Identifying Cognate Sets Across Dictionaries of Related Languages

by

Adam St Arnaud

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

Master of Science

Department of Computing Science

University of Alberta

Abstract

Cognates are words in related languages that have originated from the same word in an ancestor language, such as the English/German word pair *father/Vater*. Cognate information is critical in the field of historical linguistics, where it is used to determine the relationships between languages and to construct the ancestor languages they originated from. Most recent work in cognate identification focuses on the task of clustering cognates within lists of words each having an identical definition. In that task, only orthographic or phonetic information about a word is utilized when making cognate judgments. We present a system for the more challenging task of identifying cognate sets across dictionaries of related languages. The likelihood of a cognate relationship is calculated on the basis of a rich set of features that capture both phonetic and semantic similarity, as well as the presence of regular sound correspondences. The pairwise similarity scores are combined with an average-score clustering algorithm to create sets of words from different languages that may originate from a common proto-word. When tested on the Algonquian language family, our system detects 63% of cognate sets while maintaining cluster purity of 70%.

Acknowledgements

First, I would like to thank my supervisor Dr. Grzegorz Kondrak for his mentorship, direction, and support. I learned so much working on our project together and by participating in shared tasks with the NLP team. And I always enjoyed talking about computational linguistics and languages with Greg in his office.

I would also like to thank Dr. David Beck for his involvement in this project, especially his Totonacan expertise and hours of manual annotation.

Thank you to Dr. J. Nelson Amaral for his guidance and instruction throughout my computing science education and for encouraging me to pursue a graduate degree.

I would also like to thank Dr. Hong Zhang, Dr. Lorna Stewart, and the late Dr. Piotr Rudnicki, for providing me with opportunities to work on various projects as a summer intern.

Thanks to Dylan Stankievech, Jacob Denson, Tyler Lazar, and Cindy Xiao for their preliminary work on this project, and thanks to Mildred Lau for her help with constructing the Totonacan data files.

Last but not least, I would like to thank my parents for their continued support and encouragement throughout all my years of school, and Amy, for being by my side through it all.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cognates are words in related languages that have originated from the same word in an ancestor language. An example is the English/German word pair *father/Vater*. In this example, the words look similar and have identical definitions; however, it is possible for cognates to have differing appearances and meanings due to gradual phonetic and semantic shifts over the years. Consider the English/French cognate pair *father/père*, which no longer share obvious orthographic similarities. Alternatively, the English/German pair *starve/sterben* (where sterben means "to die") no longer share a meaning. On average, however, cognates display more phonetic and semantic similarity than random word pairs between languages (Kondrak, 2012).

1.1 Motivation

Cognate information between languages is critical to the field of historical and comparative linguistics, where it plays a central role in determining the relations and structures of language families (Trask, 1996). Finding sets of cognate words between a set of languages is the first step in what is known as the *comparative method*, one of the main approaches historical linguists have at their disposal. The comparative method is built on the assumption that phonological changes in languages occur regularly. This means that as a sound changes into another sound within a language, this change occurs across the entire language, not just in random words (Trask, 1996). This assumption allows us to find patterns between related languages, where a sound in one language generally corresponds to a sound in another language. These patterns are known as *regular sound correspondences*, or simply *correspondences*. As an example, consider Table 1.1, which contains a correspondence, Sardinian *k-* : Italian *tʃ-* : Romansch *ts-* : French *s-* : Spanish *θ-*, as well as example words where this correspondence can be seen. This correspondence can be found in many words throughout these languages; indeed this is what makes it a *regular* correspondence and not simply a coincidence for the words shown in the table. When languages share many such correspondences, we take this as evidence that they are genetically related. In this case, all of these languages

	Sardinian	Italian	Romansch	French	Spanish
Correspondence	k-	tʃ-	ts-	s-	θ-
“100”	kɛntu	tʃɛnto	tsjɛnt	sã	θjen
“sky”	kɛlu	tʃelo	tsil	sjɛl	θjelo
“stag”	kɛrbu	tʃɛrvo	tsɛrf	sɛʀ	θjerbo
“wax”	kɛra	tʃera	tsaira	siʀ	θera

Table 1.1: Example Romance language regular sound correspondence (Trask, 1996, p. 254).

are *Romance* languages, meaning they descended from Latin. The words meaning “sky,” for example, originated from the Latin word *caelum* (also meaning “sky”), making them cognates. Since there are written records of Latin, we can verify each Romance language correspondence and determine the original sound; however, even when no such ancient data exist, we can use regular correspondences to determine what the ancestor language must have sounded like. This is called *comparative reconstruction* and is a main outcome of the comparative method.

Identifying such regular correspondences between languages is central to the comparative method, and to identify these, linguists must first have sets of cognates (like the “sky” example above) to work with. Moreover, the method can be thought of as an iterative process, as regular correspondences help to identify more cognates, which in turn, can be used to identify even more correspondences. But while information regarding cognates is of the utmost importance to linguists, it is not always readily available. The languages that are the least well studied, and hence the ones that historical linguists are most interested in, are often lacking annotated cognate information. This means that linguists need to go through the effort of manually annotating cognate sets from hundreds, if not thousands, of words between various languages, which can take immense amounts of time. For this reason, there has been interest within the field of computational linguists to automate the methods of historical linguistics, and in particular, the task of cognate identification. Systems that can automatically find and identify cognates can greatly reduce the amount of time needed to perform the comparative method and give objective, reproducible results. One can even consider automated cognate identification systems as a supplement to aid expert linguists. The task for a linguist is thus reduced from analyzing all possible word pairs between language dictionaries, to annotating sets of proposed cognates.

1.2 Cognate Identification Tasks

In general, the task of automatic cognate identification can be classified according to two questions: (1) are pairs of words being classified as cognates, or are groups of words being clustered together into cognate sets? (2) are cognate classifications made based on only orthographic/phonetic information, or do they also take into account semantic information? This in turn leads to four main task variations:

1. Classify word pairs as cognate or not based on orthographic/phonetic information (Mulloni, 2007; Bergsma and Kondrak, 2007; Ciobanu and Dinu, 2013; Rama, 2015).

2. Cluster words that have identical definitions into cognate sets based on orthographic/phonetic information (Hauer and Kondrak, 2011; List, 2012; List and Moran, 2013; Bouchard-Côté et al., 2013; List et al., 2016).

3. Classify word pairs as cognate or not based on orthographic/phonetic information, as well as semantic information (Kondrak, 2004; Wang and Sitbon, 2014).

4. Cluster word pairs into cognate sets based on orthographic/phonetic information, as well as semantic information, where cognate sets can include words with different meanings (Kondrak et al., 2007; Steiner et al., 2011).

Figures 1.1-1.4 provide visual examples of these four tasks, and we refer to them as the Form-Pairs, Form-Sets, Word-Pairs, and Word-Sets tasks, respectively. Here, the term *form* represents that a task involves classifying cognates based on the orthographic or phonetic forms of words, whereas the term *word* represents that a task involves classifying cognates based on both the orthographic or phonetic forms of a words along with their definitions.

The Form-Pairs and Form-Sets tasks have received a majority of the attention in the cognate literature, while the Word-Pairs and Word-Sets tasks remain relatively understudied. In this thesis, we focus on the Word-Sets task, and the goal of our work can be described with the following claim:

> *It is possible to construct an automated system that takes language dictionaries from related languages as input and identifies a majority of the cognate sets between these languages.*

As a metaphor, imagine leafing through several physical language dictionaries in search of cognates, where each dictionary is organized as lists of word forms and their corresponding definitions.

This problem is more difficult than the Form-Sets task because one must take into account the semantic similarity of non-identical definitions when making decisions, and therefore, there are exponentially more pairwise cognate classifications to determine. Figure 1.4 shows examples of Algonquian cognate sets including word forms and definitions, where we can see that cognates need not have identical definitions. And unlike the Word-Pairs task, where only pairwise classifications are required, we must also consider how to cluster words into sets. Due to the transitive nature of cognation, all words within an outputted cognate set are deemed to be cognate with each other. Organizing words into cognate sets adds a level of complexity to the Word-Sets task that is not present in the Word-Pairs task. In the following section, we describe the assumptions that we make about our task and the steps that we took in order to achieve our end goal.

1.3 Our Work

One can analyze orthographic (how a word is written), or phonetic (how a word sounds) representations when determining cognate relationships. In our work, we focus on phonetic representations, as the string-similarity between two phonetic shapes is not dependent on the spelling or writing systems of the given languages. For example, the words displayed in Table 1.1, are written in the International Phonetic Alphabet (IPA), a standardized phonetic representation. This allows for the underlying sound of a word to be easily detected. And if language data is presented in orthographic form, it can be mapped into a phonetic representation while preserving much if not all of the similarity.

The term *cognate* is sometimes used within computational linguistics to denote words that simply have similar meanings and sound similar (Mulloni, 2007; Nakov and Tiedemann, 2012; Beinborn et al., 2013). This definition would encompass words such as proper names or lexical borrowings (when one language adopts a word from another language) but leave out true cognates that do not sound similar or have similar meanings. These non-cognates that have similar meanings and phonetic forms are sometimes referred to as *false cognates*. False cognates can prove difficult for an automated system to detect, and in practice, an expert linguist would be able to discern them from actual cognates within proposed cognate sets; however, we work under the technical definition of cognate, and hence attempt to find clusters of words with a common historical origin. One mechanism that helps discriminate false cognates from genuine cognates is the detection of regular sound correspondences. The regularity displayed by such correspondences across two languages need not be apparent in a one-off borrowing or chance resemblance. For an example of false cognates, consider the Turkish word *haber* and the Swahili word *habari*, both meaning "news." The reason that these two words are so similar is because they both originated from an Arabic borrowing, hundreds of years ago (Trask, 1996).

Gold data is a term commonly used in Natural Language Processing literature, which refers to labelled input data. In the context of cognate identification, gold data could include example word pairs and a corresponding label indicating whether each word pair are cognates or not. In our task, we do not assume the existence of any gold cognate data for the language family that we are analyzing, making this an unsupervised task. However, while our task is unsupervised in nature, our approach is still able to utilize supervised learning techniques from machine learning by employing gold cognate data from a completely unrelated language family. This means that we can use cognate knowledge gained from language families that are already well studied to inform our decisions on a language family that little is known about.

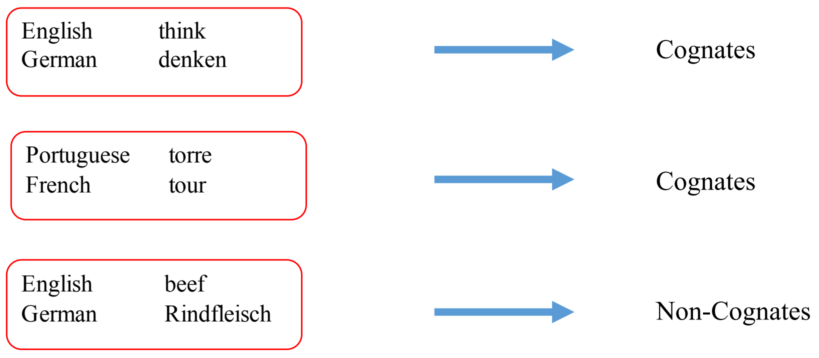


Figure 1.1: Example of cognate pair identification using forms: Form-Pairs

Our main machine learning classifier uses a variety of features based on both phonetic and semantic information. We also derive pairwise models for each language pair that are aimed at exploiting regular sound correspondences. The pairwise models utilize the substring features of Bergsma and Kondrak (2007), which learn which substrings of one language are often aligned with substrings of another language. Then, the scores from both the main and pairwise models are combined to leverage general cognate information with language-pair-specific knowledge. This combined score is provided to an average similarity clustering algorithm to construct the proposed cognate sets.

Several metrics have been used to evaluate cognate identification systems; we investigate these metrics in detail and demonstrate that the commonly used B-Cubed metric is unsuitable for evaluating a cognate clustering task across dictionaries. We argue that the total number of sets found by a system is the most important measure of a system's recall and balance this measure with cluster purity, which is a measure of precision. When tested on the Algonquian language family, our system is able to find 63.1% of gold cognate sets while maintaining a cluster purity of 70.3%.

This thesis begins with a discussion of previous work on cognate identification in Chapter 2. In Chapter 3, we describe the data sets that we are using in our experiments. In Chapter 4, we explain our methods and approach to this task. Chapter 5 contains a discussion on attempted methods that were surpassed by our current system. We analyze various possible evaluation metrics for this task in Chapter 6. In Chapter 7, we explain the experimental set-ups and discuss the results and errors made by our system. Finally, Chapter 8 contains an overview of what we achieved and possible avenues for future work.

