger, using the technique of posterior regularization [40], in order to obtain our initial word clusters. We then map each sentence into a binary feature vector indicating the presence or absence of all possible trigrams of cluster ID's.

We then apply our two data set selection methods to obtain subsets of labelled representative sentences as training data for a state-of-the-art Conditional Random Field (CRF) tagger. We

consider training sizes ranging from 50 to 400 sentences. As Figure 7.3 shows, we achieve considerable improvements in predictive accuracy, with performance improving from 78% to 88% accuracy when training on 40 sentences, and from 89% to 92% when training on 400 sentences. The results are consistent across all languages considered.

7.4.3 Task 3: Named Entity Recognition

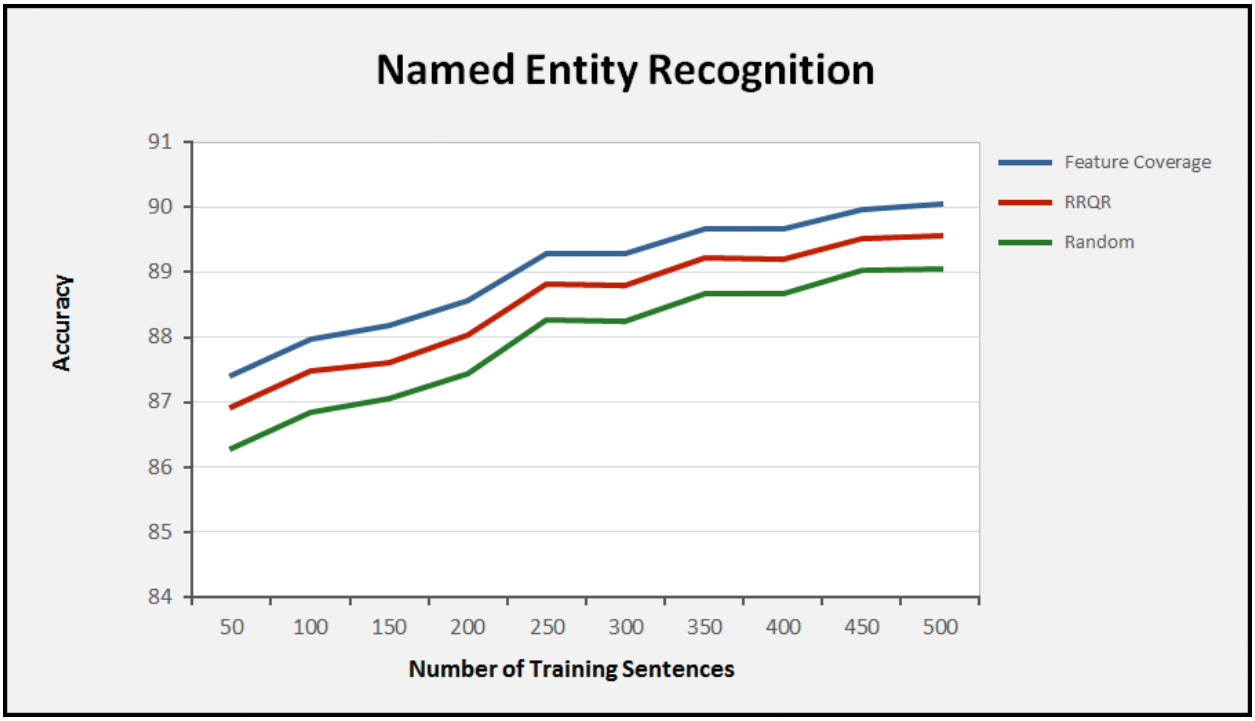

Figure 7.4: Average test sentence accuracy across the 9 languages for (1) feature coverage, (2) CSSP matrix subset selection, (3) and randomly selected sentences.

In our third task, we consider the problem of named entity recognition. The goal here is to train a model that can identify the spans of words in a sentence that refer to a named entity such as a person, place, corporation, or government. As per common practice, we treat this as a sequence tagging problem where the tag on each word indicates whether the word is outside, inside, or on the boundary of a named entity mention in the sentence.

We utilize a dataset collected from Wikipedia, covering nine languages [79]. These English data comes from the English Wikipedia snapshot of 30 July, 2010, and the subsequent snapshot for the other 8 languages in August, 2010. Each language contains 1,040,980 articles and 158,935 category pages on average. We consider training sizes from 50 to 500 sentences. As Figure 7.4 illustrates, we again achieve considerable improvements in predictive accuracy.

