Cognate Discovery to Bootstrap Lexical Resources

B.Tech Project Report presented by

Shantanu Kumar

2013EE10798

Under the guidance of

Dr. Sumeet Agarwal

Dr. Ashwini Vaidya

17th November 2016

Department of Electrical Engineering
Indian Institute of Technology Delhi

Acknowledgments

I wish to express my sincere thanks to Dr. Sumeet Agarwal for providing me with the opportunity to do this project. I am also grateful to Dr. Ashwini Vaidya. I am extremely thankful and indebted to them for sharing expertise, and sincere and valuable guidance and encouragement extended to me.

Shantanu Kumar

Contents

Abstract

The high level of linguistic diversity in South Asia poses the challenge of building lexical resources across the languages from these regions. This project is aimed at automatically discovering cognates between closely related language pairs like Hindi-Marathi or Hindi-Punjabi, in a scalable manner, which would assist the task of lexical resource creation. We would like to analyze a large part of the vocabulary for both languages, as opposed to small word lists used in most works so far. We also aim to do a linguistic analysis over the identified cognates to conclude whether lexical resources can be successfully shared between Hindi and related languages.

Chapter 1

Introduction

1.1 Motivation

Cognates are words across different languages that are known to have originated from the same word in a common ancestral language. For example, the English word '*Night*' and the German '*Nacht*', both meaning *night* and English '*Hound*' and German '*Hund*', meaning *dog* are cognates whose origin can be traced back to Proto-Germanic [13]. Cognates are not always revealingly similar and can change substantially over time such that they do not share form similarity. The English word '*wheel*' and the Sanskrit word '*chakra*' are in fact cognates which are traced back to '$*k^{w}ek^{w}elo$' from Proto-Indo-European.

Automatic cognate identification, in Natural Language Processing, refers to the application of string similarity algorithms with machine learning algorithms for determining if a given word pair is cognate or not. Identification of cognates is essential in historical linguistics. Cognate information has also been successfully applied to NLP tasks, like sentence alignment in bitexts [15], improving statistical machine translation models [8], inducing translation lexicons [11][16] and identification of confusable drug names [7]. It can also be used to bootstrap lexical resource creation for a language with low resources by finding parallels in related rich resource languages.

1.2 Outline

In this work, we have performed rigorous analysis over the existing state of the art models used for cognate identification to reveal their short comings and pitfalls and suggest possible improvements over these models to be implemented and tested in the second part of the project.

In the following chapters, we first describe the previous works in the field of automatic cognate discovery, the datasets used in the work, the experiments conducted along with the results and error analysis of the different models and the future work that we have planned motivated by our results.

Chapter 2

Previous Work

There have been many works on find cognate pairs in languages, but the word lists used so far have been very limited. The conventional approaches developed for the task of cognate identification are usually based on a combination of different string similarity measures between a pair of words as features for different classifiers like maximum-entropy, decision trees and SVMs [6][2]. These include orthographic and phonetic similarity. The objective can be finding pairs of cognates among two related languages, or finding groups of cognates among multiple languages.

Hauer and Kondrak [5] incorporate a number of diverse word similarity measures, that are manually identified, as input to a SVM classifier. These features can include number of shared bi-grams, edit distance, the length of strings and longest common subsequence. They also define binary language pair features that are believed to encode the degree of affinity between pairs of languages, ie. the likelihood of two languages sharing cognates or not. The authors employ these binary language-pair features that are used to weigh the language distance and assist the task of semantic clustering of cognates. After the classification of word pairs as cognates or non-cognates, they perform clustering over all lexical items from different languages and the same meaning. The clustering quality is evaluated against the gold standard cognacy judgments.

T. Rama [13] uses a string kernel based approach for automatic cognate identification. He defines a normalized vector for every word string, based on the various constituent subsequences of the word. The different dimensions dimensions of the vector correspond to the different subsequences of various lengths, that can be formed over the set of characters. The weights in the vector for each subsequence is weighted by the count and gaps in the subsequence inside that word. By defining a common subsequence vector between the word pairs and using that as input features to the linear classifier, he shows that subsequence based features outperform word similarity measures in this task.