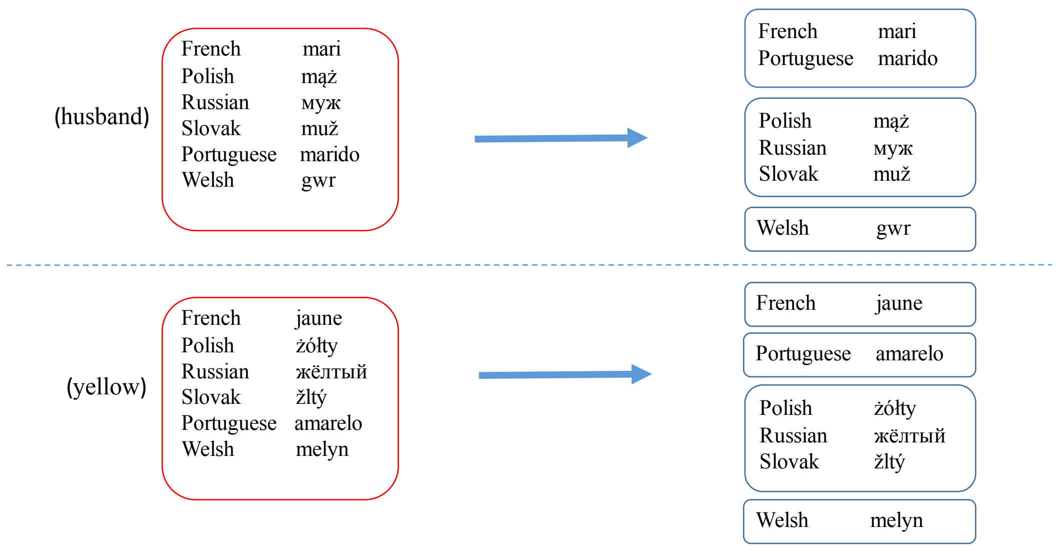


Figure 1.2: Example of cognate set identification using forms: Form-Sets

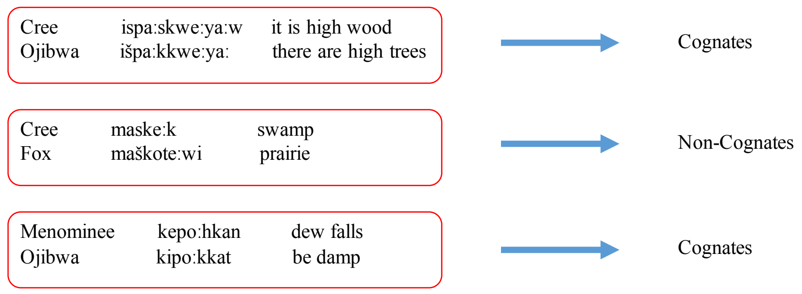
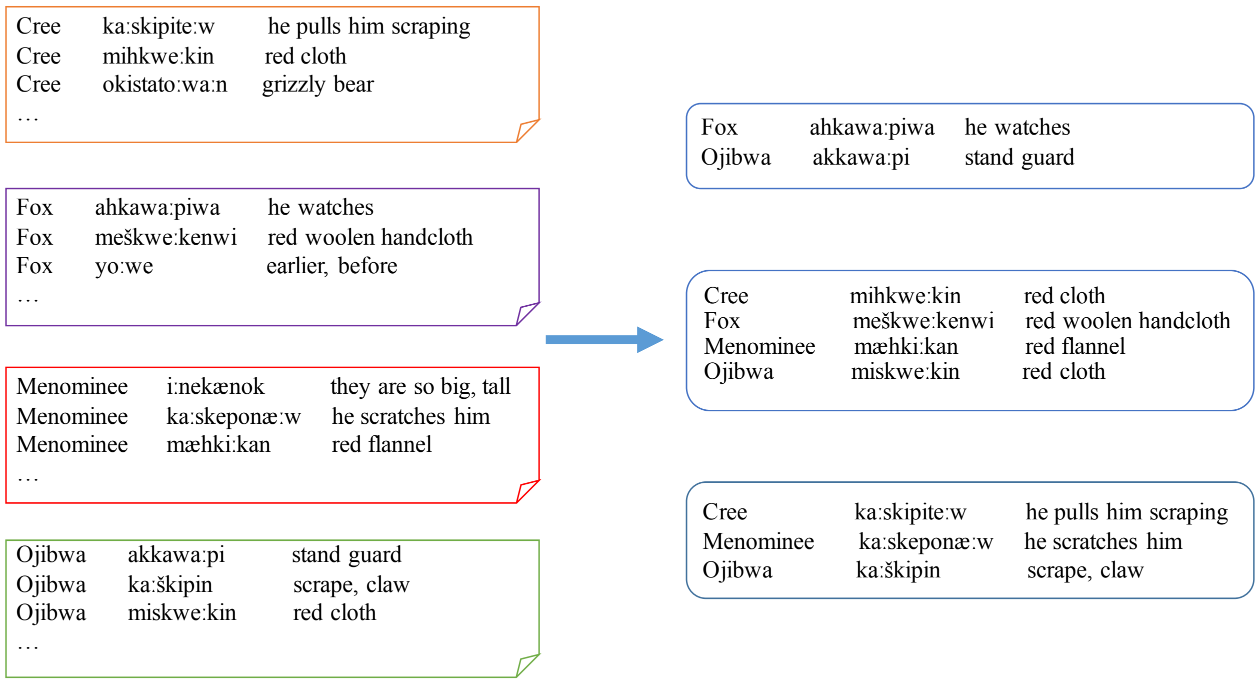

Cree ispa:skwe:ya:w it is high wood
Ojibwa išpa:kkwe:ya: there are high trees
Cognates
Cree maske:k swamp
Fox maškote:wi prairie
Non-Cognates
Menominee kepo:hkan dew falls
Ojibwa kipo:kkat be damp
Cognates

Figure 1.3: Example of cognate pair identification using words: Word-Pairs



Cree	ka:skipite:w	he pulls him scraping
Cree	mihkwe:kin	red cloth
Cree	okistato:wa:n	grizzly bear
…		

Fox	ahkawa:piwa	he watches
Fox	meškwe:kenwi	red woolen handcloth
Fox	yo:we	earlier, before
…		

Menominee	i:nekænok	they are so big, tall
Menominee	ka:skeponæ:w	he scratches him
Menominee	mæhki:kan	red flannel
…		

Ojibwa	akkawa:pi	stand guard
Ojibwa	ka:škipin	scrape, claw
Ojibwa	miskwe:kin	red cloth
…		

Fox	ahkawa:piwa	he watches
Ojibwa	akkawa:pi	stand guard

Cree	mihkwe:kin	red cloth
Fox	meškwe:kenwi	red woolen handcloth
Menominee	mæhki:kan	red flannel
Ojibwa	miskwe:kin	red cloth

Cree	ka:skipite:w	he pulls him scraping
Menominee	ka:skeponæ:w	he scratches him
Ojibwa	ka:škipin	scrape, claw

Figure 1.4: Example of cognate set identification using words: Word-Sets

Chapter 2

Related Work

As described in Chapter 1, work in automated cognate identification can broadly be classified into four tasks. In this chapter we discuss the recent work within each of these tasks.

2.1 Pairwise Cognate Identification Without Semantic Information

The simplest of the cognate identification tasks is pairwise classification based on orthographic or phonetic information only (the Form-Pairs task). In this section, we provide an overview of recent work done on this task.

Turchin et al. (2010) identify the ratio of cognate pairs to non cognate pairs in basic vocabulary lists between languages in an effort to determine the likelihood that they are related. Pairwise cognate identification is not their primary goal, but it is an important part of their system. Their method groups consonant sounds into 9 distinct classes and uses these classes to make cognate judgments. They show that their method gives expected results when determining the widely accepted relationships of the Indo-European languages and go on to provide evidence for the relatedness of the *Altaic* languages. A possible criticism of their approach is that the cognate judgments are based on a very simplistic heuristic, which only considers the first two consonants within words, rather than using more sophisticated string similarity methods or attempting to utilize sound correspondences between languages.

Ciobanu and Dinu (2013) propose a method for finding cognate pairs by using dictionaries with etymological information. Their method starts by looking up a given word in a dictionary that includes information on etymology and ancestor words. Next they translate this word using Google Translate into a target language to find a possible cognate candidate. If the candidate has the same etymology as the source word, then the words are considered cognate. They use this approach on a Romanian corpus of transcribed parliamentary debates to identify cognates in both French and Italian. They also investigate several string similarity metrics at various thresholds and determine that edit distance scores the highest accuracy for both French and Italian. A main critique of this paper is that they assume the existence of detailed information regarding ancestor words, which is often not available, especially in language families that are not well known and hence where cognate identification is most useful.

Rama (2015) experiment with features motivated by string kernels for pairwise classification. They use an SVM classifier to implement the subsequence feature vector and compares against a classifier trained with a set of state-of-the-art string similarity features. They test their method on the Comparative Indo-European Database and show that it outperforms the model trained on classical similarity features.

2.2 Cognate Clustering Without Semantic Information

While the work in the preceding section focuses on identifying or classifying pairs of words as cognates, the work in this section goes one step further. On top of making pairwise cognate classifications, these systems must also decide on a method for clustering words into cognate sets, where each word in the set is cognate with all others (the Form-Sets task). Much of the work in this area begins with words organized into *semantically aligned word lists*, or just *word lists*. These are groups of words all having an identical definition, like the Swadesh lists, commonly used in historical linguistics (Swadesh, 1952).

Hauer and Kondrak (2011) start with such word lists and use an SVM classifier with a variety of string similarity features, such as edit distance, longest common prefix, the number of common bigrams, and the difference in word lengths, to make pairwise cognate classifications. On top of the string similarity features, they also include a binary language-pair feature, which attempts to gain cognate information across an entire language pair. To utilize this language pair feature, they perform *self-training*, where the output of their system without the language pair feature becomes part of the input for a second pass. They run their classifier on all word pairs within word lists, and the resulting classifications are combined with an average score clustering algorithm. They test their approach on the Comparative Indo European Database, as well as on the Automated Similarity Judgment Program (ASJP) and achieve F-Score results in the mid 60% to low 70% range.

Hall and Klein (2010; 2011) propose a method for identifying cognate sets in a large number of related languages. In essence, their task can be considered clustering cognates within word lists, since they do not allow for cognate relationships between words with non-identical definitions. The method is based on two generative models, each describing the evolution of words along a phylogeny according to automatically learned sound laws in the form of parametric edit distances. Their model, PARSIM, uses Markov processes to allow for mutations and innovations along the phylogeny and is inspired by the parsimony principle from biology. PARSIM has parameters for mutation, innovation, and a global language model for generating new words which must be learned. They test their approach on the Formosan and Oceanic subfamilies of Austronesian, using data from the Austronesian Basic Vocabulary Database. Their main result is a pairwise recall of 62.1% and cluster purity score of 91.8% on the Oceanic subfamily.

List (2012) and List and Moran (2013) attempt an automated approach to the comparative method for finding cognate sets within word lists. Word pairs are first converted into sound classes and then aligned to find *residue pairs*, i.e. those segments that are aligned in the same column. They compare the alignments between words with the same meaning against alignments between words with different meanings in order to calculate language-specific distance scores between residue pairs. The idea is that residue pairs between words with the same meaning are more likely to signify cognation, while residue pairs between words with different meanings represent chance alignments. Distance scores are then calculated for word pairs themselves, based on the residue pair scores of their corresponding alignment. Finally, they use an average score clustering algorithm to cluster cognates within word lists based on word pair distance scores. While their system is designed for the Form-Sets task and can only create clusters of words with identical definitions, we apply it to the Word-Sets task and compare its output against our results.

List et al. (2016) extend their work to a similar task of clustering *partial* cognates in word lists. In this task, they take input word lists, but they output judgments about the cognation of morphemes, rather than whole words. They test an average score clustering technique against Markov clustering and InfoMap (Rosvall and Bergstrom, 2008), a heuristic algorithm designed for finding communities in networks, showing that InfoMap yields good results.

2.3 Pairwise Cognate Identification With Semantic Information

Using semantic information to inform cognate decisions is a relatively understudied concept. The systems described in the previous sections make pairwise or multiwise decisions on cognation based on the orthographic or phonetic form of words and hence only allow cognate judgments between words with identical meanings. In this section, we discuss work that attempts to harness semantic information to help inform pairwise cognate classifications (the Word-Pairs task).

Kondrak (2004) proposes a system for identifying cognate pairs between language dictionaries. The method is based on combining evidence of phonetic similarity, complex multi-phoneme correspondences, and semantic information. Their main result is an average 11-point Interpolated Precision of 66% on a subset of the Algonquian data under their best configuration. Possible criticisms of their work are that their feature scores are combined by hand-tuning, and that they make no use of machine learning techniques. While the end goal of our system is to create cognate sets, it can also be applied to the pairwise task; we compare our pairwise classifier with Kondrak's results in section 7.2.

Wang and Sitbon (2014) propose a method for the identification of cognates in a given text of a target language with those in a source language, to be used in the aid of language learners. The method is based on word sense disambiguation using BabelNet combined with classic string similarity measures. They test their method on six English web sources from different genres, in an attempt to identify words with French cognates which could be useful for a French learner who is learning English. A possible criticism of their approach is that it assumes the existence of a multilingual sense based lexicon such as BabelNet, and relies on having a way of translating from the target language to the source language.

2.4 Cognate Clustering With Semantic Information

Finally, we come to the most difficult version of automatic cognate identification, where sets of cognates are created using orthographic or phonetic information along with semantic information (the Word-Sets task).

Kondrak et al. (2007) extend the work of Kondrak (2004) in an attempt to create cognate sets from language dictionaries. They utilize the phonetic and correspondence scores from Kondrak (2004), combined with a simple semantic score based on the first words of definitions or the entire definitions being identical. Cognate sets are formed by using graph-based algorithms on connected components. They apply their method on a subset of Totonacan languages, and have an expert linguist evaluate their results. Of the 430 possible cognate sets proposed by their system, 89% were deemed to be true cognate sets. A possible criticism of their approach is that their measure of semantic similarity is a fairly simple heuristic, which cannot take into account slight semantic drift or variations in definition construction. As there is no complete gold annotation available for this language data, a direct comparison with our system is not possible.

Steiner et al. (2011) attempt a fully automated approach to the comparative method, including cognate set identification and language phylogeny construction. Since they allow cognate sets across non-identical definitions, the cognate clustering stage of their pipeline can be considered as the Word-Sets task. They combine a pairwise scorer inspired by the dynamic programming algorithms of bioinformatics with a graph based clustering approach. Clusters are then filtered based on semantic similarity and regular sound correspondences. They test their approach on the Tsezic and Mataco-Guaicura language families and report that their output phylogenies are consistent with the opinion of most experts. However, they do not quantitatively analyze their proposed cognate sets and instead rely on the manual inspection of an expert linguist. A possible criticism of their approach is that their semantic similarity is based on statistics of how often certain definitions correspond to certain word forms and makes no use of the actual content of the definitions themselves. They also use several hand-tuned thresholds throughout their pipeline rather than leveraging any machine learning techniques on training or development sets. Their system is not available for download, and the data sets they test on have no corresponding gold cognate annotations; therefore, we are unable to make a meaningful comparison with their system.