7.4.4 Task 4: Semantic Tagging

In our final task, we consider the problem of semantic tagging (slot filling). In this task, the goal is to match the user's utterance against a set of database templates indicating semantic queries. For example, one template may refer to movies, with various slots corresponding to dates, actors, directors, countries, genres, etc. When a user interacts with the model, its goal is to match the utterance to the template and determine which span of words fills in each slot (and with what values). As in the previous two tasks, this may be considered a sequence tagging problem (though with considerably more structure).

Domain	English	French	German
Apps	5,118	13,462	30,314
Games	22,723	32,369	30,416
Movies	45,366	22,007	43,434
Music	25,635	31,810	30,437

Table 7.7: Number of user queries in Semantic Tagging experiment.

We focus here on three domains of audiovisual media: Apps, Games, Music, and Movies. The user is expected to interact by voice with a system that can perform a variety of tasks in relation to each media, including (among others) browsing, searching, querying information, purchasing and playing. We use transcribed text utterances in our experiments. The size of the data is detailed in Table 7.7.

For this task, we consider training sets of size 500 to 1,500 user queries, and again train a Conditional Random Field (CRF) tagger on the labels of the selected examples. Figure 7.5 shows

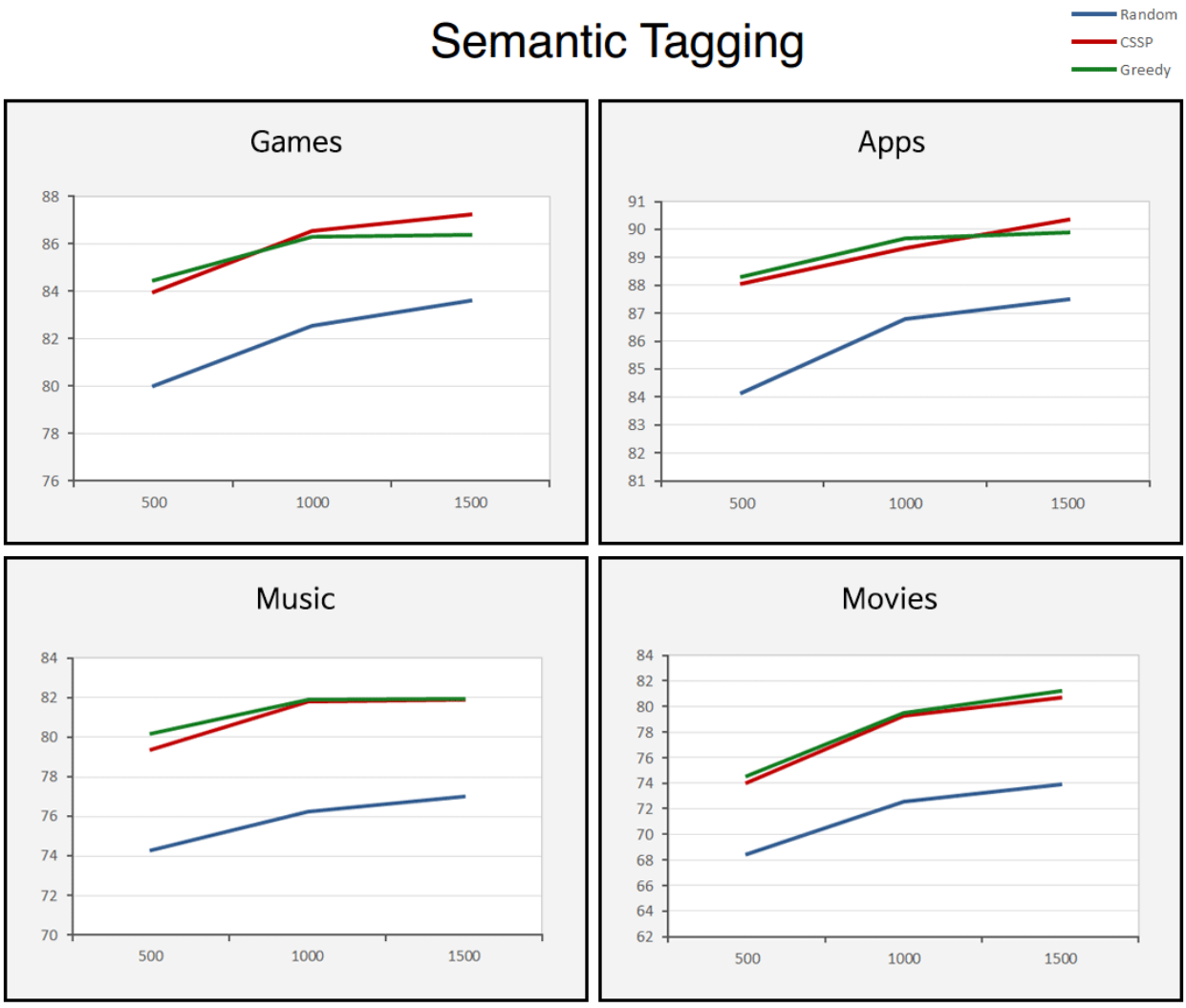

Figure 7.5: Average test query accuracy across the three languages for (1) feature coverage, (2) RRQR matrix subset selection, (3) and randomly selected queries.

the results for randomly selected queries versus our two methods for selected examples for labeling. These results again indicate that using our methods for intelligently selecting examples for labeling can lead to greatly improved performance across all domains.

