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

28th April 2017

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Acknowledgments

I wish to express my sincere thanks to Dr. Sumeet Agarwal and Dr. Ashwini Vaidya for providing me with the opportunity to do this project. I am extremely grateful to them for their expertise and sincere and valuable guidance.

Shantanu Kumar

Contents

1 Introduction **6**

2 Related Work **8**

3 Dataset **10**

4 Surface Similarity Measures **13**

4.1 Model 13

4.2 Experiments 15

4.2.1 Results 15

4.2.2 Transcription Tests 17

4.2.3 Performance on individual concepts 18

5 Deep Model Representation **20**

5.1 Motivation 20

5.2 Model 20

5.2.1 Character Embeddings 21

5.2.2 LSTM network 22

5.2.3 Attention 23

5.2.4 Language Features 24

5.2.5 Concept Features 25

5.2.6 Cross Family Pre-training 25

5.3 Experiments 26

5.3.1 Evaluation Metric 26

5.3.2 Cross Language Evaluation 26
5.3.3 Cross Concept Evaluation 30
5.3.4 Analysis 31
5.3.5 Hindi-Marathi Extractions 32

6 Conclusion **35**

Abstract

The high level of linguistic diversity in South Asia poses the challenge of building lexical resources across the languages from these regions. This work is aimed at automatically discovering cognates between closely related language pairs like Hindi-Marathi. In this work, we present a deep learning model for this task of pairwise cognate prediction. We use a character level model with recurrent architecture and attention that has previously been employed on several NLP tasks. We compare the performance of our model with previous approaches on various language families. We also employ our model specifically to the domain of discovering cognates between Hindi and Marathi, which would assist the task of lexical resource creation. We analyze a large part of the vocabulary for both languages, as opposed to small word lists used in most works so far.

Chapter 1

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 [20]. 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.

Traditionally, the identification of cognates was carried out by historical linguists, using word lists and establishing sound correspondences between words. These are useful in determining linguistic distance within a language family, and also to understand the process of language change. While the task of cognate identification is an interesting problem in its own right, cognate information has also been used in several downstream NLP tasks, like sentence alignment in bitexts [23], improving statistical machine translation models [12], inducing translation lexicons [16][26] and identification of confusable drug names [11]. Additionally, it has been proposed that cognates can be used to share lexical resources among languages that are closely related [24].

For some time now, there has been a growing interest in automatic cognate identification techniques. Most approaches for this task focus on finding similarity measures between a pair of words (e.g. orthographic or phonetic similarity). [7] [9] [14]. These are used as features for classifiers to identify cognacy for a particular word-pair. Surface similarity

measures like these miss out on capturing generalizations beyond string similarity. For example, the cognate words *Que* in Spanish and *Hvad* in Danish both map to the concept WHAT, but have nothing in common with each other orthographically. For such pairs, surface similarity measures that rely on sub-sequence similarity are not useful. Instead, we require information about phonological similarity that is beyond surface similarity, such as the sound correspondences that are used in historical linguistics to narrow down candidate pairs as cognates.

Our work shows that an end-to-end recurrent neural network based model is able to outperform both surface similarity as well as a recent Convolutional Neural Network based Siamese-style model [21] on the task. LSTM (Long Short Term Memory) networks are being used in an extensive range of NLP tasks to build end-to-end systems. LSTMs have been successfully applied to machine translation [1]. language modeling [17], information retrieval [25] and RTE [3]. In the subsequent sections, we describe our LSTM based Siamese-style architecture which uses character by character attention to enrich the representations of the input word pairs and make the cognate prediction. We perform thorough analysis on the performance of our model and compare it against existing supervised approaches, including the subsequence model described in [20] and its variants.

The task of discovering cognates can possibly be particularly useful among the languages of South Asia, which are not rich in lexical resources. Information about cognates can become an important source for assisting the creation and sharing of lexical resources between languages. Therefore, another contribution of this work is to apply our cognate detection model to a real language pair. We apply our model to the domain of Hindi-Marathi, using a large unlabeled corpus of aligned texts to find cognate pairs.