List et al. [9] introduce the novel notion of partial cogacy between words which is defined using cognate clusters of the constituent morphemes rather than the entire words. They motivate that normal cognate identification models perform poorly on South-East Asian languages due to the presence of large number of compound words in these languages. They predict the partial cognacy judgements by defining sequence similarity networks over the constituent elements or morphemes of the word. These networks are weighted by employing the same string similarity features as used in [5] but over the morphemes. They further use algorithms for network partitioning to find clusters of cognate morphemes.

In a recent work, T. Rama [14] introduces a siamese convolutional network based approach for the cognate task, motivated by models used for image similarity tasks. Here each word is transcribed using phonetic characters and he manually defines vectors for the different phonetic alphabets. This leads to each word being treated as a 2D image comprising of a vector of phonetic character vectors. These 'images' can then be used in traditional siamese convolutional network models for image similarity for pairwise cognate identification.

Chapter 3

Dataset

The input data to the models for the task of cognate identification include dictionaries, multilingual word lists, and bitexts. But the word lists that have been used in all the works so far have been relatively small. This is because cognacy judgement is a laborious task and requires expert domain knowledge, therefore not many datasets exist.

		Concepts		
		ALL	AND	ANIMAL
Languages	English	All	And	Animal
	French	Tut	Et	Animal
	Marathi	Serve	Ani	Jenaver
	Hindi	Sara	Or	Janver

Table 3.1: Sample Word List from the Indo-European Dataset

Table 3.1 shows a small part of a word list which is the typical data used in this task. The rows in the table represent individual languages and the columns represent individual concepts or meanings. Each entry in the table contains a unique cognate class label which defines the groups of cognate words.

The freely available[1] Indo-European Dataset [3] by Dyen et al., is the most commonly used dataset for cognate identification. It provides 16,520 lexical items for 200 concepts and 84 language varieties. It provides a unique CCN (Cognate Class

[1]http://www.wordgumbo.com/ie/cmp/iedata.txt

Number) to each word. Since this dataset is a very old data, it is transcribed in a broad romanized phonetic alphabet represented by the 26 characters.

The IELex Database[2] is also an Indo-European lexical database which has been derive from the Dyen Dataset and other sources and is curated by Michael Dunn. It has over 34,000 lexical items from 163 languages of the Indo-European family and information of around 5,000 cognate sets. However, the transcription in IELex is not uniform. T. Rama in their work [14] cleaned a subset of the IELex database of any non-IPA-like transcriptions and converted the data to IPA (International Phonetic Alphabet), which we have used in our work.

We would also use the TDIL Hindi-Marathi sentence-aligned corpus[3] as the testing data for our final model. This dataset would provide a large part of the vocabulary from the both the languages to search for cognates. This dataset is a sentence aligned corpus and not word aligned. It is meant for the task of machine translation. It should be noted that cognate words are not simply translations of each other in the different languages, they are words which are known to have historically evolved from the same common word in an ancestral language. Hence, the sentence aligned corpus does not provide any gold label for the cognacy detection task, it only provides a rich source of testing data. To evaluate the performance of model on this corpus, we would sample a subset of our predictions and manually annotate them for cognate judgements.

[2]http://ielex.mpi.nl
[3]http://tdil.mit.gov.in

Chapter 4

Experiments

In our works, we have implemented the gap-weighted subsequence based model [13] for cognate identification by following the paper. We implemented the model in Python, using scikit-learn [12] open source library for the classification model.