Chapter 3

Data Sets

Our experiments involve three different language families: Algonquian, Polynesian, and Totonacan. In this chapter, we describe these language families, the data sets that we are working with, and the pre-processing done by our system on the data.

3.1 Language Families

Algonquian is a subfamily of Native American languages, which consists of around 30 languages. Our first data set was compiled by Hewson (1993) and normalized by Kondrak (2002). The data set consists of four dictionary lists and is fully annotated with gold cognate information. Hewson used this data set to reconstruct thousands of Proto-Algonquian words. The fact that the dictionaries are fully annotated allows us to evaluate the effectiveness of our system on the Algonquian data set.

The second data set corresponds to a version of POLLEX, a large-scale comparative dictionary of over 60 Polynesian languages (Greenhill and Clark, 2011). The POLLEX project began over fifty years ago, with the goals of recreating Proto-Polynesian and generating insights into Polynesian prehistory with knowledge gained through historical linguistics. Unlike the Algonquian data set, the POLLEX data set is organized strictly as cognate sets, not as fully annotated dictionaries. Therefore, this data set is useful for training models and learning cognate relationships, but it cannot be used as a testing set for our full system.

The final data set consists of 10 language dictionaries from the Totonacan language family spoken in Mexico. No complete gold cognate annotation exists for this data set. And unlike in the other two data sets, where the definitions are written (primarily) in English, the definitions in the Totonacan data set are written in Spanish. Our system is applicable to definitions written in any language [1], provided that all definitions within a given language family are written in the same language. Since the Totonacan data is yet to be fully analyzed by linguistic experts, it provides an important motivation for developing our system.

[1] In the event that a data set contains definitions written in a low-resource language, certain features of our system would need to be turned off. The overall system, however, would still function.

Family	Languages	Entries	Cognate Sets	Unique Definitions
Algonquian	4	26,985	3661	22,747
Polynesian	62	26,699	3690	17,138
Totonacan	10	43,073	?	33,118

Table 3.1: Data set statistics.

Language	Word	Definition
Niuean	utu	pull up taro
Rennell	utu	harvest; get food
Samoan	utu	dig up; as yams and arrowroot
Tongan	utu	reap; harvest yams

Table 3.2: Example Polynesian cognate set.

In general, it is difficult to find data sets suitable to the task of clustering cognate sets across language dictionaries (the Word-Sets task). The data sets used for clustering cognate sets within word lists (the Form-Sets task) often contain only a couple hundred unique definitions, whereas our data sets of interest contain thousands (see Table 3.1). For example, the Indo-European Lexical Cognacy Database[2] contains over 34,000 words from 163 languages, but is limited to only 225 word meanings. Cognate relationships are therefore restricted to fall within a limited number of word lists of short, simple concepts, such as FOOT, ONE, or RUN. This is in contrast to the three data sets that we are using, where one cognate set can be composed of various definitions. Table 3.2 shows one such cognate set from the POLLEX data set.

3.2 Data Processing

Entries in the data include a word form and a corresponding definition. Definitions undergo cleaning between the raw data and our system. First, any part of a definition within parentheses are removed since this is often just a source of the data entry, a clarification, or a grammatical note. For example, the Menominee word *anæ:m* has the definition "dog (archaic vocative)," which after cleaning simply becomes "dog." Dashes, forward-slashes, and back-slashes that occur within a word in a definition are replaced by spaces. Consider the Algonquian word "onakekkokamik," whose definition "bark-lodge" is cleaned to become "bark lodge." In the last steps of our definition cleaning, punctuation is removed, any leading or trailing white spaces are stripped, and any occurrences of multiple white spaces in a row are truncated into a single space. Note that all data cleaning is automated and reproducible.

The Polynesian data set undergoes further cleaning. The small number of entries

[2]http://ielex.mpi.nl/

with definitions written in French are discarded, since our system is only capable of comparing two definitions written in the same language. Word forms that contain a parenthesis or a space are ignored, while any characters after a forward slash in a form are discarded. All forms that contain a dash as the first or last character are considered to be an affix, and any set with at least one affix is discarded.

Chapter 4

Methods

In this chapter, we explain the approach of our system. Section 4.1 introduces supervised classifiers and support vector machines, and in the following two sections, we describe the features used in our *general* and *language-specific* models, respectively. Following this, we describe our clustering approach, which uses the pairwise scores from the classifiers to create cognate sets, the end goal of our system.

4.1 Support Vector Machines

Our system makes use of pairwise classifiers to make cognate judgments between word pairs. We employ a support vector machine (SVM), which is a machine learning tool commonly used in the realm of natural language processing. SVMs utilize supervised learning, that is, they require labelled training data and learn a model that can predict the label of previously unseen data. The training data consists of positive and negative examples, i.e. input that exhibit or do not exhibit the classification in question. In our case, the SVM receives as input pairs of words and definitions labelled as cognates and pairs of words and definitions labelled as non-cognates. Once the model is learned, a new pair can be given to the model and a prediction on whether or not these two words are cognates, along with a confidence score, is produced.

SVMs work by mapping examples into a high dimensional space based on defined features. They learn a maximum margin hyper-plane in this space in an attempt to create a large gap between the positive and negative examples on either side of the hyper-plane. New examples are first mapped into this same high dimensional space and then classified based on which side of the hyper-plane they fall into. We make use of the software package SVM-Light to implement our classifiers (Joachims, 1999).

Language	Word	ASJP	Definition
Cree	isiyiːhkaːteːw	isiyihkatew	he names him so
Fox	išitehkaːtamwa	iSitehkatamwa	he names it so
Ojibwa	išinikkaːn	iSinikkan	call, name someone thus

Table 4.1: Example Algonquian words. Used to help illustrate the model features.

4.2 General Classifier

In this section, we describe the features of our *general* classifier, which are based on the definitions and phonetic structures of the words being compared. We call it the general classifier because the features learned are not specific to the two languages that the pair of words belong to. The model can be learned from any two related languages and used to classify words from a different pair of languages, even from another language family.

The full list of features can be seen in Table 4.2 and are described in the following two sub-sections. To help illustrate the features, we use the three Algonquian words and definitions from Table 4.1 as a running example.

4.2.1 Phonetic Based Features

The first category of features are based on the phonetic shape of the words being compared. To deal with the phonetic structure of a word, we convert words into ASJP format, a phonetic representation which can be viewed as a simplified version of the IPA (Brown et al., 2008). A phonetic representation enables word comparisons to be done between the underlying sounds of the words, rather than between the orthographic representations. ASJP has 41 possible characters, whereas the IPA itself has over 150. This reduced character set lends itself better to use with traditional all-or-nothing string similarity metrics, such as Edit Distance. An IPA encoding may not be able to determine the overall similarity between two words that a broader ASJP encoding can pick up on. Consider the IPA symbol *ː*, which denotes that the previous vowel is long, and leads, for example, to *e* and *eː* being considered different vowels under an IPA representation. ASJP lacks this subtle distinction and would consider both vowels to be *e*, allowing their similarity to be immediately visible to any string similarity metric. Table 4.1 displays the ASJP representation of the three words, which will be used throughout this section.

Edit distance, also known as Levenshtein distance, is a metric of string similarity that measures how many insertions, deletions, and replacements it takes to transform one string into another. (Levenshtein, 1965). This metric has been historically used in the task of cognate identification. For an example, consider transforming *isiyihkatew* into into *iSitehkatamwa*. The edit distance is six, using the following steps from left to right: (1) replace *s* with *S*; (2) replace *y* with *t*; (3) replace *i* with *e*; (4) replace *e* with *a*; (5) insert an *m*; (6) insert an *a*. Note that we can also think of transforming *iSitehkatamwa* into *isiyihkatew* instead, and the edit distance would remain as six, but with the opposite operations.

This leads us to the first of our phonetic features, *Normalized edit distance*. Normalized edit distance (NED) divides edit distance by the length of the longer string and inverts this score by subtracting it from 1. The reason for inversion is mainly conceptual; we choose the convention that a higher score for a feature indicates that a word pair has higher similarity. We divide by the length of the longer word so that words are not punished simply for being long. For example, *cat* and *dog* have an edit distance of 3, as do *support* and *supportive*, but it is clear that the second pair display more similarity. The NED of these pairs are 1 - (3/3) = 0 and 1 - (3/10) = 0.7, respectively, indicating that the first pair are completely different words, while the second pair are not so different. Looking again at *isiyihkatew* and *iSitehkatamwa*: they have an edit distance of six, and the longer *iSitehkatamwa* has a length of 13. This leads to an *Normalized edit distance* between them of 1 - (6/13) = 0.54.

Our next feature is based on common subsequences between the two words. A subsequence can be created from a string by deleting certain characters and leaving the order of the remaining characters unchanged. Therefore, a common subsequence can detect similarity between two words even if the words have lost or gained phonemes over time. *LCSR*, the longest common subsequence ratio (Melamed, 1995), measures the ratio of the longest common subsequence of the words versus the length of the longer one and is our next phonetic based feature. The longest common common subsequence between *isiyihkatew* and *iSitehkatamwa* is *iihkatw*, leading to an *LCSR* feature value of 7/13 = 0.54.

As words change over time, consonants are more stable than vowels. Therefore, words that have a consonant in the same position are more likely to have originated from the same word. To determine the number of aligned consonants between words, we first align them using the ALINE phonetic aligner (Kondrak, 2002). Our feature, *Consonant match* returns the number of aligned consonants normalized by the maximum number of consonants in either word. In the case of *iSitehkatamwa* and *iSinikkan*, we get the aligned consonant pairs *S-S*, *t-n*, *k-k*, *t-k*, and *m-n*, yielding a feature value of 5/7 = 0.71. Note that the ALINE phonetic aligner technically uses its own phonetic representation, not ASJP, but for simplicity, the alignment was shown in ASJP.

Along with creating a phonetic alignment, ALINE provides an *Alignment score*, reflecting the overall phonetic similarity between the two words. This score is used as the final phonetic feature of the model. ALINE is able to detect phonetic similarity between similar phonemes, even if the phonemes are not identical. This is in contrast to measures such NED or LCSR, which depend on exact phoneme matches.

Normalized edit distance
LCSR
Consonant match
Alignment score
Definition match
Definition content match
Normalized word-level edit distance
Content overlap
Synonym overlap
Content inflectional overlap
Content inflectional Synonym overlap
Content inflectional hyperonym overlap
Content inflectional hyperonym overlap
Average word vector cosine similarity
Content average word vector cosine similarity

Table 4.2: General model features. Phonetic features are above the horizontal line, while semantic features are below.

4.2.2 Semantic Based Features

In this section, we explain the various semantic features that are based on the definitions of the words being compared. Several of the semantic features are binary, meaning they receive a value of 1 or 0 depending on whether they occur or not. A feature receiving the value of 1 is also sometimes referred to as that feature *firing*. We use the three definitions listed in Table 4.1 to help demonstrate some of the features.

The first of our semantic features, *Definition match*, fires if any of the definitions of one word is an exact match with any of the definitions of the other word. Definitions are split on semi colons and commas, and each part of the split is treated as a separate definition for the word. In our example, this feature does not fire between any of the three definitions.

Stop words are words that are removed before performing certain natural language processing tasks. The words chosen to be part of the stop word list can vary depending on the task being performed, but they generally include "function" words, i.e. words that do not carry semantic meaning in themselves, and the language's most common words. The idea is that these stop words add little to a sentence's overall meaning and therefore do not help discriminate between sentences. For this reason, words that are not stop words are often referred to as *content words*.

Our second semantic feature, *Definition content match*, will fire if any definition of one word is an exact match of any definition of the other word when only considering content words, i.e. once stop words have been removed. After removing stop words, "he names him so" and "he names it so" simply become "names." The two parts of “call, name someone thus”, split on the comma, become “call” and “name.” This is because “he,” “him,” “so,” “it,” “someone,” and “thus” are stop words. *Definition content match* fires between “he names him so” an “he names it so”.

Next, we use the maximum *Normalized word-level edit distance* between the definitions as a feature. This is similar to the phonetic feature, normalized edit distance, described in section 4.2.1; however, by *word-level*, we mean that the words of the definitions themselves can be replaced, inserted, or deleted, rather than the characters making up the entire definitions. To see how this feature is calculated, consider transforming “he names him so” into “he names it so”. The transformation requires only the word-level replacement of “him” into “it.” This edit distance of 1 is normalized by the length of the longer definition, 4, and then finally inverted to yield a normalized word-level edit distance of 0.75. When a definition, such as “call, name someone thus”, contains multiple parts, we choose the maximum score between any of the parts and another definition.

Our next feature, *Content overlap*, fires if two definitions have at least one content word in common. Due to the common content word “names,” this feature fires between “he names him so” and “he names it so”, but it does not fire between “call, name someone thus” and either of the others.

The semantic features described thus far are able to detect similarities between definitions if they resemble each other on the surface, such as by having words in common. We would also like to detect connections between definitions even if no obvious similarities are present. To find such relationships, we must consider the actual meanings of the definitions and their constituent words. In an attempt to detect these deeper semantic similarities between definitions, our next set of semantic features utilize WordNet (Fellbaum, 1998). WordNet is a large, open-source database containing information about tens of thousands of words and their relations between each other. The main relation contained in WordNet is synonymy: when two words share exactly or nearly the same meaning. Word synonyms are grouped into what WordNet refers to as *synsets*. e.g. {sword, blade, brand, steel} compose one such synset. WordNet also contains information between synsets. Hyponymy (also called hyperonymy), is the most frequent relation which WordNet encodes between synsets. This relation links general synsets to increasingly specific ones and represents an “is a” relationship. For example, WordNet tells us that a *rapier* is a *sword*, which is itself a *weapon*. Another relation encoded in WordNet which is useful for our task is that of related inflectional form. Given the word *jump*, for example, we can search WordNet and find the related inflectional forms: *jumped*, *jumping*, and *jumps*. With WordNet at our disposal, we create several other features, which indicate the level of semantic similarity between two definitions.