In the following chapters, we first describe the previous works on this task followed by the datasets and language families used in our work. In chapter 4, we explore the models that exploit surface similarity between words highlighting its limitations and reasoning why a richer representation and alignment is needed for the task. Chapter 5 introduces our LSTM based end-to-end neural network model and its performance and comparisons with other models, including the results from the Hindi-Marathi domain test.

Chapter 2

Related Work

There have been many works on finding 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 [10][2]. These include orthographic and phonetic similarity as discussed earlier. The objective can be finding pairs of cognates among two related languages, or finding groups of cognates among multiple languages.

Orthographic features based classifier : Hauer and Kondrak [7] use a number of basic word similarity measures as input to a SVM classifier for cognate prediction. They use features like common bigrams, longest common substring, word length difference etc. They also define binary language pair features that help to encode the degree of affinity between pairs of languages, ie. the likelihood of two languages sharing 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.

Gap-weighted common subsequences : T. Rama [20] uses a string kernel based approach for automatic cognate identification. He defines a normalized vector for every word, based on the various constituent subsequences of the word. The different 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 as present inside that word. By defining a common subsequence vector between the word pairs and using that as input features to the linear classifier, it is shown that subsequence based features outperform word similarity measures.

Partial cognacy in morphemes : List et al. [13] 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 [7] but over the morphemes. They further use algorithms for network partitioning to find clusters of cognate morphemes.

Siamese ConvNet model : In a recent work, T. Rama introduces Convolutional Neural Network based siamese-style model [21] for the task. The model is inspired by image-similarity CNN models. Here each word is transcribed using phonetic characters and transformed using manually defined embedding vectors for the different phonetic characters. This leads to each word being treated as a 2D image comprising of a vector of phonetic character embeddings. By using a deep learning based model, the need for external feature engineering is avoided and the system outperforms previous works on the task.

In our work, instead of CNNs, we propose the use of LSTM networks [8]. As LSTMs are a form of recurrent network (RNN), they fit more naturally to natural language, which also has a sequential architecture and follows a linear order. In comparison, convolutional networks are hierarchical and hence seem natural for images. Even though NLP literature does not support a clear distinction between the domains where CNNs or RNNs perform better, recent works have shown that each provide complementary information for text classification tasks [30].

Chapter 3

Dataset

The task of cognate identification will make use of word lists that contain words from a number of languages of a language family. They contain words from a given set of concepts that are from the basic vocabulary such as kinship terms, body parts, numbers etc. Usually this vocabulary will represent concepts from the language itself and not borrowed items, although this is also possible at times. The word lists contain information about a particular word, its language family and a cognate class ID. These lists containing cognate class labels are usually small because cognacy judgement is a laborious task and requires expert domain knowledge and therefore not many datasets exist.

		Concept					
		ALL		*BIG*		*ANIMAL*	
Language	*ENGLISH*	all	001	big	009	animal	015
	FRENCH	tut	002	grand	010	animal	015
	MARATHI	serve	006	motha	011	jenaver	017
	HINDI	seb	006	bara	012	janver	017

Table 3.1: Sample Word List from the original Indo-European Dataset by Dyen et al.[5]

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 ID which defines the groups of cognate words.

We make use of three different datasets in our work described below.

IELex Database : The first and the primary dataset we work on is the IELex Database

Language Family	Languages	Concepts	Unique Lexical Items	Cognate Classes
Indo-European	52	208	8622	2528
Austronesian	100	210	10079	4863
Mayan	30	100	1629	858

Table 3.2: Statistics about the datasets

whic is an Indo-European dataset and which contains cognacy judgements from 208 Indo-European languages. The dataset is curated by Michael Dunn[1]. It was originally created by Dyen et al.[5] but then later expanded due to which its transcription is non-uniform. T. Rama extracted a subset of this dataset in their work [21] and cleaned it to uniform IPA (International Phonetic Alphabet) transcription, which we use in our work.