4.1 Model

Let Σ be the set of characters over which the data is defined. For any string s defined over Σ, it can be decomposed into (s_1, s_2, ..., $s_{|s|}$) where $|s|$ is the length of the string. Let I be a sequence of indices $(i_1, i_2, .., i_{|u|})$ such that $1 \leq i_1 < ... < i_{|u|} \leq |s|$. Then the subsequence u is formed by using the sequence of indices I from the sting s. For such a string s over Σ, the subsequence vector $\Phi(s)$ is defined as follows,

$$\phi_u(s) = \Sigma_{\forall I, s[I]=u} \lambda^{l(I)} \tag{4.1}$$

$$l(I) = i_{|u|} - i_1 + 1 \tag{4.2}$$

$$\Phi(s) = \{\phi_u(s); \forall u \in \cup_{n=1}^{p} \Sigma^n\} \tag{4.3}$$

Here $\lambda \in (0, 1)$ is the weight tuning parameter for the model and p is the longest length of the subsequence to be considered. The λ parameter controls the penalty of the gaps in subsequence as it is present in the string. When λ is close to 0, the subsequence is restricted to a substring as the decay for a larger length is large. When λ is close to 1, the weight $\phi_u(s)$ counts the number of occurrences of the subsequence u in s. The subsequence vector $\Phi(s)$ for every word s is further normalised by dividing it with $||\Phi(s)||$.

The model also makes it easy to incorporate class based features. We map each character in Σ to the broader character classes of $\{C, V\}$ representing consonants and vowels. The set V includes $\{a, e, i, o, u, y\}$ and C include $\Sigma - V$. Hence we can map each string s to its CV-sequence s_{CV}. For example, a string like $s = ANIMAL$ is mapped to $s_{CV} = VCVCVC$ and $s = ALL$ to $s_{CV} = VCC$. The subsequence vector for any string is then combined as vector of $\Phi(s) + \Phi(s_{CV})$.

The combined subsequence vector for two words, (s_1, s_2) can be defined in two ways,

$$\Phi_1(s_1, s_2) = \{\phi_u(s_1) + \phi_u(s_2); \forall u \text{ present in } s_1 \text{ and } s_2\} \tag{4.4}$$

$$\Phi_2(s_1, s_2) = \{\phi_u(s_1) + \phi_u(s_2); \forall u \text{ present in } s_1 \text{ or } s_2\} \tag{4.5}$$

The difference between the two combined susequence vectors mentioned above is that the first one only considers only the subsequences that are common to both s_1 and s_2, whereas the second takes the sum of all the subsequences. It can be said that the first model is *Multiplicative* while the second is *Additive* (We shall use this naming of the models for future reference). Although the *Multiplicative* model vector should capture the correct information regarding the common features between the words, it can be too sparse at times when there are not a lot of common subsequences between the word (which does not not necessarily imply that the words are not cognates). Thus in general, the *Additive* model vector has more number of non-zero feature as compared to the *Multiplicative* model.

A Linear SVM classifier model is then trained using the combined subsequence vector $\Phi(s_1, s_2)$ from either the *Multiplicative* or the *Additive.* We have used the python sci-kit learn library to train the SVM classifier.

4.2 Testing Methods

The sample points for the classifier are created by picking pairs of words from the same concept in the word list. If the Cognate Class label of these words is the same then we assign them a positive cognate label, otherwise a negative label. Using this method, we are able to extract around 600K samples with 150K positive samples. The training of the model is performed in the following two different cross validation methods.

4.2.1 Simple Cross Validation

In this method, all the lexical items in the word list are divided into 5 fold cross validation sets. The training samples are picked by considering all word pairs formed from the training folds and the testing samples consist of all word pairs formed from the testing fold. We report the average 5 fold cross validated F-Score as the measure of performance of the model, for various values of the parameter λ while keeping the maximum length of the subsequence (p) fixed at 3.

4.2.2 Cross Concept Cross Validation

In this method, all the meanings/concepts in the word list are divided into 5 fold cross validation sets. The training samples are picked by considering only word pairs from the set of meanings in the training folds and the testing samples consist of all word pairs from the meanings belonging to the testing fold. The idea here is to test if the model learns general trends in sound change across the language which are applicable to words from meanings/concepts that the model has not observed during training.

