Cognate Discovery to Bootstrap Lexical Resources

Shantanu Kumar, Dept. of Electrical Engineering, IIT Delhi
Supervisor - Prof. Sumeet Agarwal & Dr. Ashwini Vaidya

***Abstract*—The high level of linguistic diversity in South Asia poses the challenge of building lexical resources across these languages. This project is aimed at automatically discovering cognates between closely related language pairs (e.g. Hindi-Marathi or Hindi-Punjabi) in a scalable manner. We would like to analyze a large part of the vocabulary for both languages, as opposed to small word lists used in previous works. 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.**

I. Introduction

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. 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 '$*k^wek^welo$' 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, and cognate information has been successfully applied to NLP tasks, like sentence alignment in bitexts [1] and statistical machine translation [2]. It can also be used to bootstrap lexical resource creation for a language with low resources by finding parallels in related rich resource languages.

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.

II. Previous Work

The approaches developed for the task of cognate identification are usually based on combination of different similarity measures between a pair of words as features to a linear classifier. 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 [3] 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, and longest common subsequence. They also use features that encode the degree of affinity between pairs of languages. The authors employ binary language-pair feature that is 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 [4] use a string kernel based approach for automatic cognate identification. They define every word as a normalized vector, with different dimensions representing all the different subsequences of various lengths that are present in the word, weighted by the count and gaps of the subsequence inside that word. By identifying all common subsequences vector between the word pairs and using that as input features to the linear classifier, they show that subsequence based features outperform word similarity measures in this task.

III. Datasets

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 probably because the gold label for cognacy judgement is a debatable task and has been applied to small datasets only.

TABLE I. Sample Word List from Indo-European Dataset

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

Table 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 word in the table contains a unique cognate class label which defines the groups of cognate words.

The freely available Indo-European Dataset (Dyen et al., 1992) 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 Cognate Class Number to each word. The dataset is transcribed in a broad romanized phonetic alphabet.

The IELex Database is also an Indo-European lexical database which has been derive from the Dyen Dataset and other sources. It has over 34K lexical items from 163 languages of the Indo-European family and information of around 5000 cognate sets. However, the transcription in IELex is not uniform. T. Rama in their work [5] 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 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.

IV. Experiments

We have implemented the gap-weighted subsequence model [4] for cognate identification by following the paper. We implemented the model in Python, using scikit-learn open source library for the classification model.

A. Model

Let Σ be the set of characters over which the data is defined. For any string s defined over Σ, let I be a sequence of indices $(i_1, i_2, .., i_{|u|})$ and u be the subsequence of s corresponding to I. For such a string s over Σ, the subsequence vector $\Phi(s)$ is defined as follows,

$$\phi_u(s) = \Sigma_{I;s[I]=u} \lambda^{l(I)}$$

$$l(I) = i_{|u|} - i_1 + 1, \lambda \in (0, 1)$$

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

Here λ is the weight tuning parameter for the model and p is the longest length of the subsequence to be considered. The subsequence vector $\Phi(s)$ for every word s is also normalised by dividing it with $||\Phi(s)||$. The combined subsequence vector for two words, (s_1, s_2) can be defined in two ways,

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

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

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*. Although the first 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). Hence we also define the strings over a broader character set of $\{C, V\}$ representing consonents and vowels. The set V includes $\{a, e, i, o, u, y\}$ and C include $\Sigma - V$. Thus 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 the combined vector of $\Phi(s) + \Phi(s_{CV})$.

The Linear SVM classifier model is then trained using the combined subsequence vector $\Phi(s_1, s_2)$ for each sample pair of words. We have used the python sci-kit learn library to train the SVM classifier.

B. Testing Methods

The training of the model is performed in the following two different cross validation methods.

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

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.

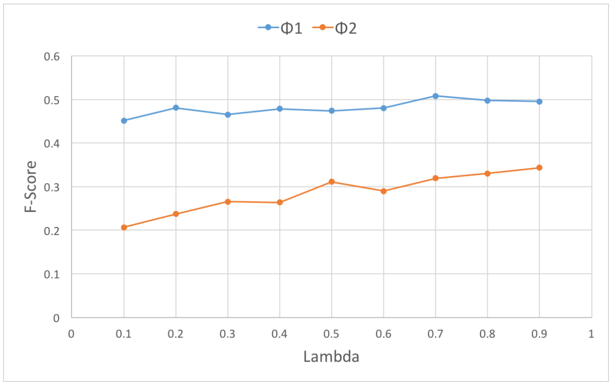

Fig. 1. 5-Fold cross validation F-Score variation with Lambda using simple cross validation method

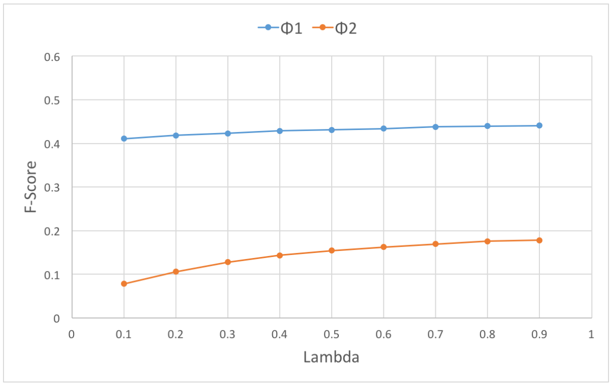

Fig. 2. 5-Fold cross validation F-Score variation with Lambda using cross concept cross validation

C. Results

From the plot of the F-score with the variation in Lambda, 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 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.