The first of these features is *Content inflectional overlap*, which determines if one definition contains a content word that is an inflectional variant of a content word in the other definition. This feature fires between “he names him so” and “call, name someone thus” as well as between “he names it so” and “call, name someone thus”, due to fact that “names” is an inflectional variant of “name.”

Content synonym overlap indicates when one definition has a content word that, according to WordNet, is a synonym of a content word in the other definition. In our example, none of the definition pairs fire this feature since “names” is not a synonym of itself, or with “name” or “call.”

Note that “call” and “name” are indeed synonyms, but the inflectional *s* on “names” stops *Content synonym overlap* from firing. Our next semantic feature, *Content inflectional synonym overlap*, makes up for this limitation. It fires if the synonym of an inflectional form of a content word in a definition is a synonym of an inflectional form of a keyword in a definition of the other word. Therefore this feature fires between “he names him so” and “call, name someone thus” and between “he names it so” and “call, name someone thus”.

The next two semantic features, *Content hyperonym overlap* and *Content inflectional hyperonym overlap*, are analogous to the previous synonym-based features but with hyperonym relationships instead. In the case of our examples, these features do not fire between any of the definitions. *Content inflectional hyperonym overlap* would fire between two definitions if one contained the word “sword” and the other contained the word “weapons,” for example.

The features based on information from WordNet can detect deeper semantic relationships than those simply based on word-level similarity; however, they fail when there is not an obvious synonym or inflectional variant in common between definitions. These features are dependent on WordNet’s database, which, while extensive, can never be absolute. Semantic similarities that these features cannot detect include derivational variants, such as “wise” and “wisdom,” or paraphrases. To extract these non-obvious similarities, our final two semantic features take advantage of advances in word vector technology. Word2Vec, the word vector implementation of Mikolov et al. (2013), uses a neural network to map words into a high dimensional real-numbered vector space based on the context of the words in a large corpus. Words that appear in similar contexts in the corpus are mapped to vectors with close proximity in the vector space. For English, we use word vectors of dimension 300 that were pre-trained on almost 100 billion words and made freely available by Mikolov et al. (2013). Word2Vec includes functions for determining the cosine similarity between two vectors, which in turn relays information about how similar the underlying words are. One can even determine the similarity between groups of words; this is calculated by first averaging the vectors for each group and then finding the cosine similarity between these two average vectors. In order to calculate this score, we use the package *gensim* (Řehůřek and Sojka, 2010), which provides a Python wrapper for Word2Vec.

The first word vector feature in our model is the average cosine similarity between all words in the definitions, and the second feature is similar but only considers the content words of the definitions. We call these features *Average cosine similarity* and *Content average cosine similarity*, respectively. “he names him so” and “he names it so” receive an average cosine similarity of 0.93 considering all words, and a full 1.0, when considering only content words (indeed, they each contain the same content word, so a maximum cosine similarity is expected). “he names him so” obtains an average cosine similarity of 0.51 with the first part of “call, name someone thus”, i.e. “call,” and an average cosine similarity of 0.78 with the second part, i.e. “name someone thus.” We select the maximum value of 0.78 for the feature value. This score of 0.78 is a high cosine similarity, which shows that word vectors are able to pick up on the semantic similarity between two definitions, even when such definitions have no words in common.

4.3 Regular Sound Correspondences

The features described above are language independent in the sense that features indicating that two words or two definitions share similarities can work to identify cognate relationships between any pair of languages. We will even show that such a model can be trained on one language family and used to classify cognates in a completely unrelated language family. However, we would also like to take into account cognate information that is specific to pairs of languages, namely regular sound correspondences, as described in Chapter 1. For example, *th/d* is a correspondence between English and German, occurring in words such as *think/denken* and *leather/Leder*. A model trained on another language family would not be able to learn that a corresponding *d* and *th* is an important indicator of cognation in German/English pairs.

For each language pair, we derive a *specific model* by implementing the approach of Bergsma and Kondrak (2007). The features of this model are pairs of substrings, with one part of the pair coming from each word being compared. If a given substring pair occurs often between cognate words in two languages, then the model is able to learn that that substring pair correlates with cognation between those languages.

The features are created by first appending boundary characters, ^ and $, to signify the beginning and end of each word and then aligning the words using the alignment induced by edit distance. A substitution cost of two is used for edit distance, following Bergsma and Kondrak (2007), which means that a replacement costs twice as much as an addition or subtraction. Two types of features are created. The first type, *phrase* pairs, are substring pairs that are *consistent* with the alignment, up to a given maximum substring length. For a substring pair to be consistent with the alignment means that if an aligned character from one of the words belongs to the substring pair then that character’s aligned partner from the other word must also belong to the substring pair. Including non-aligned characters to a substring pair does not make that pair inconsistent. We restrict the size of the phrase features to be at most three. Bergsma and Kondrak (2007) found that a substring size of three allows for important correspondences to be found, while keeping the calculation of features tractable. The second type, *mismatch* pairs, allow the model to learn mismatching segments between cognates that are longer than the maximum

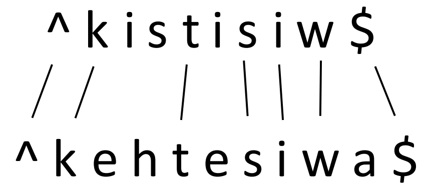


Figure 4.1: Example alignment induced by edit distance. Used to determine the substring-based features of the specific models.

substring length. They contain all of the unaligned characters between two aligned end points. Two types of mismatch features are created: one including the aligned end characters, and one including just the unaligned characters.

For an example, consider the Cree word *kistisiw*, meaning "he is great, important," and the Fox word *kehtesiwa*, meaning "he is big, old," which are a cognate pair. The alignment produced by edit distance for this pair is shown in Figure 4.1. This alignment leads to phrase pair features such as ^ki-^ke, st-hte, t-t, and iw-iw, and mismatch features such as kist-keht, is-eh, tis-tes, is-es, w$-wa$, and -a. These features could be created using the IPA or ASJP representation; in development it was seen that both ways produce similar results, so we default to the richer IPA.

In order to train the specific models, we need a substantial number of cognate pairs, which are not initially available in our unsupervised setting. We use a heuristic method to overcome this limitation. We create sets of words that satisfy the following two constraints: (1) identical dictionary definition, and (2) identical first phoneme. For example, the Cree and Menominee cognates *niːpaːtipisk* and *niːpaːtepæh*, both meaning "at night," will be placed together, while their Ojibwan cognate *niːpaːtipikk* will be missed due to its full definition being "at night, during the night." The resulting word sets are mutually exclusive, and contain mostly cognates (in fact, we use this method as our baseline in our experiments). We explored using the heuristic that two words are considered cognates if their LCSR is greater than or equal to 0.58, a threshold used in the cognate literature (Bergsma and Kondrak, 2007). However, a LCSR threshold is not a transitive relationship; one word could exceed this LCSR threshold with two other words that do not exceed the threshold with each other. Therefore, there is no simple way to use this measure as part of a heuristic clustering. Conversely, the identical first letter constraint does not come with this added complication.

We extract positive training examples from these high-precision sets, and create negative examples by sampling random entries from the language dictionaries. A separate specific model is learned for each language pair to capture regular sound correspondences. For example, there is a model with features for Cree-Menominee and another model for Fox-Ojibwa, etc. Note that the specific models disregard semantic information.

4.4 Cognate Clustering

The first step in our clustering approach is to use the general model from Section 4.2 to score pairs of words across languages. Featurizing all possible pairs of words from all languages is very time consuming, so we first filter out dissimilar word pairs that obtain a normalized ALINE score below 0.35. This threshold allows the system to run in a reasonable amount of time, while, in development experiments, we observed that over 95% of cognate pairs exceed this threshold. Next, we use the appropriate specific model to classify all word pairs that were positively classified by the general model. The final pairwise score is an average of the score given by the two models. This allows for words to be clustered that have a high likelihood of being related, as judged by the correspondences learned from the specific model, even if their similarity determined by the general model is not as high. Theoretically, the weight of each model score could be tuned; however, we elect to default to an equal weight for each model score in order to keep our system robust and to prevent over-tuning on a specific data set.

Once pairwise scores have been computed, we cluster words into sets with a version of the UPGMA (Unweighted Pair Group Method with Arithmetic Mean) clustering algorithm (Sokal and Michener, 1958), which has been used in previous work on cognate clustering (Hauer and Kondrak, 2011; List, 2012; List et al., 2016). First, each word is placed into their own cluster. The score between clusters is computed as the average of all pairwise scores between the words within those clusters. In each iteration, the two clusters with the highest average score are selected to be merged, and the scores between the merged cluster and all others are re-calculated. The algorithm terminates when no two clusters have a positive average score. In order for the system to run with a reasonable amount of memory usage, only positive scores are included in the pairwise similarity matrix. In effect, merges are only allowed between clusters where all pairwise scores between them are positive, making it a more conservative clustering approach.

Chapter 5

Discontinued Methods

Chapter 4 explained our final system for producing cognate sets, which combines a pairwise score with an average-score based clustering algorithm. In this chapter, we discuss previous approaches that we attempted. The results of these methods were surpassed by the system described in Chapter 4, but it is interesting to present them as negative results and to see where they went wrong.

Our previous method involved the use of the general and specific classifiers of Sections 4.2 and 4.3, but used a very different strategy in creating cognate sets than that of Section 4.4. We used a pipeline constructed of three stages to create proposed cognate sets. The first stage assembled words into preliminary *nucleus* sets of presumed cognates. The second stage added remaining words from the language dictionaries into these nucleus sets based on the general model. Finally, the third stage added even more words to the nucleus sets based on the specific models. We also experimented with an alternative approach of learning regular sound correspondences and using them to generate proposed cognates.

Note that the following chapters of this thesis do not depend on the methods described in this chapter, and as such, it is possible for the reader to omit this chapter without hindering their understanding of the remainder of our work.

5.1 Nucleus Sets

The first stage involves creating *nucleus* sets of proposed cognates. These sets are created via two approaches, which ostensibly result in high-precision cognate sets as a starting point.

Heuristic Sets: Our first approach to creating nucleus sets uses the heuristic that words with identical first letters and identical definitions are grouped together. This is the same heuristic used in Section 4.3 in order to create training data for the specific models.

Double-Match Sets: To create additional nucleus sets, we apply the general classifier described in Section 4.2 to pairs of words that have not been placed into a nucleus set by the heuristic. Similarly to the clustering method described in Section

4.4, words must pass a 0.35 ALINE threshold to be classified. We use the pairwise classifications to inform our next approach at creating nucleus sets. Our method assigns two words to a new nucleus set if the pair receives a positive score from the model and are a *double-match*, that is, they are each other's respective top scoring word. Moreover, their definitions must also have a content word or inflectional variant of a content word in common.

5.2 General Model Additions

The next stage of the pipeline begins by constructing a *proto-word* for each Stage 1 set, which can be thought of as the proposed *etymon*, or ancestor word, in which each cognate originated from. Each word in the set is converted into ASJP format, and then aligned using a multi-alignment algorithm inspired by techniques used in bioinformatics. From the alignment, we select the most common letter in each position (possibly NULL) and combine these letters to give us the proto-word. In the event that there is a tie for the most common letter in a position, we choose the one that occurs most frequently throughout the language dictionaries.

For nucleus sets constructed via the heuristic approach, the definition assigned to the proto-word is simply the shared common definition of all words in its set (the heuristic assures that all words have the same definition). For the Stage 1 sets constructed using double-match words, there is an additional step in determining the proto-word's definition, since the definitions of the two words need not be identical in this case. We create the proto-word definition using all common words between the two double-match definitions. Additionally, the double-match pairs also may contain words with inflectional variants in common; in these cases, the lemma of these word forms is added to the proto-word definition. For example, the definitions "it blows ashore" and "is blown ashore" are intersected to create the new definition "blow ashore." Notice that "blow" is the lemma of "blows" and "blown."

Once we have these proposed proto-words, our next strategy is to analyze the remaining words and to add them to a nucleus set when applicable. To achieve this, the words that are not part of any nucleus set are classified against each created proto-word by using the general model (an ALINE threshold of 0.35 is used as usual). In this way, the proto-words are used as a representative for each set. A word is added to the set represented by the highest scoring proto-word, assuming the pair has a positive classification. At the end of General Model Additions, we have the heuristic and double-match nucleus sets, along with any additions to these sets based on the general model scores.

5.3 Specific Model Additions

The third stage is similar to the second stage, in that additional words are added to an already-created cognate set. Starting from the output sets of stage two, we next compare the as-of-yet unplaced dictionary words against the words belonging to a set. We do not compare a word for a specific set if that set already contains a word from the same language. Once more we use the ALINE threshold of 0.35 to determine which word pairs are considered by the model. All word pairs passing this threshold are classified using the corresponding specific model for that language pair. To determine which words are added, we calculate the average score between the dictionary word and all words in a given set. A word is added to the set with the highest average score, as long as that score is greater than 0.

5.4 Analysis of the Stages

The pipeline approach, as outlined above, was able to find some cognate sets; however, the results were surpassed by the methods described in Chapter 4. One explanation is that using proto-words, while linguistically appealing, is an implicitly information-losing process. The proto-word is created to act as a representative for each cognate set, but in doing so, we lose information about the individual words making up each proposed cognate set. In double-match sets, the proto-word definition may end up different from either of the definitions of the words in those sets, and in both kinds of Nucleus Sets sets, the constructed ASJP representation of the proto-word, by nature of its creation, cannot contain the phonetic information of all words in its set. The average score based clustering approach described in Section 4.4, leverages the pairwise scores between all word pairs, so is not susceptible to this information loss.