7.5 Conclusions

Using supervised machine learning to create natural language models has one serious flaw: it requires labelled data in each language, domain, and task. In this chapter, we present two new methods to allow rapid proptyping and construction of robust natural language understanding systems while minimizing the need for annotated input. Our key premise is that by selecting examples for labeling which efficiently span the full space of examples, we can get by with many fewer datapoints without a loss in model accuracy.

To achieve this goal, we proposed the task of optimal data set selection in the unsupervised setting. In contrast to active learning, our methods do not require repeated training of multiple models and iterative annotations. Since the methods are unsupervised, they also avoid tying the selected data set to a particular model class (or even task).

We developed two methods for optimally selecting a small subset of examples for labeling. The first uses techniques developed by the numerical linear algebra and theory communities for approximating matrices with subsets of columns or rows. For our second method, we developed a novel notion of *feature coverage*. Experiments across a diverse set of natural language understanding tasks, ranging over 20 languages and six domains, show our method yielding performance improvements in all scenarios, averaging about 26% reduction in error. These results indicate that building robust natural language understanding systems may not require as much time consuming and expensive annotation as previously believed by researchers. For future work, we plan to improve upon these results by developing non-linear dimensionality reduction techniques using kernerlized SVD and deep learning.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we introduced two novel techniques to robustly handle languages lacking annotated resources.

In the first thread of our argument, the main assumption is that no annotated resources are available for the target language. For this scenario, we proposed a new framework cross-lingual supervised learning. The key idea underlying this framework is that through a joint analysis of a broad array of languages, languages with annotated resources can be used as training data for resource-poor languages. We have applied this idea to several fundamental task of NLP, morphological analysis in Chapter 2, grapheme-to-phoneme analysis in Chapter 3, language decipherment in Chapter 5 and part-of-speech tagging in Chapter 6. In all cases, ours methods yielded substantial performance gains without any human annotation for the target language.

In the second thread of our argument, the main assumption is that there is only limited budget or time for supervised annotation. For this scenario, we proposed two techniques for identifying an optimal set of examples to be labeled in order to produce a high-performance supervised model. These techniques are applied to natural language understanding tasks of pronunciation dictionary induction, part-of-speech prediction, named entity recognition, and semantic tagging. In all cases, we showed that our selection methods are effective at yielding a small, but optimal set of labeled examples. Existing state-of-the-art supervised models with proposed methods yielded substantial performance gains over randomly selected examples.

To conclude, we briefly introduce how to apply 1) cross-lingual supervised learning to domains lacking annotated resources especially in spoken language understanding (SLU) systems in practice and 2) the optimal selection algorithm to task of selecting compact lexicon from large, noisy gazetteers.

8.2 Future Work: Cross-domain Supervised Learning

First, the main goal of SLU is to automatically extract the meaning of spoken or typed queries. In recent years, this task has become increasingly important as more and more speech-based applications have emerged. Recent releases of personal digital assistants such as Siri, Google Now, Dragon Go and Cortana in smart phones provide natural language based interface for a variety of domains (e.g. places, weather, communications, reminders). The SLU in these domains are based on statistical machine learned models which require annotated training data. Typically each domain has its own schema to annotate the words and queries. However the meaning of words and utterances could be different in each domain. For example, "sunny" is considered a weather condition in the `weather` domain but it may be a song title in a `music` domain. Thus every time a new application is developed or a new domain is built, a significant amount of resources is invested in creating annotations specific to that application or domain.

For domain lacking annotated resource, we might attempt to apply cross-lingual supervised learning to this problem, but a straightforward application is not possible because these techniques assume that the label set is invariant.

One simple solution to this problem is to map label types across different domains or to find generic label types by abstracting the label types using the canonical correlation analysis (CCA) by Hotelling [50] a powerful and flexible statistical technique for dimensionality reduction. We derive a low dimensional representation for each label type that is maximally correlated to the average context of that label via CCA. These shared label representations, or label embeddings, allow us to map label types across different domains. We can apply the cross-lingual supervised learning techniques to solve the problem.