Austronesian Dataset : The second dataset is the Austronesian language family dataset that is taken from the **Austronesian** Basic Vocabulary project [6]. The dataset has been semi-automatically cleaned and transcribed to ASJP character set [21].

Mayan Dataset : The third dataset comes from the Mayan language family [27] that is spoken in Meso-America. This dataset is a small dataset and the word lists are uniformly transcribed in ASJP.

All the 3 datasets present are widely separated in time and geography provide a big domain to test out the systems. However, there are several differences in transcription in each of these datasets. While IELex is available in both IPA and a coarse 'Romanized' IPA encoding, the Mayan database is available in the ASJP format (similar to a Romanized IPA) [4]. IELex and the Austronesian dataset have also been semi-automatically converted to ASJP [21]. We use subsets of the original databases converted to ASJP due to lack of availability of uniform transcription. The statistics about the final sizes of the datasets used is mentioned in Table 3.2.

For the purposes of our task, we form word pairs using words from the same concept and such a pair is assigned a positive cognate label if their cognate class ids match. In this way, we can extract a total of 525,941 word pairs from Austronesian, 326,758 pairs from Indo-European and 63,028 pairs from Mayan, for training and testing the models.

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

Language Family	Unique Words	Words with >1 Meaning	Words belonging to >1 Languages	Fraction of subsequences found in data	
				Length 2	Length 3
Indo-European	8622	667 (∼7.7%)	1333 (∼15.4%)	0.82	0.43
Austronesian	10079	1119 (∼11.1%)	1682 (∼16.7%)	0.86	0.45
Mayan	1629	148 (∼9.1%)	384 (∼23.6%)	0.72	0.26

Table 3.3: Overlapping Statistics of the datasets

Table 3.3 contains some useful information about the overlaps within each dataset. We can see that since the Austronesian is a very big dataset with relatively few unique lexical items, it has a high overlap with respect to the number of words having multiple meanings with in the same language family. Also in terms of the number of words belonging to more than one language in the family, almost 24% of the words Mayan datasets belong to multiple languages. When using orthographic features like common subsequences between word pairs, it would be interesting to note how many possible subsequences actually appear in the dataset. It can be seen that for all possible subsequences of length 3 only around 45% of the subsequences for IELex and Austronesian and only 26% of the subsequences for Mayan actually appear in the training data. Thus, the test samples can have many common sequences that are not seen during training and are hence ignored.

We also use the TDIL Hindi-Marathi sentence-aligned corpus[2] for the domain test of extracting cognate words from Hindi and Marathi. This dataset consists of aligned-sentences from Hindi and Marathi originally constructed for a task like machine translation. The sentences are tokenized and POS tagged. They are transcribed in Devanagari and are automatically converted to IPA before testing on the model. They provide a large part of the vocabulary from the both the languages to search for cognates. 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 sample a subset of our predictions and manually judge them for cognates.

[2]http://tdil.mit.gov.in

Chapter 4

Surface Similarity Measures

In this chapter, we look at the gap-weighted subsequence model for cognate identification introduced by T.Rama [20] which tries to exploit surface similarity between words by using the common subsequences present in the words as features.

In our works, we have implemented and extended the gap-weighted subsequence based model and performed rigorous testing to identify its pitfalls and that of its extensions. We implemented the model in Python, using scikit-learn [18] 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 combined subsequence vector for two words, (s_1, s_2) can be defined in two ways,

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

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

We also define a third form of common subsequence vector for the word pair, which is a hybrid between the Φ_{Mul} and Φ_{Add} vectors, control by parameter $\alpha \in [0, 1]$.

$$\Phi_{Avg}(s_1, s_2) = (1 - \alpha) \cdot \Phi_{Mul}(s_1, s_2) + \alpha \cdot \Phi_{Add}(s_1, s_2) \tag{4.6}$$

The difference between the two combined subsequence 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). The *Average* model is then a hybrid between the *Multiplicative* and the *Additive* models. It tries to maintain a high overall weight for the common subsequence, but gives some amount of weight to the non-common subsequences as well depending on the value of α.