Another key difference is that our final method is able to utilize the general and specific models while composing the cognate sets, but General Model Additions and Specific Model Additions use these models in a post-hoc fashion to add to already constructed sets. Moreover, the pipeline approach is not able to merge or partition the Nucleus Sets sets once they have been created, so any errors get propagated through each stage of the system without a chance for correction. Conversely, our final clustering method does not require any nucleus sets and by its very nature allows merging of cognate sets during construction.

5.5 Regular Sound Correspondences Through Alignment

This section outlines an alternative attempt to extract and apply regular sound correspondences, without using the substring features of the specific models. This method was designed to closely emulate the comparative method. We use the output of General Model Additions as a starting point, so this procedure can be thought of as an alternative to Specific Model Additions.

For the sake of this method, we assume the system is being used as a tool for an expert linguist. The cognate sets proposed by General Model Additions are manually annotated by the expert linguist, partitioning each proposed set into true cognate sets. If there are more than one word from a given language, a word is chosen randomly from that language, so that each cognate set contains at most one word per language.

These annotated sets are then converted into ASJP format and aligned using the same multi-alignment algorithm used to create proto-words. From the aligned sets, we determine regular sound correspondences. For each position in the alignment we create a correspondence, where the correspondence contains a letter for each language in the set. When a language does not belong to a given cognate set, then the correspondences produced by that set will not represent that missing language. A correspondence can also be multiple characters long if those characters are either all vowels or all consonants for each language represented in the correspondence. Figure 5.1 provides an example Algonquian annotated cognate set along with the correspondences created from the set in this fashion. The "*" characters within each correspondence signify that no Fox word belongs to this set.

The correspondences from all such cognate sets are then clustered together based on pairwise similarity. The similarity between two correspondences is determined by adding a score for each language in the data set. If a correspondence contains the same characters for a given language, then a score of 1 is added to the similarity. If the correspondence differs for a given language, then a score of -1 is added to the similarity. And finally, if either correspondence does not represent a given language, then similarity is not affected. We use the same average-score-based clustering algorithm for correspondences as is used in the cognate clustering of our final method, described in Section 4.4. The goals of clustering are: (1) to associate correspondences that do not represent a given language with ones that do represent that language; (2) to reduce noise from correspondences that are not common throughout the cognate sets.

For each cluster, a *summary* correspondence is created based on the frequencies of characters in each correspondence within the cluster. From the summaries, we create regular expressions, which are then used to find missing cognates from sets that do not contain a given language. For example, there is no Fox cognate in the example set shown in Figure 5.1. We start with the first correspondence, c1, and look up its cluster, C_1. This cluster contains other correspondences from various cognate sets, such as "Cree:*n* Fox:*n* Menominee:*n* Ojibwa:*n*," and "Cree:*n* Fox:*nw* Menominee:*n* Ojibwa:*," and many others. The summary correspondence from C_1 is "Cree:*(n|t|m)* Fox:*(n|nw)* Menominee:*(n|m)* Ojibwa:*n*." This means that in all the correspondences contained within C_1, the characters representing Fox that occurred with a non-negligible frequency are *n*, and *nw*. Therefore, the regular expression provided by this cluster is *(n|nw)*. Following this procedure for each correspondence yields an overall regular expression of *(n|nw)(e|i)(ht)(a)(m|mw)*, and we predict that the missing Fox cognate matches this pattern. If the regular expression matches with a word in the Fox language dictionary, then that word is added to the cognate set. In this case, no such word exists in the Fox dictionary,

Language	ASJP	Definition
Cree	nistam	first
Menominee	nEqtam	first in time, before, earliest
Ojibwa	nittam	first, for the first time

Correspondence	Cree	Fox	Menominee	Ojibwa
c1	n	*	n	n
c2	i	*	E	i
c3	st	*	qt	tt
c4	a	*	a	a
c5	m	*	m	m

Fox regular expression: (n|nw)(eli)(ht)(a)(m|mw)

Figure 5.1: Top: An example Algonquian cognate set. Middle: The correspondences produced by a multi-alignment. Bottom: The regular expression for the missing Fox cognate.

so no additional word is added to the cognate set. Coincidentally, the gold cognate set for this example does not actually contain a Fox word, but this did not affect whether a prediction was created and found or not.

Unfortunately, this method was unable to find many cognates. One probable reason for this is that the method used to create predictions is very "brittle." That is, if even one character of the regular expression is incorrect, then the prediction will not be found in the language dictionaries. There is no mechanism to compare the similarity of a prediction to words that actually exist in the dictionaries. Another downside to this method is that when every correspondence within a cluster does not represent a given language, then no prediction is possible from the correspondences involved in that cluster. Finally, this procedure could only be used in an attempt to add missing words to a pre-existing cognate set. The approach of our final system uses information regarding regular sound correspondences to inform clustering decisions, rather than in a post-hoc manner.

Chapter 6

Evaluation Metrics

One key consideration for this project is how to evaluate the output cognate sets. None of the previous work on the Word-Sets task, which we are working on, systematically and quantitatively evaluated their outputted cognate sets, and therefore, there is no precedent (Kondrak et al., 2007; Steiner et al., 2011). There are, however, several metrics that have been proposed for evaluating cognate clustering within word lists (the Form-Sets task) and clustering in general. Indeed, the question of how to evaluate a general clustering is not conclusively answered, even without taking cognation into account. In this chapter we explore several possible clustering metrics and analyze their applicability to cognate clustering across language dictionaries. While we consider many evaluation metrics, we argue that the number of found sets, balanced with cluster purity, provides a good criteria for evaluating cognate clusterings.

In pairwise cognate identification (the Form-Pairs and Word-Pairs tasks), system output is generally evaluated by determining the recall, precision, and F-Score. Recall is defined as the number of true cognate pairs that are correctly identified as cognates. Precision measures how many of a system's predictions are true positives, i.e., how many times did a system claim a pair are cognates when they actually are. In general, a system's performance is a trade-off between recall and precision. A system that proposes all pairs as cognates would trivially receive a recall score of 100%, while its precision score would be near zero. On the other hand, a system that only predicts that one pair are actually cognates and gets this prediction correct would achieve a precision score of 100% but a recall of just one over the number of actual cognate pairs. Clearly, a system will need to sacrifice some precision to improve recall and vice versa. To quantify this balancing act and get a measure of a system's overall performance, the metric known as F-Score is often used. A common version of F-Score is defined as

$$\text{F-Score} = \frac{2 * P * R}{(P + R)}, \tag{6.1}$$

which is the harmonic mean of precision (P) and recall (R).

These measures cannot be immediately applied to our task of cognate clustering. Some thought needs to go into the question of what constitutes the recall and precision of a clustering, and whether these measures make intuitive sense when evaluating the quality of a clustering. In the following sections, we discuss some metrics that can be considered when evaluating cognate clusterings. The chapter concludes with an in depth example, which analyzes multiple clusterings and their performance according to the various metrics. We define a *cluster* to be a single set of words (all deemed to be cognate with each other), whereas a *clustering* is the total partitioning of all words into various clusters.

6.1 Pairwise Metrics

The first metric we consider when evaluating cognate clusterings is pairwise precision and recall. This metric has seen some use in the cognate literature (Hall and Klein, 2011; List, 2012; List and Moran, 2013). Each cluster induces a set of pairs, with each pairwise combination of elements within that cluster contributing a pair. The pairwise recall of a proposed clustering C is the percentage of pairs induced by the gold clusters that are also induced by the clusters of C. Similarly, the pairwise precision of C is the percentage of pairs induced by C that are also induced by the gold clustering. By considering all the induced pairs in this way, we relate the problem of evaluating clusters back to the identification evaluation versions of recall and precision, i.e. a list of pairs of words to be evaluated. Let $\mathrm{pairs}(C)$ be a function that takes a clustering and returns the set of all pairs induced by it, and let G represent the gold clustering. Then formally:

$$\text{Pairwise-Recall}(C, G) = \frac{|(\mathrm{pairs}(C) \cap \mathrm{pairs}(G)|}{|\,\mathrm{pairs}(G)|} \tag{6.2}$$

$$\text{Pairwise-Precision}(C, G) = \frac{|\,\mathrm{pairs}(C) \cap \mathrm{pairs}(G)|}{|\,\mathrm{pairs}(C)|} \tag{6.3}$$

As an example, suppose a gold clustering G = $\{a_1,\ a_2\}$, $\{b_1,\ b_2,\ b_3\}$, and a proposed clustering C = $\{a_1,\ a_2,\ b_2\}$, $\{b_1,\ b_3\}$. Then $\mathrm{pairs}(G)$ = $\{a_1a_2,\ b_1b_2,\ b_1b_3,\ b_2b_3\}$, and $\mathrm{pairs}(C)$ = $\{a_1a_2,\ a_1b_2,\ a_2b_2,\ b_1b_3\}$. We can see that $\mathrm{pairs}(C) \cap \mathrm{pairs}(G)$ = $\{a_1a_2,\ b_1b_3\}$, leading to a pairwise recall of $2/4 = 50\%$ and a pairwise precision of $2/4 = 50\%$.

A downside to using pairwise metrics when evaluating cognate clusterings is that they are biased towards larger sets, as the number of induced pairs grows quadratically with the size of a set. For example, a set containing 10 words will induce 45 pairs and hence contribute as much as 45 two-word cognate sets to the final scores. A cognate clustering system that correctly identifies 45 such clusters, while missing the 10-word cluster, will only achieve 50% pairwise recall for this portion, even though intuitively the "recall" of such a system appears much higher.

6.2 MUC

Another metric that could be used to evaluate a cognate clustering system is MUC (from the Message Understanding Conference) (Vilain et al., 1995). MUC was originally designed to score co-reference algorithms but could be used to evaluate other kinds of clusterings, such as in our task. MUC assigns precision, recall and F-Score based on the number of *links* created in the proposed clusters versus the links created by the gold clusters.

Following Vilain et al. (1995), but in the context of our cognate task, let S be a cognate set in the gold G, and let C be a proposed clustering. Next, define $c(S)$ to be the number of correct links needed to connect S. We can see that $c(S) = |S| - 1$, by simply linking one element with the next. Now let $p(C, S)$ be the intersection of S with all sets in C containing at least one element of S. Vilain et al. (1995) refer to $p(C, S)$ as the partition of S relative to the proposed clustering C. We can then think of the number of links of S missing from C as the number of links required to reunite the sets of $p(C, S)$. Hence, the missing links of S, $m(C, S)$, can be calculated as $m(C, S) = |p(C, S)| - 1$. To get the MUC recall of a clustering relative to a single gold cluster, we have the equation:

$$\begin{aligned}\text{MUC-Recall}(C, S) &= \frac{c(S) - m(C, S)}{c(S)} \\ &= \frac{(|S| - 1) - (|p(C, S)| - 1)}{|S| - 1} \\ &= \frac{|S| - |p(C, S)|}{|S| - 1}\end{aligned} \tag{6.4}$$

Finally, to extend this notion of recall to the entire gold clustering, it is sufficient to sum the numerator and denominator for each cluster. Put another way, the MUC recall of a clustering C is the total number of links necessary to generate all sets of G found by C. And more formally:

$$\text{MUC-Recall}(C, G) = \frac{\sum_{S_i \in G}(|S_i| - |p(C, S_i)|)}{\sum_{S_i \in G}(|S_i| - 1)} \tag{6.5}$$

To make this more concrete, consider an example where G = {a_1, a_2}, {b_1, b_2, b_3}, {c_1}, and C = {a_1, a_2, b_2}, {b_1, b_3, c_1}. $\text{MUC-Recall} = ((2 - 1) + (3 - 2) + (1 - 1))/((2 - 1) + (3 - 1) + (1 - 1)) = 2/3 = 66.6\%$.

We can use this example to explore the notion of MUC precision. In general, precision is the notion of how often a system's output is correct when it proposes a desired outcome. With MUC, we are considering links, and therefore, precision is how many links proposed by a clustering are actual links within the gold. In the example, b_2 is erroneously linked with a_1 and a_2 in the proposed clustering, and c_1 is linked with b_1 and b_3 in error. The links between a_1 and a_2 and between b_1 and a_3 are found in the gold, so can be deemed correct. Therefore the total MUC precision is $2/4 = 50\%$. With some thought, we can see that to calculate the MUC Precision of a proposed clustering, it is sufficient to swap the roles of the clustering and the gold in the MUC-Recall formula.

A critique of MUC, illuminated by Bagga and Baldwin (1998), is that it will award the same precision for words placed incorrectly into a small cluster as words placed incorrectly into a large cluster, while intuitively, the second error is more detrimental. For example, consider a gold clustering G = {a_1, a_2, a_3, a_4, a_5}, {b_1, b_2}, {c_1, c_2, c_3, c_4, c_5}, and two proposed clusterings: C_1 = {a_1, a_2, a_3, a_4, a_5}, {b_1, b_2, c_1, c_2, c_3, c_4, c_5} and C_2 = {b_1, b_2}, {a_1, a_2, a_3, a_4, a_5, c_1, c_2, c_3, c_4, c_5}. At a glance, C_2 looks like a much worse clustering; all of the a_i elements are grouped with all of the c_j elements, indicating many false pairwise associations. Perhaps counter-intuitively, both C_1 and C_2 receive a MUC-Precision score of 90%.

Bagga and Baldwin (1998) also remark that MUC does not give any credit for a proposed clustering correctly separating out singletons, i.e. those elements that do not belong in a set with any others. However, this appears to be an artifact from the conventions of co-reference chain evaluation, which MUC was originally designed for. In a clustering task, it is straightforward to see that MUC-Precision will be higher for a clustering that correctly places a singleton element into its own set (or equivalently, does not place it into a set at all). For example, imagine that c_1 was placed correctly into its own set in the first example from this section. Then there would be one fewer incorrect link, and precision would rise to 2/3 = 66.6%.