8.2.1 Inducing label embeddings

CCA-LABEL

Input: labeled sequences $\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$, dimension k

Output: label vector $v \in \mathbb{R}^k$ for each label type

1. For each label type $l \in \{1 \dots d\}$ and word type $w \in \{1 \dots d\}$ present in the sequences, calculate
   - **count**(l) = number of times label l occurs
   - **count**(w) = number of times word w occurs
   - **count**(l, w) = number of times word w occurs under label l
2. Define a matrix $\Omega \in \mathbb{R}^{d \times d'}$ where:

$$\Omega_{l,w} = \frac{\textbf{count}(l, w)}{\sqrt{\textbf{count}(l)\textbf{count}(w)}}$$

3. Perform rank-k SVD on Ω. Let $U \in \mathbb{R}^{d \times k}$ be a matrix where the i-th column is the left singular vector of Ω corresponding to the i-th largest singular value.
4. For each label l, set the l-th normalized row of U to be its vector representation.

Figure 8.1: CCA algorithm for inducing label embeddings.

I simply describe how to use CCA to induce vector representations for label types. Let n be the number of instances of labels in the entire data. Let $x_1 \dots x_n$ be the original representations of the label samples and $y_1 \dots y_n$ be the original representations of the associated words set contained in the labels.

We employ the following definition for the original representations for reasons we explain below. Let d be the number of distinct label types and d' be the number of distinct word types.

- $x_l \in \mathbb{R}^d$ is a zero vector in which the entry corresponding to the label type of the l-th instance is set to 1.

- $y_l \in \mathbb{R}^{d'}$ is a zero vector in which the entries corresponding to words spanned by the label are set to 1.

The motivation for this definition is that similar label types often have similar or same word.

For instance, consider two label types `start-time`, (start time of a calendar event) and `end-time`, meaning (the end time of a calendar event). Each type is frequently associated with phrases about time. The phrases {"9 pm", "7", "8 am"} might be labeled as `start-time`; the phrases {"9 am", "7 pm"} might be labeled as `end-time`. In these examples, both label types share words "am", "pm", "9", and "7" even though phrases may not match exactly.

Figure 8.1 gives the CCA algorithm for inducing label embeddings. It produces a k-dimensional vector for each label type corresponding to the CCA projection of the one-hot encoding of that label. With their vector representations, we can find generic label type across domains by simply clustering them or create a direct label mapping by finding the nearest neighbor of each label type between train domains and test domain.

Since label type across domains becomes same, any crosslingual supervised learning we introduced in this thesis can be used for domain lacking annotated resources.

8.3 Future Work: Compact Lexicon Selection

Discriminative models trained with large quantities of arbitrary features are a dominant paradigm in spoken language understanding (SLU) [87; 66; 2; 15; 47; 104]. An important category of these features comes from *entity dictionaries* or *gazetteers*—lists of phrases whose labels are given. For instance, they can be lists of movies, music titles, actors, restaurants, and cities. These features enable SLU models to robustly handle unseen entities at test time.

However, these lists are often massive and very noisy. This is because they are typically obtained by continuously mining the web for recent entries (such as newly launched movie names).

Ideally, we would like an SLU model to have access to this vast source of information at deployment. But this is difficult in practice because an SLU model needs to be light-weight to support fast user interaction. It becomes more challenging when we consider multiple domains, languages, and locales.

To tackle this challenge, I introduce the task of selecting a small, representative subset of noisy gazetteers that will nevertheless improve model performance as much as the original lexicon. This will allow an SLU model to take full advantage of gazetteer resources at test time without being overwhelmed by their scale.

The selection method is two steps. First, we gather relevant information for each gazetteer element using domain-specific search logs. Then we perform CCA using this information to derive low-dimensional gazetteer embeddings [50]. Second, we use a subset selection method based on RRQR to locate gazetteer embeddings whose span approximates the entire lexicon space [13].

8.3.1 Gazetteer Embeddings via CCA

In order to perform the selection algorithm in Figure 8.2, we first need a d-dimensional representation for each of n gazetteer elements. We can use CCA for its simplicity and generality.

In this case, we want to induce gazetteer embeddings that correlate with the relevant information about gazetteers. For this purpose, we use three types of features: context features, search click log features, and knowledge graph features.

Context features. For each gazetteer element g of domain l, we take sentences from search logs on domain l containing g and extract five words each to the left and the right of the element g in the sentences. For instance, if $g =$ "The Matrix" is a gazetteer element of domain $l =$ "Movie", we collect sentences from movie-specific search logs involving the phrase "The Matrix". Such domain-specific search logs is collected using a pre-trained domain classifier.