4.3 Results

From the plot of the F-score with the variation in Lambda (Fig 4-1), it is clearly observed that the *Multiplicative* model, i.e. the vector comprising of only the common subsequences, performs better than the *Additive* model despite having sparser vectors

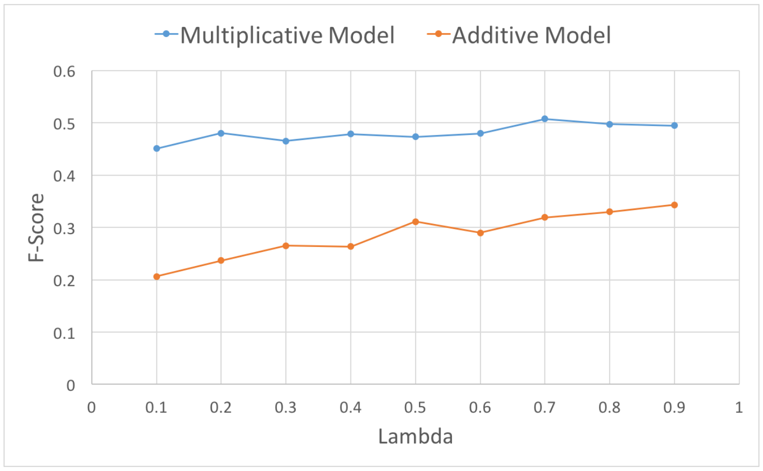

Figure 4-1: Average Cross validation F-Score variation with Lambda using Simple Cross Validation method

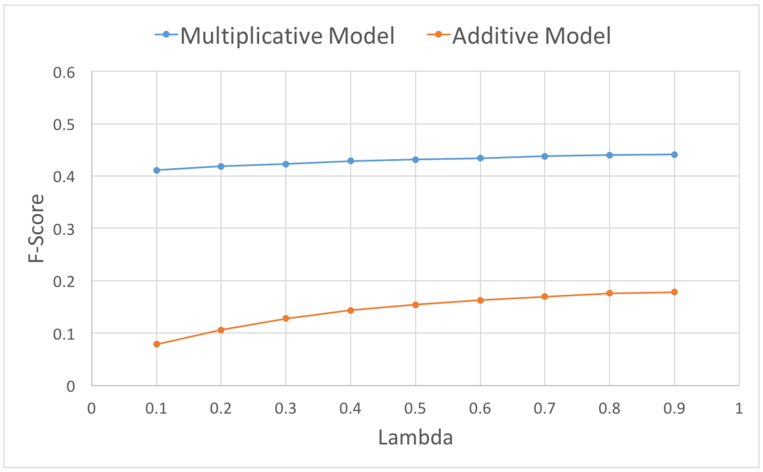

Figure 4-2: Average Cross validation F-Score variation with Lambda using Cross Concept Cross Validation method

and learning over a smaller feature space. The models learns better for values of lambda closer to 1. As lambda approaches 1, the weight of a subsequence in the vector corresponds to the count of the subsequence in the string, while lambda closer to 0 restricts the subsequence to a substring. We get a maxima in the F-score for lambda equal to 0.7. It is observed that generally low performance of the models is due to poor recall, when the precision stays high around 80-90%.

From the cross-concept cross-validation experiment, it is interesting to note that the training and testing samples were obtained from separate concepts altogether. Hence the model is learning general trends of sound change that have emerged over the languages which stay valid across concepts.

The *Multiplicative* models maintains its performance across both the cross validation methods, but it is observed that the *Additive* model performs much worse in the cross-concept setting. The *Additive* model overfits on the training data despite setting a high regularization penalty. This can be because when combining the word subsequence vectors in this format, the combined vector can get dominated by the word with the longer length as its subsequence count is higher. We perform this analysis in the following sections.

4.4 Error Analysis

From the results, its was apparent that the *Additive* model trained using $\Phi_2(s_1, s_2)$ performs poorly and overfits on the training despite tuning the regularisation penalty. In the following sections we report the analysis performed on these models.

4.4.1 Division of Meanings to Broad Categories

As the first step of analysis the performance of the *Multiplicative* model was observed over three different broad categories in which the samples were divided based on their POS tags. These categories were labeled into 'Noun', 'Adjective' and 'Others' using the Penn Treebank standard POS labels for the concepts. The following trends were observed over the three broad categories for the models tuned to their best parameters.