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 Experiments

We test the common subsequence model on the IELex datset transcribed in the IPA and the Romanized IPA characters as mentioned earlier. The testing is done in 2 forms of evaluation,

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.

Cross-Concept cross validation : In this method, all the meanings/concepts in the word list are divided into 5 fold cross validation sets. Thus the samples in the training and testing sets come from mutually exclusive sets of concepts.

4.2.1 Results

The performance of the subsequence models is reported in Figure 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 and learning over a smaller feature space. The models learns better for values of lambda closer to 1.

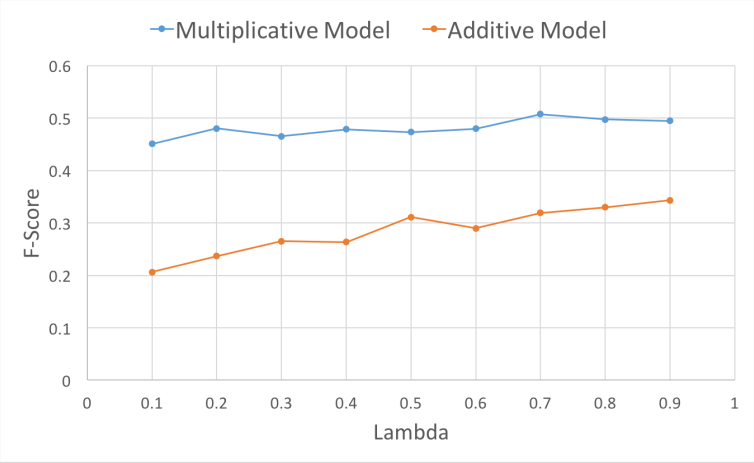

Figure 4-1: Average Cross validation F-Score (Simple Cross Validation)

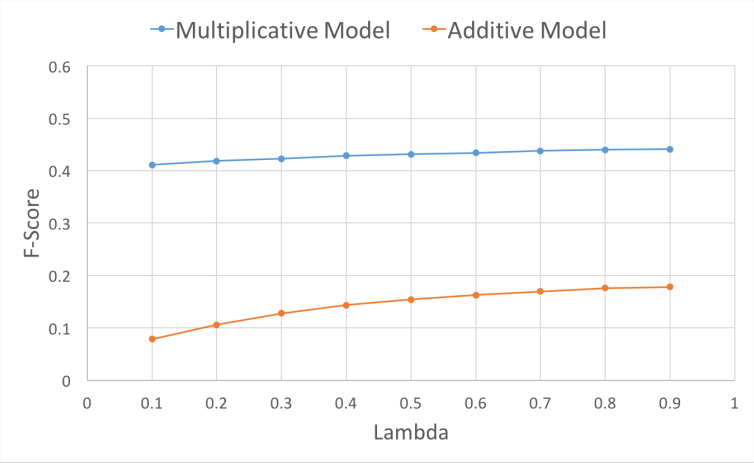

Figure 4-2: Average Cross validation F-Score (Cross Concept Cross Validation)

For the cross-concept cross-validation experiment, 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.

The performance of the *Average* models is noted in figures 4-3 and 4-4. It can be noted that as the model moves from the *Multiplicative* to the *Additive* model, the recall of the system gets a substantial increase around $\alpha = 0.4$ before falling again as α approaches 1. This causes the overall F-score of the model to increase at the cost of precision of th model.