Search click log features. Large-scale search engines such as Bing and Google process millions of queries on a daily basis. Together with the search queries, user clicked URLs are also logged

anonymously. These click logs have been used for extracting semantic information for various NLP tasks [58; 102; 46]. We used the clicked URLs as features to determine the likelihood of an entity being a member of a movie dictionary. These features are useful because common URLs are shared across different movies. Table 8.1 shows the top three most frequently clicked URLs for movies "Furious 7" and "Romeo & Juliet".

Furious 7	**The age of adaline**
imdb.com	imdb.com
en.wikipedia.org	en.wikipedia.org
www.furious7.com	www.youtube.com
www.rottentomatoes.com	www.rottentomatoes.com
www.furious7.com	www.movieinsider.com

Table 8.1: Top clicked URLs of two movies.

One issue with using only click logs is that some entities may not be covered in the query logs since logs are extracted from a limited time frame (e.g. six months). Even the big search engines employ a moving time window for processing and storing search logs. Consequently, click logs are not necessarily good evidence. For example, "apollo thirteen" is a movie name appearing in the movie training data, but it does not appear in search logs. One solution is to search in *real time* so that we can obtain extra up-to-date URLs in addition to the URLs in search logs.

Thus we run live search through the search engine for all entities whether they appear in click logs or not. Each URL returned from a live search is considered to have one click.

Knowledge graph features. The graph in `www.freebase.com` contains a large set of tuples in a resource description framework (RDF) defined by W3C. A tuple typically consists of two entities: a subject and an object linked by some relation.

The most interesting information is the entity type defined in the graph for every entity. In the knowledge graph, the "type" relation represents the entity type. Table 8.2 shows some examples of

entities and their relations in the knowledge graph. From the graph, we learn that "Romeo & Juliet" could be a film name or a music album since it has two types: "film.film" and "music.album".

Subject	Relation	Object
Jason Statham	type	film.actor
Jason Statham	type	tv.actor
Jason Statham	type	film.producer
Romeo & Juliet	type	film.film
Romeo & Juliet	type	music.album

Table 8.2: Entities & relation in the knowledge graph.

8.3.2 Gazetteer Selection Algorithm

The algorithm is a two-stage procedure. In the first step, we randomly sample $O(m \log m)$ rows of A with carefully chosen probabilities and scale them to form columns of matrix $\bar{A} \in \mathbb{R}^{d \times O(m \log m)}$. In the second step, we perform RRQR factorization on $\bar{A}$ and collect the gazetteer elements corresponding to the top components given by the RRQR permutation. The algorithm is shown in Figure 8.2. The first stage involves random sampling and scaling of rows, but it is shown that $\bar{A}$ has $O(m \log m)$ columns with constant probability.

This algorithm has the following optimality guarantee:

Theorem 8.3.1 ([13]) Let $\hat{B} \in \mathbb{R}^{m \times d}$ be the matrix returned by the algorithm in Figure 8.2. Then with probability at least 0.7,

$$\left|\left|A - A\hat{B}^{+}\hat{B}\right|\right|_F \leq O(m\sqrt{\log m}) \times \min_{\substack{\tilde{A} \in \mathbb{R}^{n \times d}: \\ \text{rank}(\tilde{A}) = m}} \left|\left|A - \tilde{A}\right|\right|_F$$

In other words, the selected rows are not arbitrarily worse than the best rank-m approximation of A (given by SVD) with high probability.

Input: d-dimensional gazetteer representations $A \in \mathbb{R}^{n \times d}$, number of gazetteer elements to select $m \leq n$

Output: m rows of A, call $B \in \mathbb{R}^{m \times d}$, such that $||A - AB^{+}B||_{F}$ is small

- Perform SVD on A and let $U \in \mathbb{R}^{d \times m}$ be a matrix whose columns are the left singular vectors corresponding to the largest m singular values.
- Associate a probability p_i with the i-th row of A as follows:

$$p_i := \min \left\{ 1, \lfloor m \log m \rfloor \frac{||U_i||^2}{m} \right\}$$

- Discard the i-th row of A with probability $1 - p_i$. If kept, the row is multiplied by $1/\sqrt{p_i}$. Let these $O(m \log m)$ rows form the columns of a new matrix $\bar{A} \in \mathbb{R}^{d \times O(m \log m)}$.
- Perform RRQR on $\bar{A}$ to obtain $\bar{A}\Pi = QR$.
- Return the m rows of the original A corresponding to the top m columns of $\bar{A}\Pi$.

Figure 8.2: Gazetteer selection algorithm.

LIST OF REFERENCES

[1] Meni Adler and Michael Elhadad. An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 665–672, 2006.

[2] Tasos Anastasakos, Young-Bum Kim, and Anoop Deoras. Task specific continuous word representations for mono and multi-lingual spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3246–3250. IEEE, 2014.