	Testing Data From		
Training Data From	Adjectives	Nouns	Others
Adjectives	0.513	0.330	0.160
Nouns	0.422	0.490	0.208
Others	0.350	0.380	0.360
All	0.5223	0.4947	0.351

Table 4.1: Average Cross Validated F-Scores over the different categories of test data

Here each row in the table comes from a different model, that is trained using training samples from the specified category. It is observed that there is an apparent division of performance of the models based on the three categories of samples. The model trained from samples belonging to ‘Others’ category performs poorly as compared to the remaining models. Also the model trained on all data performs poorly on test samples from the ‘Others’ category as compared to ‘Noun’ and ‘Adjectives’. Hence, we can say that there is a demarcation in the performance of the model based on the kind of data that it is being applied on. The cognate pairs present within a category like ‘Adjectives’ is defined or influenced by rules that are easily captured by the current model while those present with ‘Others’ are not. The evolution of cognate word pairs seems to be driven by the semantics of the word which would control its frequency of usage and change over time.

It is interesting to note that the model trained using all data performs better on the Adjectives class by a margin as compared to model trained using data only from the Adjectives class. This suggests there are also some general trends or rules that govern cognate evolution. There must be some cognate similarity information being shared across concepts that comes from the ‘Nouns’ or ‘Others’ data and stand helpful for samples in ‘Adjectives’.

4.4.2 Performance over individual meanings

To further investigate the performance, the results were divided over individual meanings from which the samples were derived in the word list.

It was observed that the results varied drastically over the different meanings. As mentioned earlier, the F-score was affected only due to the Recall of the samples when the Precision was mostly constant around 90%. The Recall varied from as high as 80% for some meanings like ‘CHILD’, ‘TOOTH’, ‘LAKE’ to as low as 5% for concepts like ‘WHEN’, ‘WHERE’, ‘WHAT’, as shown in Tables 4.2, 4.3.

Again we can observe the general trend that the model is learning better for concepts that belong to Nouns and Adjective classes as compared to the non-Nouns and non-Adjectives. By observing the data it was realised that the number of distinct cognate classes in the dataset from which the words are sampled is on average less for concepts that perform poorly for the model. Such concepts have large variations

Concept	**Precision**	**Recall**	**F-Score**	**Num Cognate Classes**
CHILD	99.98	79.99	0.888	24
TOOTH	99.99	76.92	0.869	5
BLACK	85.70	85.70	0.856	14
LAKE	81.81	89.99	0.856	22
EARTH	99.99	71.3	0.831	19

Table 4.2: Performance on individual Concepts : Best Results

Concept	**Precision**	**Recall**	**F-Score**	**Num Cognate Classes**
WHEN	99.98	7.59	0.141	8
HOW	79.98	7.69	0.140	8
WHERE	99.998	7.35	0.136	6
WHAT	999.95	5.49	0.103	5
IN	59.98	3.99	0.074	12

Table 4.3: Performance on individual Concepts : Worst Results

of sounds or transcription within a class of cognates. For example, Table 4.4 shows a small part of the word list for the concept 'WHAT', from the Indo-European dataset by Dyen et al.

Language	**Word**	**Cognate Class**
Takitaki	HOESAN	1
Singhalese	MOKADA	1
Hindi	KYA	2
Nepali	KE	2
Spanish	QUE	2
Slovak	CO	2
Swedish	VA	2
Danish	HVAD	2

Table 4.4: Part of Word list for concept 'WHAT'

Even within the same cognate class (class 2), there is a lot of variation between the words, so much so that the Danish *Hvad* and the Spanish *Que* do not actually share any subsequences in their normal form. Even when the strings are translated to the CV character string ($Hvad => CCVC$ and $Que => CVV$), they share only one common subsequence $\{CV\}$. Clearly the model cannot learn to predict cognates from such word pairs.