6.3 B-Cubed

B-Cubed is a further metric originally created for co-reference scoring, in response to the downsides of MUC (Bagga and Baldwin, 1998). B-Cubed looks at the recall and precision of each element relative to its own set, and the final statistic is an average of all of these individual scores. This metric has previously been used to evaluate cognate clusterings (Hauer and Kondrak, 2011; List et al., 2016).

Consider an element i that belongs to the gold cluster G_i and has been placed into the proposed cluster C_i. The element-wise recall and precision of i are computed as follows:

$$\text{B-Cubed-Recall}(i, C, G) = \frac{|G_i| \cap |C_i|}{|G_i|} \tag{6.6}$$

and

$$\text{B-Cubed-Precision}(i, C, G) = \frac{|G_i| \cap |C_i|}{|C_i|} \tag{6.7}$$

Then the final metrics for the entire clustering are:

$$\text{B-Cubed-Recall}(C, G) = \sum_{i=1}^{N} w_i * \text{B-Cubed-Recall}(i, C, G) \tag{6.8}$$

and

$$\text{B-Cubed-Precision}(C, G) = \sum_{i=1}^{N} w_i * \text{B-Cubed-Precision}(i, C, G), \qquad (6.9)$$

where N is the total number of elements, and w_i is the weight assigned to element i. Bagga and Baldwin (1998) note that different weighting schemes can be used to tailor the metric to a specific task; however, the use of B-Cubed to evaluate cognate clustering has so far only considered the case of equal weights for all elements (Hauer and Kondrak, 2011; List et al., 2016).

Amigó et al. (2009) provide an extensive analysis on various evaluation metrics for clustering. They show that B-Cubed satisfies their four constraints for a sensible clustering metric. These constraints are as follows:

1. Cluster Homogeneity: A metric should reward a system for splitting up a cluster with elements from multiple gold sets into multiple clusters each containing only elements from the same gold set.

2. Cluster Completeness: A metric should reward a system for merging two clusters which contain only elements from the same gold set.

3. Rag Bag: A metric should reward a system for placing an outlier element into a cluster which already contains elements from various gold sets, rather than placing that element into a *clean* cluster which only contains elements from one gold set. The intuition of this constraint is that "introducing disorder into a disordered cluster is less harmful than introducing disorder into a clean cluster." (Amigó et al., 2009)

4. Clusters size vs. quantity: A metric should reward a system for making a small error in a large cluster as opposed to making a small error in multiple small clusters.

The first two constraints are very intuitive. It seems that any clustering evaluation metric should satisfy these constraints. In fact, Amigó et al. (2009) performed an empirical study on human participants, where the participants were shown clustering pairs specifically designed to highlight a specific one of the four constraints, and at least 90% of the participants preferred a clustering where the first two constraints were upheld. Perhaps surprisingly, the participants showed even stronger support for the second two constraints.

After assessing the empirical "validity" of their constraints, Amigó et al. (2009) then went on to use them as a meta analysis for various clustering metrics. They found that B-Cubed is the only metric that satisfies all four of their constraints. It was this evidence that first lead Hauer and Kondrak (2011) to apply B-Cubed as their metric of choice to the word list cognate clustering task.

A downside of the B-Cubed metric is that it awards an element-wise recall and precision for an element i for each element of G_i, including i itself. This means that an element will always be awarded recall and precision points for "finding itself," even if that element does not belong to a cognate set with any other word, or even if it is not placed into a cluster with any other word in the proposed clustering. For example, consider the gold clustering $G = \{a_1, a_2\}, \{b_1\}, \{c_1\}$ and a proposed clustering $C = \{a_1\}, \{a_2\}, \{b_1\}, \{c_1\}$. C is a clustering where no words have been proposed as cognates, and yet its recall is calculated as $\frac{1}{4} * \frac{1}{2} + \frac{1}{4} * \frac{1}{2} + \frac{1}{4} * \frac{1}{1} + \frac{1}{4} * \frac{1}{1} = 75\%$. Moreover, the precision of this proposed clustering is $\frac{1}{4} * \frac{1}{1} + \frac{1}{4} * \frac{1}{1} + \frac{1}{4} * \frac{1}{1} + \frac{1}{4} * \frac{1}{1} = 100\%$. Intuitively, a clustering that proposes no cognate sets should not be rewarded with high scores like this.

For cognate evaluation, a weighting scheme that gives singleton elements zero weight seems to make more sense. If a word does not belong to a cognate set, then it should not provide positive recall to the overall score. Under a zero-weight for singletons scheme, the same example as above now scores a recall of $\frac{1}{2} * \frac{1}{2} + \frac{1}{2} * \frac{1}{2} + 0 * \frac{1}{1} + 0 * \frac{1}{1} = 50\%$. Precision is not changed. The recall score is now lower, although it can still be considered counter-intuitive that a clustering that proposes no cognate sets has a reported recall of 50%.

6.4 Found Sets

While the above-mentioned metrics can be used to evaluate clusterings in general, we believe that a more appropriate measure of the utility of a cognate identification system is to count the total number of *found sets*. The main purpose of an automatic cognate clustering system is to aid an expert linguistic in the comparative method, or to assist automated systems in the creation of language phylogenies (Bouchard-Côté et al., 2013). In this regard, having a large number of identified cognate sets, even with a certain degree of false positives, is the desired outcome. For an expert linguist to filter out false positives from proposed cognate sets is still far less time consuming than for the linguist to have to discover cognate sets "from scratch."

We consider a set to be *found* if any of the words within the set are clustered together by a system. A distinction can also be made between partially and completely found sets. Arguably, the number of partially found sets is more important, as it is easier for a human expert to extend a found set to other languages than to discover the set in the first place. In fact, a discovery of a single pair of cross-lingual cognates demonstrates the existence of a corresponding proto-word in the ancestor language, which is likely to have descendant words in the other languages of the family.

Our proposed *found sets* measure, which roughly corresponds to recall, must, however, be balanced against precision. For example, a trivial strategy that clusters all words together would be considered 100% accurate according to the number of found sets. We argue that reporting the number of found sets along with a metric of precision is an effective way of measuring the quality of a cognate clustering.

6.5 Cluster Purity

Purity is precision metric that has been used to evaluate cognate clusterings (Hall and Klein, 2011; Bouchard-Côté et al., 2013). Each word in a given cluster is assigned to a gold set based on which gold set the majority of that cluster's words belong to. Then purity is calculated as the fraction of total words assigned to the correct gold cluster. Put another way, a given cluster's purity corresponds to the largest possible overlap between that cluster and any gold cluster. More formally, let $G = \{G_1, G_2, ..., G_g\}$ be a gold clustering and $C = \{C_1, C_2, ..., C_c\}$ be a proposed clustering. Then

$$\text{purity}(C, G) = \frac{1}{N} \sum_{i=1}^{c} \max_j |G_j \cap C_i|, \tag{6.10}$$

where N is the total number of elements. For example, consider the gold clustering G = $\{a_1$, $a_2\}$, $\{b_1$, b_2, $b_3\}$, $\{c_1\}$, and the proposed clustering C = $\{a_1$, a_2, $b_2\}$, $\{b_1$, b_3, $c_1\}$. $\text{purity}(C, G) = \frac{1}{6} * (2+2) = 4/6 = 66.6\%$. In this case, the largest overlap is achieved by intersecting the first cluster of C with the first cluster of G and the second cluster of C with the second cluster of G. This purity score of 66$\%$ makes nice intuitive sense, since each cluster in C has two out of three words belonging to the same set in G.

Purity is not perfect as a stand-alone measure for determining the overall quality of a clustering. Amigó et al. (2009) note in their study of various cluster metrics, that purity does not reward merging clusters composed of elements from the same gold cluster. Consider the simple example where G = $\{a_1$, a_2, a_3, $a_4\}$ and C = $\{a_1$, $a_2\}$, $\{a_3$, $a_4\}$. This clustering achieves a purity score of 100$\%$, while it is clear that the clustering would benefit from merging its two proposed clusters. Since purity is a measure of precision, it must always be reported along side a measure of recall.

6.6 Toy Example

Next, we present a toy example with evaluation scores for the metrics discussed in this chapter. Multiple clustering variations are shown in table 6.1. The gold clustering is shown in the first row, representing how the elements should actually be clustered. Each s_i represents a singleton element, and for all other elements, those with the same letter belong to the same cluster in the gold. The different proposed clusterings are described below:

- Max Recall: This naive clustering places all elements into a single cluster.
- Max Precision: This naive clustering places (almost) all elements into their own cluster. c_4 and c_5 are placed into the same cluster for convenience; several of the metrics would have divide-by-zero problems if every element is assigned its own set.
- Decent: The decent clustering finds all gold sets but is noisy.

Clustering	Sets
Gold	$\{s_1\}$, $\{s_2\}$, ..., $\{s_8\}$, $\{a_1, a_2\}$, $\{b_1, b_2, b_3\}$, $\{c_1, c_2, c_3, c_4, c_5\}$
Max Recall	$\{s_1, s_2, ..., s_8, a_1, a_2, b_1, b_2, b_3, c_1, c_2, c_3, c_4, c_5\}$
Max Precision	$\{s_1\}$, $\{s_2\}$, ..., $\{s_8\}$, $\{a_1\}$, $\{a_2\}$, ..., $\{c_3\}$, $\{c_4, c_5\}$
Decent	$\{a_1, a_2, s_1, s_2, b_1\}$, $\{b_2, b_3\}$, $\{c_1, c_2, c_3, s_3, s_4, s_5\}$
Good	$\{a_1, a_2, s_1\}$, $\{b_1, b_2, b_3, s_4\}$, $\{c_1, c_2, c_3, s_2, s_3\}$

Table 6.1: Example of various clusterings of a given gold clustering.

Clustering	Pairwise	MUC	B^3	$B^3 - 0W$
Max Recall	16.8	58.0	24.9	34.9
Max Precision 13.3	25.0		77.6	50.7
Decent	25.0	47.0	66.0	61.7
Good	42.4	62.5	75.8	72.9

Table 6.2: F-Score values of the example clusterings.

- Good: The good clustering also finds all gold sets but even more gold elements and is less noisy than the Decent clustering.

Table 6.2 contains F-Score results for the various metrics on the different clusterings. Based on our intuition, we would hope that the Good clustering is given a high score, which is higher than all other clusterings, by a possible metric. However, we see that this is not always the case.

Stunningly, B-Cubed prefers the Max Precision clustering over the Good clustering. In fact, the Max Precision clustering, which does not place any cognates together, receives a B-Cubed F-Score of close to 80%. This can be attributed in part to B-Cubed giving "free" recall for every element identifying itself as a cognate. For example, no matter where a singleton element is placed in a clustering, it will receive an element-wise recall of 100%. Moreover, placing an element into its own cluster guarantees an element-wise precision of 100%. These facts combine to produce overly generous B-Cubed results, and this problem would only get worse for a gold clustering with more singletons. B-Cubed with zero-weighted singletons gets a somewhat more reasonable score of 50.7% for the Max Precision clustering, but this is still highly unintuitive. We take this as strong evidence that B-Cubed, while receiving use in the cognate clustering literature, is not the appropriate metric to evaluate our task.

A glaring problem with the evaluations given by MUC, is that the Max Recall clustering, which places all words into one cluster, receives an F-Score of 58%, only 4.5% lower than the Good clustering. This means that a naive clustering, which does nothing to identify cognates, receives a relatively high score. For this reason, we discard MUC as a viable metric.

Pairwise F-Score does rank the proposed clusterings in a reasonable fashion;

Clustering	Purity	Found Sets
Max Recall	27.8	100.0 (100.0)
Max Precision	100.0	33.3 (0.0)
Decent	66.7	100.0 (33.0)
Good	77.8	100.0 (66.6)

Table 6.3: The percentage of found sets and cluster purity of the example clusterings. The percentage of complete found sets are in parentheses.

however, the Good clustering only receives a score of 42.4%. Even though the Good Clustering correctly identifies all of the a_i and b_i cognates and three of the five c_i cognates, it only receives a Pairwise Recall of 50%. This highlights the fact, mentioned in Section 6.1, that Pairwise metrics are biased towards larger sets, whereas in a cognate clustering task, the total number of found sets is arguably more useful. The low scores given by Pairwise F-Score do not accurately reflect the end goal of a cognate clustering system, and therefore we choose not to use this metric to evaluate our methods.

Cluster purity and the percentage of found sets are shown in Table 6.3. Since cluster purity only measures precision, it can not be used in isolation. This is evident by the Max Precision clustering receiving a purity score of 100%. Purity does, however, rank the rest of the clusterings in an intuitive order. When we combine purity with the number of found Sets, a clearer picture emerges. For example, the Max Precision clustering only finds a third of the cognate sets and fully finds zero of the sets. On the other hand, the Max Recall clustering fully finds 100.0% of cognate sets, but its cluster purity lags behind that of other clusterings. Note that as the number of cognate sets increases, the Max Recall clustering's purity and the Max Precision clustering's found sets would approach zero; their non-negligible purity and found sets, respectively, are strictly due to the small overall size of the current toy example. Based on the measures displayed in Table 6.3, namely purity and the percentage of found sets, one can determine how well each clustering performs, both relatively and absolutely. It is clear that the Good clustering performs well and is in fact better than all others. The Max Precision and Max Recall clusterings are shown to be useless by the found sets and purity metrics, respectively, and the Good clustering outperforms the Decent clustering in all regards, as expected. The Good clustering at least partially finds all cognate sets, while at the same time maintaining a cluster purity of close to 80%. We argue that the description provided by purity in conjunction with the number of found sets gives a more accurate representation of the quality of clusterings than any of the F-Score measures shown in Table 6.2. Moreover, these two metrics cover each other's weaknesses; purity is biased towards placing each word into their own cluster, and the number of found sets is biased towards placing all words into the same cluster. This balancing act is reminiscent of the precision/recall trade-off described in the beginning of this chapter. Finally, an added bonus to using the number of found sets and cluster purity is that they are relatively easy to interpret compared to other metrics. For these reasons, we choose to report these two metrics when evaluating our system's output in the next chapter.