[3] R. Harald Baayen, Richard Piepenbrock, and Léon Gulikers. The celex lexical database (version release 2)[cd-rom]. *Philadelphia, PA: Linguistic Data Consortium, University of Pennsylvania*, 1995.

[4] Francis R Bach and Michael I Jordan. A probabilistic interpretation of canonical correlation analysis. 2005.

[5] MSo Bartlett. The square root transformation in analysis of variance. *Supplement to the Journal of the Royal Statistical Society*, pages 68–78, 1936.

[6] Emily M. Bender. Linguistically naïve != language independent: why NLP needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics*, pages 26–32, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[7] Taylor Berg-Kirkpatrick and Dan Klein. Phylogenetic grammar induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[8] Maximilian Bisani and Hermann Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451, 5 2008.

[9] C.H. Bischof and G. Quintana-Ortí. Algorithm 782: codes for rank-revealing qr factorizations of dense matrices. *ACM Transactions on Mathematical Software (TOMS)*, 24(2):254–257, 1998.

[10] Phil Blunsom, Trevor Cohn, and Miles Osborne. Bayesian synchronous grammar induction. *Advances in Neural Information Processing Systems*, 21:161–168, 2009.

[11] Alexandre Bouchard-Côté, David Hall, Thomas L Griffiths, and Dan Klein. Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*, 110(11):4224–4229, 2013.

[12] Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. Unsupervised feature selection for principal components analysis. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–69, 2008.

[13] Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 968–977. Society for Industrial and Applied Mathematics, 2009.

[14] David Burkett, Slav Petrov, John Blitzer, and Dan Klein. Learning better monolingual models with unannotated bilingual text. In *Proceedings of the 14th Conference on Computational Natural Language Learning*, 2010.

[15] Asli Celikyilmaz, Dilek Z Hakkani-Tür, Gökhan Tür, and Ruhi Sarikaya. Semi-supervised semantic tagging of conversational understanding using markov topic regression. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 914–923, 2013.

[16] Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. A Bayesian mixture model for part-of-speech induction using multiple features. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 638–647. Association for Computational Linguistics, 2011.

[17] George N Clements. Feature economy in sound systems. *Phonology*, 20(3):287–333, 2003.

[18] Shay B Cohen and Noah A Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–82. Association for Computational Linguistics, 2009.

[19] Michael Collins. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, 2002.

[20] Piero Cosi, Roberto Gretter, and Fabio Tesser. Festival parla italiano. *Proceedings of GFS2000, Giornate del Gruppo di Fonetica Sperimentale, Padova*, 2000.

[21] Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2-3):201–233, 2002.

[22] Mathias Creutz and Krista Lagus. Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0. Publications in Computer and Information Science Report A81, Helsinki University of Technology, 2005.

[23] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1), 2007.

[24] Ido Dagan, Alon Itai, and Ulrike Schwall. Two languages are more informative than one. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 130–137, 1991.

[25] Peter T Daniels and William Bright. *The world's writing systems*, volume 198. Oxford University Press New York, NY, 1996.

[26] Dipanjan Das and Slav Petrov. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 600–609. Association for Computational Linguistics, 2011.

[27] Sajib Dasgupta and Vincent Ng. Unsupervised part-of-speech acquisition for resource-scarce languages. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 218–227, 2007.

[28] John DeNero and Klaus Macherey. Model-based aligner combination using dual decomposition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 420–429. Association for Computational Linguistics, 2011.

[29] Paramveer Dhillon, Jordan Rodu, Dean Foster, and Lyle Ungar. Two step CCA: A new spectral method for estimating vector models of words. In *Proceedings of the 29th International Conference on Machine learning*, 2012.

[30] Paramveer S Dhillon, Dean P Foster, and Lyle H Ungar. Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems*, volume 24, pages 199–207, 2011.

[31] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings 12th International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann Publishers, Inc., 1995.

[32] Matthew S Dryer, David Gil, Bernard Comrie, Hagen Jung, and Claudia Schmidt. *The world atlas of language structures*, volume 1. Oxford University Press, USA, 2005.

[33] Greg Durrett, Adam Pauls, and Dan Klein. Syntactic transfer using a bilingual lexicon. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–11. Association for Computational Linguistics, 2012.

[34] Kenneth Dwyer and Grzegorz Kondrak. Reducing the annotation effort for letter-to-phoneme conversion. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, pages 127–135. Association for Computational Linguistics, 2009.

[35] Matthias Eck, Stephan Vogel, and Alex Waibel. Low cost portability for statistical machine translation based on n-gram coverage. In *Proceedings of the Machine Translation Summit X*, 2005.

[36] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[37] Tomaz Erjavec. Multext-east version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *Fourth International Conference on Language Resources and Evaluation, LREC*, volume 4, pages 1535–1538, 2004.