4.4.3 IPA versus Romanized IPA

Figure 4-3 shows the variation of the cross validated F-score with the parameter Lambda for the data from the two different word lists, i.e. the Indo-European dataset by Dyen et al. (henceforth referred to as Dyen Dataset) and the IELex dataset. The main difference between the Dyen Dataset and the IELex is in the transcription of the data. The cleaned IELex is transcribed in uniform IPA or International Phonetic Alphabet which is the standardized phonetic notation for representing sounds of a spoken language. The Dyen dataset is an older dataset which contains the words transcribed in a romanized version of the IPA. This romanized character is a more broader character that the IPA and it is only a set of 26 characters as opposed to 108 in IPA.

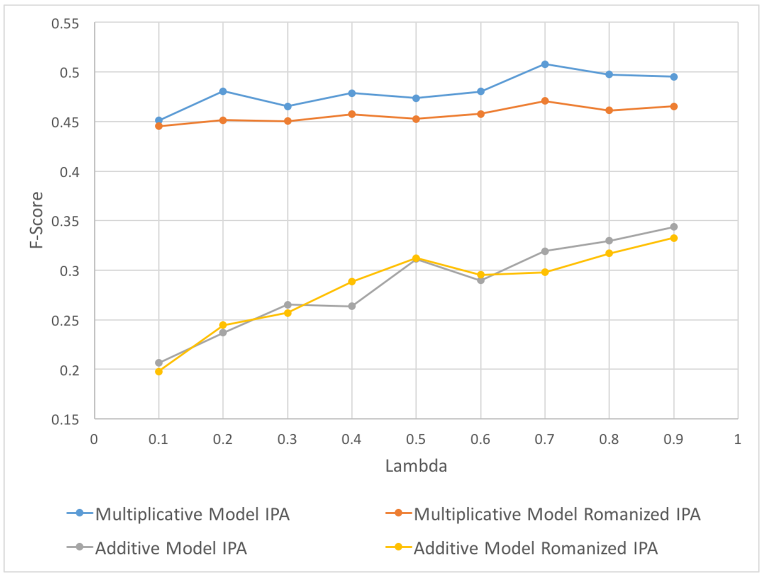

Figure 4-3: 5-Fold cross validation F-Score variation with Lambda for different transcription of data and different models

Since the IPA is a finer character, it represents sound change between the languages better and hence we get a better performance on the IELex dataset over the Dyen Dataset. Also since the character set is bigger, the space over which the samples are defined is of higher dimension. It can be seen that for both the *Multiplicative* and the *Additive* models, the performance over the IELex dataset (IPA) is generally better than the Dyen Dataset (Romanized IPA).

4.4.4 Analysis of Additive Model

From figures 4-1, 4-2, we had observed that the *Additive* model performs significantly poorly as compared to the *Multiplicative* model. The model seemed to overfit on the training data as is apparant from the following figures.

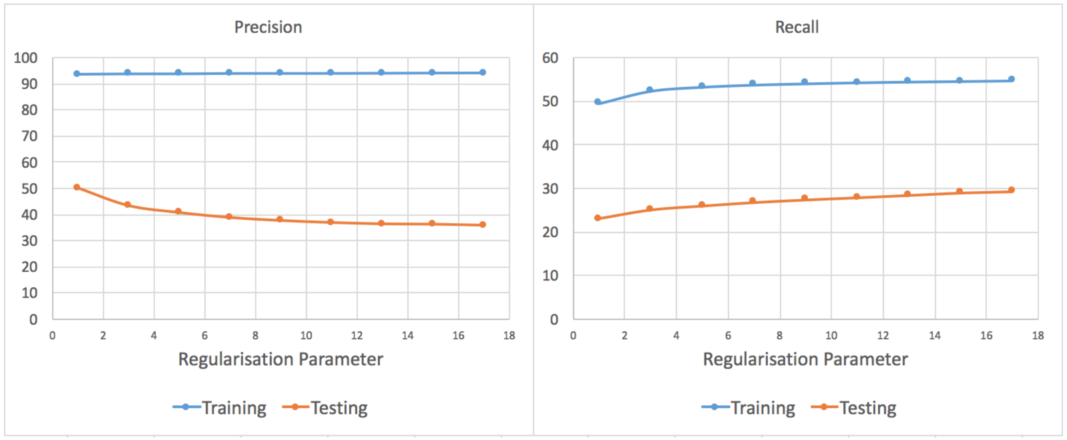

Figure 4-4: Precision & Recall curves with varying Regularisation Penalty

Fig 4-4 shows the variation of the Precision and Recall with varying regularisation for the SVM model. There is an unusually high and constant gap between the training and testing results for both Precision and Recall which does not seem to vary a lot with the regularisation penalty.