Chapter 7

Experiments

In this chapter, we explain our experimental setups, and discuss our results. In the first section, we explain how we train our general model. In the next section, we present our first experiment, where we test our general and specific classifiers against a state-of-the-art Word-Pairs system. After this, we move on to our main experiment: testing our full system on the Algonquian language dictionaries and evaluating our proposed cognate sets. Finally, we discuss our full system results and analyze the errors made by our system.

7.1 Training the General Model

As described in Chapter 3, we have three data sets at our disposal: for the Algonquian, Polynesian, and Totonacan language families. Since the Algonquian data set is fully annotated with gold cognate data, we use it to evaluate the effectiveness of our system. Therefore, in development, we chose to train our general model on the Polynesian data and test it on the Totonacan data. By doing this, we were able to determine which features were beneficial without over-tuning on the Algonquian testing data. Also, this was our first experiment to determine if a model trained on one language family is able to properly classify a completely unrelated language family.

The Totonacan dictionaries lack full cognate annotation; however, preliminary output of our system while in development was annotated by a Totonacan expert, providing us with several hundred annotated cognate sets. From these annotations, we created a pairwise development set, including all possible 6755 cognate pairs and 67,550 randomly selected non-cognate pairs. This development set was used to experiment with the features of the general model by evaluating a general model that was trained on the Polynesian data.

To create a training set from the Polynesian data, we randomly selected 25,000 cognate and 250,000 non-cognate word pairs. When creating cognate pairs, we take into account information about the relatedness of the languages within the POLLEX data set. Similar languages, as defined as being in the same subfamily, are not used to construct cognate examples. This is because words in the same cognate set from similar languages were observed to often have very similar, if not identical, forms and definitions. By disregarding similar language word pairs, we attempt to remove "easy" pairs from the training data. While the examples are chosen randomly, the system will assure that each cognate set in the POLLEX data is used at least once to create a cognate pair, as long as that set does not contain only similar language pairs. We also experimented in development with using an ALINE score threshold on example negative pairs in an attempt to make them more *competitive* (Bergsma and Kondrak, 2007); however, this was shown to actually slightly decrease performance on the development test set when compared with not using any threshold for negative examples. The procedure explained here is the same one used to train the general model in our full system setup.

When testing our final Polynesian-trained general model on the Totonacan development set, it achieved a pairwise F-Score of 88.4%. This finding is highly promising, since it shows that a cognate classification model need not be trained on the same language family that it is classifying. This is useful since the language families of interest to historical linguists generally do not have annotated gold data to train on (indeed the goal of an automated cognate identification system is to help produce such annotation). Moreover, this result confirmed that our system can function on data sets where definitions are written in a language other than English (in this case Spanish), even when the model was trained on a data set where definitions are written in English.

7.2 Pairwise Classification

While the end goal of our system is to take input language dictionaries and produce cognate sets as output (the Word-Sets task), a subset of our full system, namely the general and specific classifiers, can be applied to the Word-Pairs task as well.

In our first experiment, we compare the effectiveness of our pairwise classifiers against the system of Kondrak (2004), which was designed to process one language pair at a time. Following Kondrak (2004), we classify the noun subset of the Algonquian data. As their system contains no machine learning component, it requires no training data, but the Cree-Ojibwa noun pair was used for development and tuning. We train two versions of our general model: one trained on the Cree-Ojibwa noun subset, and another on the POLLEX data set. We also train a specific model for each language pair, using the method described in Section 4.3.

After we classify the pairs, they are outputted in order of confidence at being cognates. This output is evaluated using 11-point interpolated precision. Starting from the top of the list (the word pairs deemed most likely to be cognates by the classifier), the precision is recorded at different recall thresholds. At each recall threshold in 0, 0.1, 0.2, ..., 1.0, the corresponding precision is recorded. The precision scores are interpolated, which means that the precision at recall level r is set to be the highest possible precision for any recall level r' where $r' >= r$. Interpola-

Language Pair	Kon2004	FullSystem-CreeOjibwa	FullSystem-Polynesian
Cree Ojibwa	78.7	84.8	82.3
Cree Fox	69.8	77.8	76.6
Cree Menomini	61.8	78.4	80.5
Fox Menomini	65.2	81.7	81.8
Fox Ojibwa	69.5	83.3	79.3
Menomini Ojibwa	64.1	80.3	81.7
Average on Test Sets	66.1	80.3	80.0

Table 7.1: System results: 11-Point interpolated precision on Algonquian noun pairs.

Language Pair	GeneralModel-CreeOjibwa	GeneralModel-Polynesian	Specific-Model
Average on Test Sets	79.4	77.0	45.9

Table 7.2: System ablation: 11-Point interpolated precision on Algonquian noun pairs.

tion is motivated by the fact that if precision can be increased while also increasing recall, there is no reason not to move on to the higher recall level. In effect, interpolation ensures that the precision-recall curve is monotonically decreasing. Finally, these eleven precision values are averaged to give one summative metric detailing how well the system classified the pairs at different levels of recall. By definition, we set the precision at recall of 0% to be 100% and the precision at a recall of 100% to be 0% (Kondrak, 2004).

7.2.1 System Comparison

Table 7.1 contains the 11-point precision results, with each row containing the score for a given language pair. As a baseline, Kon2004 displays the results reported by Kondrak (2004). To make a direct comparison, we follow Kondrak (2004) when reporting the average results on the testing language pairs by simply taking the arithmetic mean of the five scores.[1] FullSystem-CreeOjibwa displays the results when combining the score of the general model, trained on Cree-Ojibwa noun pairs, and the scores of the specific models, trained on the respective language pairs. FullSystem-CreeOjibwa outperforms Kon2004 on every language pair, for an average improvement of 14.2% and an error reduction of 41.9% on the test sets. This improvement is most likely attributable to the richness of our feature set and to the use of machine learning in our system, as opposed to Kon2004, which applies a

[1]Technically it may make more sense to combine the output for each language pair into one file to determine the overall score across all language pairs; however, this distinction results in a less than one percent difference in average score for each of our system configurations.

manual tuning of weights based on the training set.

While it is nice to see such positive results when training on the Cree-Ojibwa noun pairs, it is not a fully realistic scenario to have gold cognate data to train on in the same language family that we are classifying. It is perhaps more interesting to see what kind of results are achievable when training the model on a completely unrelated language family. It is a fair assumption to make that we have this kind of gold data available and does not go against the unsupervised categorization of our task.

FullSystem-Polynesian shows the results of combining the scores of the general model trained on Polynesian and the specific models. As expected, the results are not as good when training the general model on Polynesian as compared to training on Cree-Ojibwa (since the Cree-Ojibwa data is from the same language family and same original source); however, the scores are still higher than Kon2004 for all language pairs and by an average of 13.9% on the testing sets. In fact, FullSystem-Polynesian even surpasses Kon2004 on Cree-Ojibwa itself, which is impressive since Kon2004 was tuned on this language pair. These results are more evidence that a model trained on one language family can still achieve impressive results when classifying cognates in a completely unrelated language family. Moreover, when both general models are combined with the specific models, the one trained on Polynesian was able to score on average on the test sets within 0.3% of the model trained on Cree-Ojibwa.

7.2.2 System Ablation

In Table 7.2, we report the average 11 point interpolated precision on the noun test sets when only using the general model or only using the specific models in isolation.

GeneralModel-CreeOjibwa displays the results of only using the score from the general model trained on the Cree-Ojibwa noun pairs. The average result on the test sets of 79.4% is 0.9 percent lower than when combining with the specific model scores in the full system (as shown in Table 7.1). GeneralModel-Polynesian reports the results of using the general model, trained on Polynesian data, without information from the specific models. This result shows that the specific model is able to increase the score of the general Polynesian model by three percent on average on the test sets. Both cases provide evidence that correspondence-based language-specific knowledge can be leveraged to aid a general model. It is perhaps intuitive that the specific models do more to assist GeneralModel-Polynesian than GeneralModel-CreeOjibwa, since the latter may already be biased towards the Algonquian family.

Finally, Table 7.2 shows that the specific models, when used in isolation, do not fare as well. In fact, the average specific model score on the test sets is 20.2% lower than Kon2004. These system ablation results indicate that a combination of the general model with the specific models leads to a pairwise classifier that is better than either model on its own.

We have shown that our classifiers combine to form a state-of-art pairwise cognate identification system when considering phonetic and semantic information (the Word-Pairs task).

7.3 Cognate Set Identification

In our main experiment, we apply our full system to the task of creating cognate sets from the entire Algonquian dictionaries (the Word-Sets task). We train the general model on the Polynesian data set, according to the description provided in Section 7.1, and follow the clustering methods outlined in Section 4.4.

We run our system on a 3.5GHz machine with 12 CPUs and 32GB of memory. Featurizing and training the general Polynesian model and the specific models takes only a few minutes. Featurizing and classifying all dictionary pairs with the general and specific models takes a matter of hours. Finally, cognate clustering is completed in under an hour.

We report results for four configurations of our system, based on two parameters: (1) whether the pairwise scores are calculated solely by the general model or combined with the specific model scores; (2) whether the system allows proposed cognate sets with multiple words per language or restricts each set to one word per language.

To get a measure of how well our full system performs, we need something to compare it against. Limited work has been done on the task of clustering cognates across definitions (Kondrak et al., 2007; Steiner et al., 2011); however, no direct comparison is possible due to the lack of gold annotation on the data sets used and system code being unavailable for download. As a baseline, we compare our system against the heuristic sets of Section 4.3. These sets are created following the two heuristics: (1) words must have an identical definition; (2) words must have an identical first letter. We also report the results obtained with LEXSTAT (List, 2012).[2] While this system was designed for clustering cognates within word lists (the Form-Sets task), we apply it to language dictionaries in part to observe how well a state-of-the-art word list system performs on this task, and to serve as another baseline to compare our system against. LEXSTAT has no capability to propose cognates with distinct definitions, so we first group together the words that have identical definitions and provide these as its input.

7.4 Results

In Chapter 6, we discussed various possible metrics that could be used for evaluation. The B-Cubed metric has been used recently to evaluate cognate clustering within word lists (Hauer and Kondrak, 2011; List and Moran, 2013; List et

[2]We use the code from `http://lingpy.org` with default parameters.

System	Found Sets	Purity
Heuristic Baseline	18.9 (9.9)	96.4
LEXSTAT	19.6 (10.5)	97.1
General (1 Word Per Language)	66.2 (51.3)	66.5
General + Specific (1 Word Per Language)	63.1 (48.2)	70.3
General	71.6 (57.9)	52.1
General + Specific	67.9 (53.6)	57.7

Table 7.3: System results: The percentage of found sets and cluster purity for the Algonquian language dictionaries. The number of fully found sets is shown in parenthesis.

al., 2016), but we argued that this metric is not appropriate to measure clusterings across language dictionaries, since it produces highly unintuitive results. To verify this claim, we evaluate the *Max Precision* clustering, which was introduced in Chapter 6, on the Algonquian language dictionaries. Recall that this clustering places all words into their own cluster, in effect, identifying zero cognate relationships. Max Precision achieves a B-Cubed F-Score of 89.6% on this data set. To emphasize: a clustering that does nothing to find cognates is awarded with an extremely high score by B-Cubed. This result is strong evidence supporting our claim that B-Cubed is not the proper metric to evaluate this task. We argue that the total number of sets found by a system (a measure of recall), balanced by the cluster purity (a measure of precision), provides an accurate depiction of how well a system identifies cognate sets. Table 7.3 displays the number of found sets and cluster purity for the baselines and system configurations.

The Heuristic Baseline is able to find 18.9% of gold cognate sets while scoring 96.4% for cluster purity. The high purity score is partially attributable to the number of words placed into their own cluster. However, even just counting the clusters that contain multiple words still yields an average cluster purity of 69%. LEXSTAT performs slightly better than the Heuristic Baseline, but the number of sets found by LEXSTAT is still limited by the fact that it cannot propose cognates between words with varying definitions. In fact, in the Algonquian data set, only 21.4% of all gold cognate sets contain at least two words with identical definitions, making this an upper bound on the number of found sets possible for any word list cognate identification system.

Moving next to our system configurations: All four configurations are able to find more than three times as many cognate sets as the baselines. By allowing more than one word for a given language in a cluster, the number of found sets predictably rises, but at a cost of purity. Combining the scores of the general model with the specific models lowers the percentage of found sets while increasing purity. The choice of which configuration to use comes down to whether one values recall (the number of found sets) or precision (cluster purity) more. For optimal recall, a configuration composed of the general model for pairwise scores and a clustering method that allows multiple words per language works best, finding 71.6% of total sets (57.9% being completely found). This configuration still maintains a cluster purity of 52.1%, meaning that on average, just over half of each proposed cluster actually belong to the same cognate set in the annotated data. To maximize precision (while still finding a large number of cognate sets, unlike the baselines), we use the specific model scores in conjunction with the general model and only allow one word per language per cluster. This configuration is able to find 63.1% of total cognate sets and achieve a cluster purity of 70.3%. Regardless of the exact configuration chosen, our system is able to find a majority of true cognate sets while keeping cluster purity at a relatively high level.