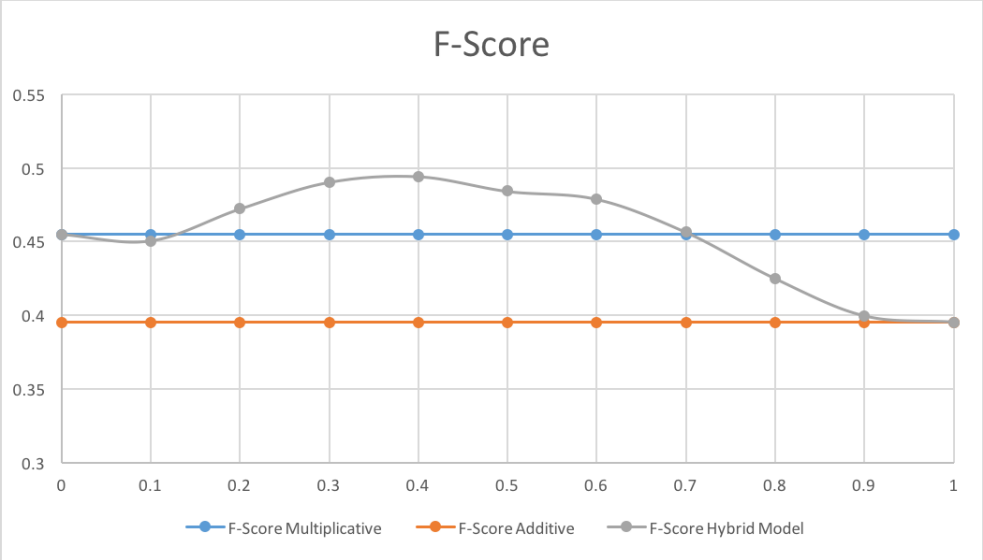

Figure 4-3: Average Model F-Score

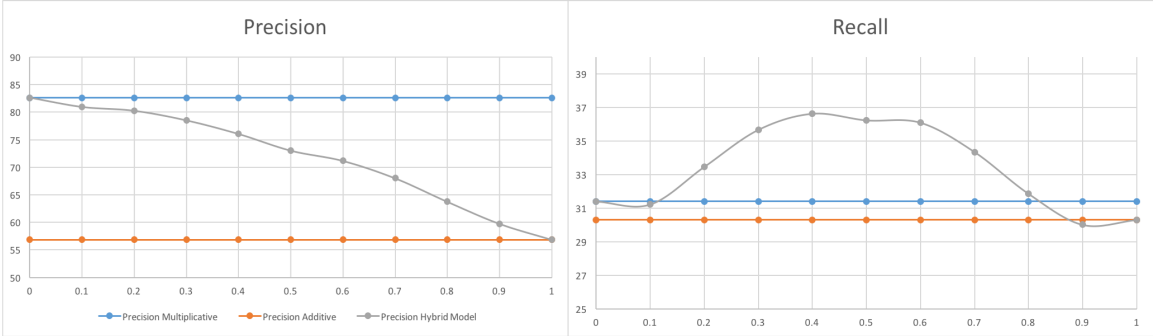

Figure 4-4: Average Model Precision and Recall

Even with the improved performance of the *Average* model, it is found that the model is still not good enough for real world application to new domains, with the best model only giving a recall of around 37% at a precision of 75%. It is clear that common subsequence models with just their surface similarity fundamentally lack the representation power to capture words the important features discovering cognates which goes beyond string matching.

4.2.2 Transcription Tests

We even evaluated the subsequence model on both forms of transcription of the IELex dataset, that is the IPA and the Romanized IPA forms. The IPA character set is a finer character set that consists of around 108 phonetic characters. The Romaized IPA is a broader notation that combines several characters from IPA into a single character, thus having a smaller character vocabulary of around 26 characters.

As can be seen in Figure 4-5, the performance of the subsequence models is similar on either form of transcription of the data. Even though the IPA provides a richer detail of structure in the word, the subsequence model is not able to exploit it, as a finer representation actually means less number of common subsequences.