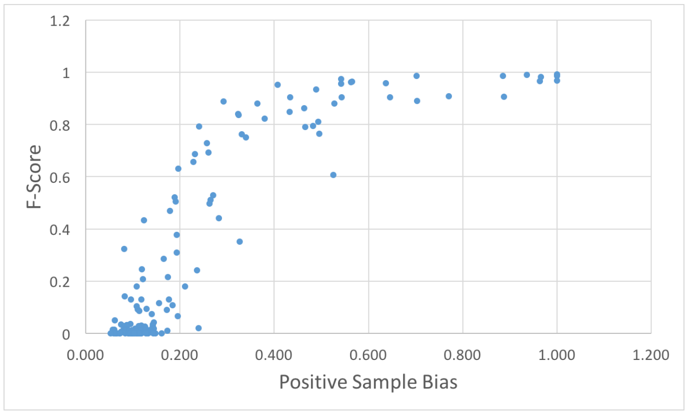

Figure 4-5: F-Score with positive sample bias for *Additive* model

Fig 4-5, shows the performance of the *Additive* model over different concepts with different Positive Sample Biases, i.e. the ratio of positive samples to total samples for the concept. The figure shows more clearly that the model overfits on the concepts with a high positive samples bias and does not learn to classify meanings where the positive sample bias is low.

The *Additive* model was created out of the motivation that the *Multiplicative* model returns very sparse vectors to the classifier and also cuts down on information regarding the subsequences that are not common between the words. However, the *Additive* model turns out to not be the ideal way to combine the individual subsequence vectors of the words for classification. A better way to create non-sparse vectors and maintain the information regarding non-common subsequences would be to create a hybrid between the *Additive* and the *Multiplicative* model, which we have talked about in our Future Work section.

Chapter 5

Conclusion

The analysis of the results from the subsequence models reveals that the performance of the system is not uniform across concepts. There are concepts where the cognate classes are segregated finely enough so that the model is able to learn cognates pairs with high precision and recall but others where the cognate classes are so broad that the model is not able to predict anything. This split in the data can come from the fact that for some concepts, the sound classes have evolved a lot, such that there is significant variation in the structure of the words, while in others this evolution has not been so drastic.

"WHAT"	
Language	**Word**
Hindi	KYA
Nepali	KE
Spanish	QUE
Swedish	VA
Danish	HVAD

"HOW"	
Language	**Word**
Swedish	HUR
Icelandic	HVERSU
Hindi	KESA
Russian	KAK
Spanish	COMO

"LAKE"	
Language	**Word**
Ukrainian	OZERO
Slovak	JAZERO
Serbocroatian	JEZERO
Polish	JEZIORO
Lusatian	JEZOR

"CHILD"	
Language	**Word**
Ukrainian	DYTYNA
Slovak	DIETNA
Serbocroatian	DETE
Czech	DITE
Lusatian	ZESE

We can see such examples of drastic and non-drastic changes in structure, in the tables shown that present few words from the same cognate class for the specified meanings.

Cognate formation results from the evolution of sound changes in the words over time. From our experiments we have seen that there is a link in this evolution of sound class with the semantics of the words. Because words with different meanings are used in different frequencies, some appear to go through rapid adaptation and while others do not change by a lot. Nouns and Adjective words are seen to have better performance and more number of cognate classes. In particular, words like *'WHAT'*, *'WHEN'*, *'HOW'* show a lot of variation even within a cognate class, so much that some cognate word pairs do not share any subsequence. Thus, the semantics of a word seem to be playing a significant role in the cognate prediction task and should be used along with the phonetic and orthographic features.