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

It is observed that the *Additive* model performs much worse in this setting as compared to the common subsequences only model. The second model overfits on the training data despite setting a high regularization penalty. This is probably 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.

D. 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. Thus, we use the common subsequence only model (*Multiplicative* model) and perform the error analysis on that.

1) Division of Meanings to Broad Categories: As the first step of analysis the performance of the two models were observed over three different broad categories over which the samples were divided. These categories were formed by dividing the meaning from which the sample was derived into 'Noun', 'Adjective' and 'Others'. The following trends were observed over the three broad categories for the models tuned to their best parameters.

TABLE II. 5-FOLD CROSS VALIDATED F-SCORES FOR THE DIFFERENT MODELS OVER THE DIFFERENT CATEGORIES OF TEST DATA

	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

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'. It is also observed 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. Hence there must be some cognate similarity information being shared across concepts that stand as universal for the languages.

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. 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'.

TABLE III. PERFORMANCE ON INDIVIDUAL CONCEPTS : BEST RESULTS

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 IV. PERFORMANCE ON INDIVIDUAL CONCEPTS : WORST 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

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. It is observed that such concepts have large variations of sounds or transcription within a class of cognates. For example, the following data is from the word list of concept 'WHAT'.

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

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 anything from such word pairs.

3) IPA versus Romanized IPA: Figure 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 as it only a set of 26 characters as opposed to 108 in IPA.

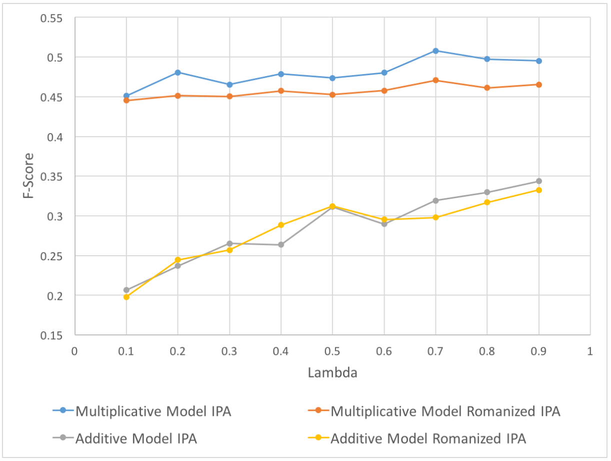

Fig. 3. 5-Fold cross validation F-Score variation with Lambda for different transcription of data and different models

Since the IPA is a finer character, it represents sound change between the languages better and hence we get a better performance over the IELex dataset as compared to 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 is generally better than the Dyen Dataset.

V. Conclusion

The analysis of the results from the subsequence model 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. We can link the evolution of sound class with the semantics of the words. 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.

VI. Future Work

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 as it has been observed in the analysis of these models, the semantics of the word play a role in the performance of the model. Also, since we aim to use the model on a larger portion of data, where the words are not aligned or grouped by meanings as in word lists, some sort of semantic information would be needed by the classifier. We propose to introduce this semantic information by utilizing the word embedding features.

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 unsupervised approached using deep learning models. They are task independent features that are arranged such that their structure captures some sort of semantic relationships between the words. We propose to use the multilingual word embeddings Polyglot [6] 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 deep learning based model for classification of cognates. By utilizing recurrent networks like RNNs and LSTMs to encode the input words (character sequences), we can use attention based models to classify the word pairs. Once the model is trained in the multilingual setting, we would like to apply it specifically to the domain of Hindi-Marathi and observe the cognate pairs that the model is able to identify.

References

[1] M. Simard, G. F. Foster, and P. 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*, pp. 1071–1082, IBM Press, 1993.

[2] G. Kondrak, D. Marcu, and K. 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*, pp. 46–48, Association for Computational Linguistics, 2003.

[3] B. Hauer and G. Kondrak, “Clustering semantically equivalent words into cognate sets in multilingual lists.,” in *IJCNLP*, pp. 865–873, Citeseer, 2011.

[4] T. 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*, pp. 1227–1231, 2015.

[5] T. Rama, “Siamese convolutional networks based on phonetic features for cognate identification,” *arXiv preprint arXiv:1605.05172*, 2016.

[6] R. Al-Rfou, B. Perozzi, and S. Skiena, “Polyglot: Distributed word representations for multilingual nlp,” in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, (Sofia, Bulgaria), pp. 183–192, Association for Computational Linguistics, August 2013.