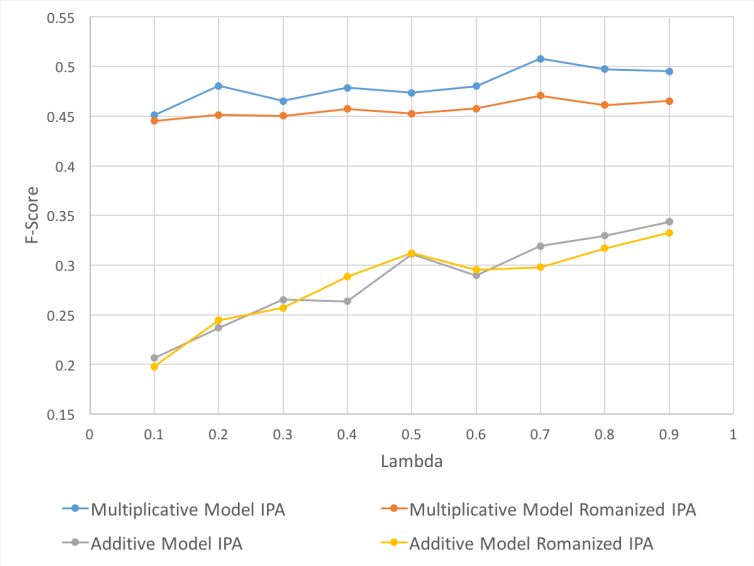

Figure 4-5: Cross validation F-Score for different transcription of data

4.2.3 Performance on individual concepts

To investigate the performance of the models, 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. It is observed that 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.

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 4.1: High Performance Concepts

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

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 4.2: Poor Performance Concepts

dataset from which the words are sampled is on average less for concepts that perform poorly for the model. Such concepts have large variations 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 dataset.

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 4.3: 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. Clearly a model looking at only the common subsequences cannot learn to predict cognates for such word pairs. Thus as motivated before, we need to work towards a model that is able to exploit deeper representations and recognize and learn sound correspondences and sound change between characters.

Chapter 5

Deep Model Representation

5.1 Motivation

As we observed with the common subsequence model, the models using these simple manually engineered string matching features are limited in their representation power and the ability to use or learn sound correspondences. We need a deeper level of representation that is learnt automatically without explicit feature engineering. Deep learning models avoid the need of manual feature engineering and provide a method for learning this representation. T. Rama [21] exploited the representation power of convolutional neural networks that learn n-gram level features over the words to define a feature space. Inspired from this idea we employ a recurrent neural network architecture for the task.

5.2 Model

The overall model used in our system is called the Recurrent Co-Attention Model (*CoAtt* for short) and is extended from the word-by-word attention model used by Rocktaschel et al.[22] for the task of recognising textual entailment in natural language sentences. The network used is illustrated in Figure 5-4. It is a Siamese style network that encodes a word pair parallely and then makes a discriminative judgement in the final layer. The input words are first encoded into character level embeddings followed by a bidirectional LSTM network and finally a character by character attention layer as described in the subsections