Chapter 6

Future Work

Motivated by the results of our analysis over the existing models, we have seen that semantics of the words influence the behavior and performance of the models. All the previous works on cognate identification mainly use phonetic or orthographic features of words for this judgment. The subsequence model particularly considers only such features. However it is evident that the semantics are also important features that should be considered while predicting cognates. Also, since the aim of this project is to use the model on a larger vocabulary of data from the aligned corpus, where the words are not grouped by meanings as in word lists, the model is expected to get confused without any semantic information. We propose to introduce this semantic information by utilizing the word embedding features in out model.

Word embeddings are representation features where the words in the vocabulary are represented as points in a low dimensional space as compared to the vocabulary size. These are learnt by using a unsupervised probabilistic modeling approach with deep learning models. They are task independent features that are arranged such that their structure captures some sort of semantic relationships between the words. It has been shown that word embedding vectors capture semantic information [4] and there have also been works to improve these semantics encoded in the embedding [10]. We propose to use the multilingual word embeddings Polyglot [1] that provide word vector embeddings for 116 languages over a rich vocabulary. These are trained on the processed Wikipedia text dumps of the various languages.

Along with using the word embedding features in our model, we would also like to move towards a neural network based model for classification of cognates. By utilizing recurrent networks like RNNs and LSTMs to encode the input words (character sequences) at the character level, we can use attention based models to classify the word pairs. Such a model will not be prone to problems like sparse vectors and loss of information in the case of the *Multiplicative* model.

As mentioned earlier in our analysis, we would also like to try and test a Hybrid model between the *Additive* and the *Multiplicative* models. Such a model can be implemented be considering the vector output by the *Multiplicative* model, and then stealing and redistributing the weights from all the high positive feature to the zero subsequence features that are present in the string. This concept is inspired from smoothing of sparse vectors. Some initial analysis of such a model has provided interesting results that improve the performance of the *Multiplicative* model, but we need to perform further analysis before reporting its results.

Once our models have been trained in the multilingual setting, we would like to apply it specifically to the domain of Hindi-Marathi using the sentence aligned corpus and evaluate the cognate pairs that we are able to discover.

Bibliography

[1] Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[2] Shane Bergsma and Grzegorz Kondrak. Alignment-based discriminative string similarity. In *Annual meeting-Association for Computational Linguistics*, volume 45, page 656, 2007.

[3] Isidore Dyen, Joseph B Kruskal, and Paul Black. An indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical society*, 82(5):iii–132, 1992.

[4] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

[5] Bradley Hauer and Grzegorz Kondrak. Clustering semantically equivalent words into cognate sets in multilingual lists. In *IJCNLP*, pages 865–873. Citeseer, 2011.

[6] Diana Inkpen, Oana Frunza, and Grzegorz Kondrak. Automatic identification of cognates and false friends in french and english. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 251–257, 2005.

[7] Grzegorz Kondrak and Bonnie Dorr. Identification of confusable drug names:

A new approach and evaluation methodology. In *Proceedings of the 20th international conference on Computational Linguistics*, page 952. Association for Computational Linguistics, 2004.

[8] Grzegorz Kondrak, Daniel Marcu, and Kevin Knight. Cognates can improve statistical translation models. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 46–48. Association for Computational Linguistics, 2003.

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

[10] Andrew L Maas and Andrew Y Ng. A probabilistic model for semantic word vectors. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[11] Gideon S Mann and David Yarowsky. Multipath translation lexicon induction via bridge languages. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] Taraka Rama. 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, May 31–June 5, 2015 Denver, Colorado, USA*, pages 1227–1231, 2015.

[14] Taraka Rama. Siamese convolutional networks based on phonetic features for cognate identification. *arXiv preprint arXiv:1605.05172*, 2016.

[15] Michel Simard, George F Foster, and Pierre Isabelle. Using cognates to align sentences in bilingual corpora. In *Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: distributed computing-Volume 2*, pages 1071–1082. IBM Press, 1993.

[16] Dan Tufis. A cheap and fast way to build useful translation lexicons. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.