7.5 Discussion

Including the specific model scores leads to system configurations that have higher purity than those that only use the general model scores. This shows that the regular correspondences learned by the specific models help to discriminate some non-cognates that the general model, due to phonetic or semantic similarity, believes to be cognates. These non-cognates could include false cognates such as lexical borrowings or chance resemblances. However, while purity increases by including the specific models, the overall improvement is somewhat negligible. One possible explanation is that the Algonquian languages are in general quite similar to each other, and therefore, the general model is able to discern most cognate relationships alone. For example, the specific models learn regular correspondences such as *s:s* or *hk:kk*, which the general model is already able to determine are similar phonemes. It is possible that for a more distantly-related language family, the specific models would provide more utility to the system. On the other hand, in the pairwise experiment of Section 7.2, it was seen that the specific models were able to help the Polynesian general model, so the problem could actually be due to how the two scores are combined in the full system. Due to memory constraints, only positive scores from the general or specific models are allowed into the similarity matrix before clustering (see Section 4.4), so word pairs that receive a negative score from one of the two models cannot be clustered together regardless of the other model's score. For example, Table 7.4 shows a cognate set with three words found by the system but with one cognate missing. The missing Menominee word "ehkuah" is given a negative score with the other words by the specific models and therefore is placed alone. If the system were able to include negative scores while clustering, then this missing cognate would likely be included in the proper clustering due to the high general model scores.

Table 7.4 also displays another type of error made by the system, again in regard to the specific models. They are trained on the heuristic sets where all words must begin with identical first letters, and this leads them to prefer words with the same first letter, to a fault. As mentioned in Section 4.3, the first letter heuristic is useful in that it is transitive, simple, and leads to high-precision training sets; however, it

	Language	Word	Definition
Found Set	Cree	ihkwa	louse
	Fox	ihkwa	louse
	Ojibwa	ikkw	louse
Missing Word	Menominee	ehkuah	louse

Table 7.4: Example system error due to the specific model.

	Language	Word	Definition
Found Set	Cree	wisakisiwin	bitterness, pain
	Ojibwa	wissakisiwin	bitterness
Missing Word	Menominee	weqsakæsen	sickness

Table 7.5: Example system error due to semantic drift.

is not perfect. In this example, the first-letter bias stops a cognate from being found by the system.

Another category of error made by our system occurs when large semantic drift has taken place between cognates. An example of such error is shown in Table 7.5, where the definition "sickness" no longer shares a similar meaning with "bitterness" or "pain." It is possible that this word could have been identified through regular sound correspondences, but in this case, the general model score was negative, prohibiting the clustering.

It should be noted that while semantic drift can lead to errors in our system, there are many instances where our system identifies such occurrences. Found cognates of this nature are due in large part to the word vector features of our model. Table 7.6 provides several examples of cognate sets found by our system that would have been extremely difficult to identify without word vector technology. The table includes cognates that have undergone small or large semantic drift, such as between "he is angry" and "he is nauseated" or "he is in mourning" and "she is widowed." And it also includes cognates that have similar meanings but a connection that is not obvious to simpler measures of semantic similarity, such as "he is in bits" and "he is ground up."

Another type of error made by our system originates from the complex morphology of the Algonquian language family. Due to its polysynthetic morphology, a single Algonquian word can express a meaning that would take several words to express in English. This manifests itself in the data set as distinct cognate sets that only slightly differ in their overall definitions and phonetic forms. These similar sets prove very difficult for our system to correctly cluster. For example, consider the example shown in Table 7.7. The Menominee word *a:kuaqtæ:hsen*, meaning "he is in the shade," has been placed into a cluster with two similar Cree and Ojibwa words (shown above the horizontal line); however, it should have been placed with

Language	Word	Definition
Cree	wanaːtisiw	he is inconsistent
Menominee	wanaːtesew	he is uneasy
Cree	maːyaːčiteːheːw	he is angry
Menominee	mianaːčetæhæːw	he is nauseated
Cree	pistahtam	he eats it by mistake
Ojibwa	pittantaːn	bite something accidentally
Cree	wiːhtikoːw	gigantic man eating monster, windigo
Fox	wiːtekoːwa	owl
Cree	piːsisiw	he is in bits
Ojibwa	piːssisi	he is ground up
Cree	ayiwiskaweːw	he is taller than someone else
Ojibwa	aniwiškaw	precede, surpass someone
Cree	siːkaːwiw	he is in mourning
Menominee	seːkawew	she is widowed

Table 7.6: Example cognates found with the assistance of word vector information.

Language	Word	Definition
Cree	aːkawaːsteːsimoːw	he lies down in the shade
Menominee	aːkuaqtæːhsen	he is in the shade
Ojibwa	aːkawaːtteːššimoːn	be in the shadow
Ojibwa	aːkawaːtteːššin	make shadow with one's body

Table 7.7: Example system error due to morphology.

the Ojibwa word *aːkawaːtteːššin*, meaning "make shadow with one's body" (shown below the horizontal line). All the phonetic shapes and definitions in this example display high levels of similarity, making the true cognate relationships difficult to discern. It could be argued that these similar morphological variants are technically cognates by definition; none the less, the gold annotations make cognate distinctions based on morphology, and this contributes to a large amount of our system errors.

Another type of system error comes from clusters of likely cognates that do not actually appear in the gold annotation. These outputs decrease cluster purity, perhaps pessimistically. Some examples of this type of error are shown in Table 7.8. Any cognate identification system could be forgiven for making mistakes when the word forms and definitions share such striking similarity. It is possible that they are missing in error from the gold annotation. On the other hand, these may be examples of false cognates.

Finally, some errors made by our system are attributable to the ALINE threshold used to decide which pairs are classified. This constraint, as explained in Section 4.4, is needed to allow the system to run in a reasonable amount of time, but it

Language	Word	Definition
Menominee	pekuač	growing wild
Ojibwa	pekwači	growing wild
Cree	niso:te:w	twin
Ojibwa	ni:šo:te:nq	twin

Table 7.8: Examples of proposed cognate sets that are not found in the gold data.

comes with the price of not classifying some true cognates. However, this problem is mainly evident in the versions of the system that allow multiple words per language per cluster. There is a higher likelihood of words below the ALINE threshold preventing a desired cluster merge in the larger clusters created by these versions. Overall, this does not seem to contribute to many of the system errors.

Chapter 8

Conclusion

In this final chapter, we provide a summary of our work and findings. We also mention possible paths for future exploration and conclude with some final thoughts.

8.1 Summary

Identifying and clustering cognate sets within a family of related languages is a complicated problem with application in historical linguistics. Up to this point, most research has only considered the problem of clustering cognates within groups of words having identical definitions. In that task, semantic information from definitions is not critical, as cognate relationships are constrained to words with identical definitions. In our work, we explored the under-studied problem of finding and clustering cognate sets across entire dictionaries of related languages. This task necessarily requires the use of semantic information from the definitions themselves, since cognate sets need not be formed from words having identical definitions. This assumption is more realistic and requires less pre-processing when dealing with large language dictionaries containing thousands of unique definitions.

We have presented a machine learning model capable of learning cognate relationships based on a set of rich phonetic and semantic features. We also implemented a language-pair-specific model based on substrings of given words, in an attempt to extract regular sound correspondences. The model scores are combined with an average-score-based clustering algorithm to produce proposed cognate sets.

We investigated several evaluation metrics and have shown that B-Cubed, which has been used to evaluate cognate clusterings within word lists, is not suitable to evaluate cognate clusterings across entire language dictionaries. We argue that the total number of found cognate sets provides an important measure of the utility of a system, and we balance this metric with cluster purity.

In a pairwise cognate identification task, our models surpassed the state-of-the-art, with an error reduction of 41.9%. When applying our full system to the Algonquian language family, a majority of the true cognate sets were found, while maintaining a relatively high cluster purity.

Our work is the first to apply machine learning techniques to the task of cognate clustering across language dictionaries and the first, to the best of our knowledge, to apply advances in word vector technology to cognate identification in general.

8.2 Future Work

In the future, we would like to apply our full system to the Totonacan language family. This family has yet to be fully analyzed by an expert linguist, and as such, provides motivation for an automated system such as ours. Our system can be used as a tool to provide prospective cognate sets to a Totonacan expert, since manual annotation of proposed cognate sets requires far less work than analyzing entire language dictionaries by hand.

A possible approach to removing the first-letter bias from the specific models would be to use an iterative process. We could run the system with only the general model scores, and use these preliminary output sets as training data for the specific models. This procedure would mimic the comparative method, and could possibly lead to improvements in identifying regular sound correspondences.

Another direction for future work would be to use morphological information to guide cognate classifications, as our current system's inability to do so contributes to a large number of its errors.

8.3 Final Thoughts

A main insight from our work is that a cognate classification model can be trained on one language family and achieve impressive results when classifying a completely unrelated language family. This fact allows cognate information from a high-resource language family to guide cognate identification between languages that little is known about. In effect, this allows us to use supervised learning techniques on an unsupervised problem.

There are aspects of cognate identification that will most likely always require a language expert, such as the detection of cognates that have undergone both phonetic and semantic change or the detection of large-scale borrowing between languages. However, we believe that our system has made a large step in the field of automated cognate identification and can prove useful as a tool in the hands of expert linguists.

References

[Amigó et al.2009] Enrique Amigó, Julio Gonzolo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. In *Information Retrieval*, pages 461–486.

[Bagga and Baldwin1998] Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

[Beinborn et al.2013] Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2013. Cognate production using character-based machine translation. In *IJCNLP*, pages 883–891.

[Bergsma and Kondrak2007] Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 656–663, Prague, Czech Republic, June. Association for Computational Linguistics.

[Bouchard-Côté et al.2013] Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated reconstruction of ancient languages using probabilistic models of sound change. *PNAS*.

[Brown et al.2008] Cecil H. Brown, Eric W. Holman, and Soren Wichmann. 2008. Automated classification of the world's languages: A description of the method and preliminary results. In *Language Typology and Universals*, volume 61, pages 285–308.

[Ciobanu and Dinu2013] Aline Maria Ciobanu and Liviu P. Dinu. 2013. A dictionary-based approach for evaluating orthographic methods in cognates identification. In *Proceedings of Recent Advances in Natural Language Processing*, pages 141–147, Hissar, Bulgaria.

[Fellbaum1998] Christiane Fellbaum. 1998. *WordNet: An Eletronic Lexical Database*. MIT Press, Cambridge, MA.

[Greenhill and Clark2011] Simon J. Greenhill and Ross Clark. 2011. Pollex-online: The polynesian lexicon project online. In *Oceanic Linguistics*, volume 50, pages 551–559.

[Hall and Klein2011] David Hall and Dan Klein. 2011. Large-scale cognate recovery. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 344–354, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

[Hauer and Kondrak2011] Bradely Hauer and Grzegorz Kondrak. 2011. Clustering semantically equivalent words into cognate sets in multilinguial lists. In

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 865–873, Chiang Mai, Thailand.

[Hewson1993] John Hewson. 1993. *A computer-generated dictionary of proto-Algonquian*. Canadian Museum of Civilization, Hull, Quebec.

[Joachims1999] T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA.

[Kondrak et al.2007] Grzegorz Kondrak, David Beck, and Philip Dilts. 2007. Creating a comparative dictionary of totonac-tepehua. In *Proceedings of Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology*, SigMorPhon '07, pages 134–141, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Kondrak2002] Grzegorz Kondrak. 2002. *Algorithms for Language Reconstruction*. Ph.D. thesis, University of Toronto, July.

[Kondrak2004] Grzegorz Kondrak. 2004. Combining evidence in cognate identification. In *Proceedings of the Seventeenth Canadian Conference on Artificial Intelligence*, pages 44–59.

[Kondrak2012] Grzegorz Kondrak. 2012. Similarity patterns in words. In *The EACL 2012 Workshop on Uncovering Language History from Multilingual Resources*.

[Levenshtein1965] Vladimir I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710.

[List and Moran2013] Johann-Mattis List and Steven Moran. 2013. An open source toolkit for quantitative historical linguistics. In *Proceedings of the ACL 2013 System Demonstrations*, pages 13–18, Stroudsburg.

[List et al.2016] Johann-Mattis List, Philippe Lopez, and Eric Bapteste. 2016. Using sequence similarity networks to identify partial cognates in multilingual wordlists. Berlin, Germany.

[List2012] Johann-Mattis List. 2012. LexStat: Automatic detection of cognates in multilingual wordlists. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pages 117–125, Avignon, France.

[Melamed1995] Dan Melamed. 1995. Automatic evaluation and uniform filter cascades for inducing n-best translational lexicons. In *Proceedings of the Third Workshop on Very Large Corpora*.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Mulloni2007] Andrea Mulloni. 2007. Automatic prediction of cognate orthography using support vector machines. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 25–30, Prague, Czech Republic, June. Association for Computational Linguistics.

[Nakov and Tiedemann2012] Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 301–305. Association for Computational Linguistics.

[Rama2015] Taraka Rama. 2015. Automatic cognate identification with gap-weighted string subsequences. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1227–1231, Denver, Colorado, May–June. Association for Computational Linguistics.

[Řehůřek and Sojka2010] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

[Rosvall and Bergstrom2008] Martin Rosvall and Carl T Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

[Sokal and Michener1958] Robert R. Sokal and Charles D. Michener. 1958. A statistical method for evaluating systematic relationships. In *University of Kansas Science Bulletin*, volume 38, pages 1409–1438.

[Steiner et al.2011] Lydia Steiner, Peter F Stadler, and Michael Cysouw. 2011. A pipeline for computational historical linguistics. *Language Dynamics and Change*, 1(1):89–127.

[Swadesh1952] Morris Swadesh. 1952. Lexico-statistic dating of prehistoric ethnic contacts: with special reference to north american indians and eskimos. *Proceedings of the American philosophical society*, 96(4):452–463.

[Trask1996] R.L. Trask. 1996. *Historical Linguistics*. St Martin's Press, Inc., New York, NY.

[Turchin et al.2010] Peter Turchin, Ilia Peiros, and Murray Gell-Mann. 2010. Analyzing genetic connections between languages by matching consonant classes. In *Journal of Language Relationship*, pages 117–126.

[Vilain et al.1995] Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 45–52, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Wang and Sitbon2014] Haoxing Wang and Laurianne Sitbon. 2014. Multilingual lexical resources to detect cognates in non-aligned texts. In Gabriela Ferraro and Stephen Wan, editors, *The Twelfth Annual Workshop of the Australasia Language Technology Association*, pages 14–22, Melbourne, Australia, November.