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

[39] Victoria Fossum and Steven Abney. Automatically inducing a part-of-speech tagger by projecting from multiple source languages across aligned corpora. In *Natural Language Processing–IJCNLP 2005*, pages 862–873. Springer, 2005.

[40] Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049, 2010.

[41] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

[42] Harry R Glahn. Canonical correlation and its relationship to discriminant analysis and multiple regression. *Journal of the atmospheric sciences*, 25(1):23–31, 1968.

[43] John Goldsmith. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, 27(2):153–198, 2001.

[44] John Goldsmith. An algorithm for the unsupervised learning of morphology. Technical report, University of Chicago, 2005.

[45] Gene Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7(3):206–216, 1965.

[46] Dilek Hakkani-Tür, Gokhan Tur, Larry Heck, Asli Celikyilmaz, Ashley Fidler, Dustin Hillard, Rukmini Iyer, and S. Parthasarathy. Employing web search query click logs for multi-domain spoken language understanding. In *IEEE Automatic Speech Recognition and Understanding Workshop*, December 2011.

[47] Dustin Hillard, Asli Celikyilmaz, Dilek Z Hakkani-Tür, and Gökhan Tür. Learning weighted entity lists from web click logs for spoken language understanding. In *12th Annual Conference of the International Speech Communication Association*, pages 705–708, 2011.

[48] Yoo Pyo Hong and C-T Pan. Rank-revealing factorizations and the singular value decomposition. *Mathematics of Computation*, 58(197):213–232, 1992.

[49] H Hotelling. Canonical correlation analysis (cca). *Journal of Educational Psychology*, 1935.

[50] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[51] Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*, 11(3):311–325, 2005.

[52] Keki B Irani. Multi-interval discretization of continuous-valued attributes for classification learning. 1993.

[53] Sittichai Jiampojamarn and Grzegorz Kondrak. Letter-phoneme alignment: An exploration. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 780–788. Association for Computational Linguistics, 2010.

[54] Sham M Kakade and Dean P Foster. Multi-view regression via canonical correlation analysis. In *Learning Theory*, pages 82–96. Springer, 2007.

[55] Ronald M Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378, 1994.

[56] Michael Kenstowicz and Charles Kisseberth. *Generative phonology*. Academic Press San Diego, CA, 1979.

[57] Young-Bum Kim, João V Graça, and Benjamin Snyder. Universal morphological analysis using structured nearest neighbor prediction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 322–332. Association for Computational Linguistics, 2011.

[58] Young-Bum Kim, Minwoo Jeong, Karl Startos, and Ruhi Sarikaya. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[59] Young-Bum Kim and Benjamin Snyder. Universal grapheme-to-phoneme prediction over latin alphabets. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing*, pages 332–343, Jeju Island, South Korea, July 2012. Association for Computational Linguistics.

[60] Young-Bum Kim and Benjamin Snyder. Optimal data set selection: An application to grapheme-to-phoneme conversion. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1196–1205. Association for Computational Linguistics, 2013.

[61] Young-Bum Kim and Benjamin Snyder. Unsupervised consonant-vowel prediction over hundreds of languages. pages 1527–1536, 2013.

[62] Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. Unsupervised analysis for decipherment problems. In *Proceedings of the 2006 Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*, pages 499–506. Association for Computational Linguistics, 2006.

[63] John Kominek and Alan W Black. Learning pronunciation dictionaries: language complexity and word selection strategies. In *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 232–239. Association for Computational Linguistics, 2006.

[64] Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. Simple type-level unsupervised POS tagging. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 853–861. Association for Computational Linguistics, 2010.

[65] Shen Li, Joao V Graça, and Ben Taskar. Wiki-ly supervised part-of-speech tagging. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1389–1398. Association for Computational Linguistics, 2012.

[66] Xiao Li, Ye-Yi Wang, and Alex Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009.

[67] Percy Liang, Michael I Jordan, and Dan Klein. Type-based MCMC. In *Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 573–581. Association for Computational Linguistics, 2010.

[68] Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics, 2006.

[69] Johan Liljencrants and Björn Lindblom. Numerical simulation of vowel quality systems: the role of perceptual contrast. *Language*, pages 839–862, 1972.

[70] David JC MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[71] K.Z. Mao. Identifying critical variables of principal components for unsupervised feature selection. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(2):339–344, 2005.

[72] Yannick Marchand and Robert I Damper. A multistrategy approach to improving pronunciation by analogy. *Computational Linguistics*, 26(2):195–219, 2000.

[73] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1994.

[74] Ryan McDonald, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 62–72. Association for Computational Linguistics, 2011.