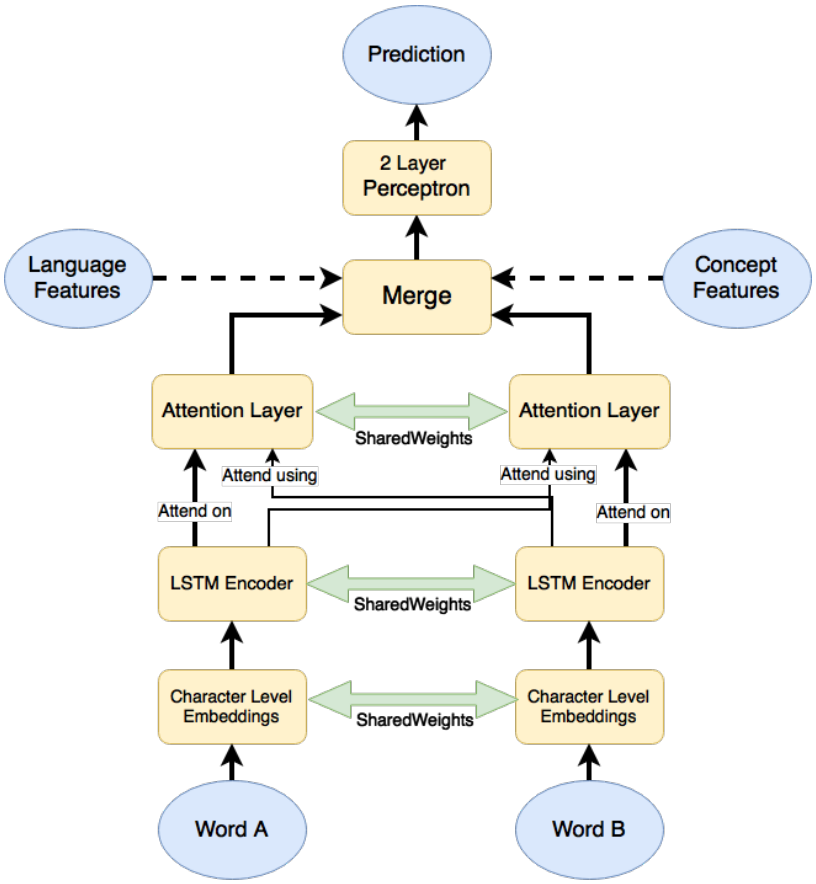

Figure 5-1: Recurrent Co-Attention Network for Cognate Discovery

that follow. The encodings of both the words are merged and passed through a 2-layer neural network with *tanh* and *sigmoid* activations to make a final binary prediction. Additionally we also add a *Language features* vector or a *Concept features* vector to the model by concatenating it with the merged attention vector before passing it to the 2-layer neural network.

5.2.1 Character Embeddings

The network starts by encoding the input words into their character embeddings. We encode the words using a character level embedding matrix $E \in \mathbb{R}^{n_e \times |C|}$. Here n_e is the size of the character embeddings and C is the vocabulary of all characters. Thus for an input word x which can be represented as sequence of characters $x = \{c_{i_1}, c_{i_2}, ..., c_{i_n}\}$, is transformed into a sequence of vectors $y = \{e_{i_1}, e_{i_2}, ..., e_{i_n}\}$ where e_j is the j^{th} column of the

E matrix. This embedding matrix is learnt during training and each column in the matrix represents the embedding vector of the respective token in the vocabulary.

T.Rama [21] manually defined the character embeddings using various properties of the respective phoneme and fixed these embeddings during training. However, we observe that such a method restricts the power of the distributional representation to world knowledge known to us and letting the embeddings be learnt themselves should help learn a representation that is useful for the task in hand.

5.2.2 LSTM network

Recurrent Neural networks (RNN) with Long Short-Term Memory (LSTM) have been extensively been used in several NLP tasks. After the input words to the network are encoded using the character embedding matrix, we transform them use LSTM cells. Given the input words $y = \{e_1, e_2, ..., e_n\}$, at every time step t the LSTM of hidden unit size n_h uses the next input e_t, the previous output h_{t-1} and the previous cell state c_{t-1} to compute the next output h_t and the next cell state c_t as follows,

$$H = [e_t h_{t-1}] \tag{5.1}$$

$$i_t = \sigma(W^i H + b^i) \tag{5.2}$$

$$o_t = \sigma(W^o H + b^o) \tag{5.3}$$

$$f_t = \sigma(W^f H + b^f) \tag{5.4}$$

$$c_t = i_t * tanh(W^c H + b^c) + f_t * c_{t-1} \tag{5.5}$$

$$h_t = o_t * tanh(c_t) \tag{5.6}$$

Here W^i, W^o, W^f, $W^c \in \mathbb{R}^{n_e + n_h \times n_h}$ and b_i, b_o, b_f, $b_c \in \mathbb{R}^{n_h}$ are trained weights of the LSTM and [] is the *concatenate* operator. The final output of the LSTM gives us a sequence $\{h_1, h_2, ..., h_n\}$ for every word, where $h_j \in \mathbb{R}^{n_h}$.