[75] Philippe Mousty and Monique Radeau. Brulex. une base de données lexicales informatisée pour le français écrit et parlé. *L'année psychologique*, 90(4):551–566, 1990.

[76] Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. Multilingual part-of-speech tagging: two unsupervised approaches. *Journal of Artificial Intelligence Research*, 36(1):341–385, 2009.

[77] Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. Multilingual part-of-speech tagging: Two unsupervised approaches. *Journal of Artificial Intelligence Research*, 36(1):341–385, 2009.

[78] Radford M Neal. Slice sampling. *Annals of statistics*, 31:705–741, 2003.

[79] Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194:151–175, 2012.

[80] Sebastian Padó and Mirella Lapata. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 1161 – 1168, 2006.

[81] Gerald Penn and Travis Choma. Quantitative methods for classifying writing systems. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 117–120. Association for Computational Linguistics, 2006.

[82] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*, 2011.

[83] Hoifung Poon, Colin Cherry, and Kristina Toutanova. Unsupervised morphological segmentation with log-linear models. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 209–217, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[84] Sujith Ravi and Kevin Knight. Learning phoneme mappings for transliteration without parallel data. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 37–45. Association for Computational Linguistics, 2009.

[85] Sravana Reddy and John Goldsmith. An mdl-based approach to extracting subword units for grapheme-to-phoneme conversion. In *Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 713–716. Association for Computational Linguistics, 2010.

[86] Philip Resnik and David Yarowsky. A perspective on word sense disambiguation methods and their evaluation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, pages 79–86, 1997.

[87] Ruhi Sarikaya, Asli Celikyilmaz, Anoop Deoras, and Minwoo Jeong. Shrinkage based features for slot tagging with conditional random fields. In *In Proceeding of ISCA - International Speech Communication Association*, September 2014.

[88] Patrick Schone and Daniel Jurafsky. Knowledge-free induction of inflectional morphologies. In *Proceedings of the 2001 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–9, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[89] Terrence J Sejnowski and Charles R Rosenberg. Parallel networks that learn to pronounce english text. *Complex systems*, 1(1):145–168, 1987.

[90] Burr Settles. Active learning literature survey. Technical Report TR1648, Department of Computer Sciences, University of Wisconsin-Madison, 2010.

[91] Benjamin Snyder and Regina Barzilay. Cross-lingual propagation for morphological analysis. In *Proceedings of the 23rd National Conference on Artificial Intelligence.*, pages 848–854, 2008.

[92] Benjamin Snyder and Regina Barzilay. Unsupervised multilingual learning for morphological segmentation. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 737–745, 2008.

[93] Benjamin Snyder, Regina Barzilay, and Kevin Knight. A statistical model for lost language decipherment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1048–1057. Association for Computational Linguistics, 2010.

[94] Benjamin Snyder, Tahira Naseem, and Regina Barzilay. Unsupervised multilingual grammar induction. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, pages 73–81, 2009.

[95] Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. Unsupervised multilingual learning for POS tagging. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1041–1050, 2008.

[96] Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. Adding more languages improves unsupervised multilingual part-of-speech tagging: A bayesian nonparametric approach. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 83–91. Association for Computational Linguistics, 2009.

[97] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*, volume 81. The MIT Press, 2000.

[98] Richard Sproat, Tao Tao, and ChengXiang Zhai. Named entity transliteration with comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 73–80. Association for Computational Linguistics, 2006.

[99] Richard W. Sproat. *A computational theory of writing systems*. Cambridge Univ Press, 2000.

[100] Hervé Stoppiglia, Gérard Dreyfus, Rémi Dubois, and Yacine Oussar. Ranking a random feature for variable and feature selection. *The Journal of Machine Learning Research*, 3:1399–1414, 2003.

[101] Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12, 2013.

[102] Huihsin Tseng, Longbin Chen, Fan Li, Ziming Zhuang, Lei Duan, and Belle Tseng. Mining search engine clickthrough log for matching n-gram features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 524–533. Association for Computational Linguistics, 2009.

[103] Lior Wolf and Amnon Shashua. Feature selection for unsupervised and supervised inference: The emergence of sparsity in a weight-based approach. *The Journal of Machine Learning Research*, 6:1855–1887, 2005.

[104] Puyang Xu and Ruhi Sarikaya. Targeted feature dropout for robust slot filling in natural language understanding. In *Proceedings of 15th Annual Conference of the International Speech Communication Association*, September 2014.

[105] David Yarowsky and Grace Ngai. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of the 2001 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–8, 2001.

[106] David Yarowsky, Grace Ngai, and Richard Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics, 2001.

[107] David Yarowsky and Richard Wicentowski. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

[108] Zheng Zhao and Huan Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on Machine learning*, pages 1151–1157, 2007.