The LSTM network essentially helps to give a context aware encoding of the input word. Every output encoding h_j represents an encoding of the e_j character that is aware

of its history $\{e_1, e_2, ..., e_{j-1}\}$. It is also popular to use bi-directional LSTMs where the reverse sequence is also fed into the network and then the final outputs are concatenated. Using this mechanism, every output encoding h_j becomes aware of both its past context $\{e_1, e_2, ..., e_{j-1}\}$ and its future context $\{e_{j+1}, e_{j+2}, ..., e_n\}$. We use bi-directional LSTMs in our implementation.

5.2.3 Attention

Attention neural networks have been used extensively in tasks like machine translation [15], image captioning [28] and visual question answering [29]. The attention mechanism helps to enhance the representation obtained from the LSTM cell state by giving it context that is used for attending. More precisely, we attend over the LSTM encoding of a given word, using a single character encoding of the second word. This helps to generate a weighted representation of the first word, to include its important segments with respect to the second word's character.

Given a character vector $h \in \mathbb{R}^{n_h}$ using which we would like to attend on a sequence of character vectors $Y = \{c_1, c_2, ..., c_L\} \in \mathbb{R}^{n_h \times L}$, we generate a set of attention weights α and a attention-weight representation $r \in \mathbb{R}^{n_h}$ of Y as,

$$M = tanh(W^y Y + W^h h * e_L) \tag{5.7}$$

$$\alpha = softmax(w^T M) \tag{5.8}$$

$$r = Y\alpha_t^T \tag{5.9}$$

Using the mechanism followed by Rocktaschel et al.[22] for word-by-word attention, we employ a character-by-character attention model, wherein we find an attention weighted representation of the first word $Y = \{c_1, c_2, ..., c_L\} \in \mathbb{R}^{n_h \times L}$ at every character of the second word $H = \{h_1, h_2, ..., h_N\} \in \mathbb{R}^{n_h \times N}$.

$$M_t = tanh(W^y Y + (W^h h_t + W^r r_{t-1}) * e_L) \tag{5.10}$$

$$\alpha_t = softmax(w^T M_t) \tag{5.11}$$

$$r_t = Y\alpha_t^T + tanh(W^t r_{t-1}) \tag{5.12}$$

Here W^y, W^h, W^r, $W^t \in \mathbb{R}^{n_h \times n_h}$ and $w \in \mathbb{R}^{n_h}$ are trained weights of the Attention layer. The final output gives us $r_N = r_{YH}$ which can considered as attention weighted representation of Y with respect to H. Similarly, we also obtain r_{HY}. The final feature vector r^* that is passed to the multi-layer perceptron for classification is the concatenation of these 2 vectors.

$$r^* = [r_{HY} r_{YH}] \tag{5.13}$$

This method of making both the sequences attend over each is called the *Co-Attention* model. At the end of this process of attention, we end up with an embedding representation of each word, that is aware of its own context as well as the context of the other word in the pair.

5.2.4 Language Features

It is known that some languages are more closely related to each other as compared to other languages. Thus, these languages which are closer would naturally tend to share more cognate pairs than they do with other languages. T. Rama[21] tried to exploit this information about language *relatedness* by providing the network with 2-hot encoding vector that represents the respective languages of the 2 input words being tested. The network would then use this information about the languages to learn which language pairs may be more related using the training data provided.

We follow the same approach and provide the model with these addition *Language Features* before the final classification by the 2-layer MLP. The 2-hot input language pair

vector x_{lang} is concatenated with the attention weighted representation of the input words h^*, before being fed into the final 2-layer perceptron for classification.

5.2.5 Concept Features

As the information about the language of the input words can be beneficial for the task of cognate discovery, we hypothesise that information regarding the semantics or the meaning of the input word pair should also be helpful. The word semantics can provide information like the POS category of the word, which can be an useful if some POS classes show higher degree of variation in cognates while others show less.

We implement this by using GloVe word embeddings [19]. Word embeddings are distributional representation of words in a low-dimensional space compared to the vocabulary size and they have been shown to capture semantic information about the words inherently. We use the GloVe embedding for the English concept of the word pair as obtained from the label in the dataset, and input this vector to the network before the final MLP for classification. The word embedding of the concept $x_{concept}$ is concatenated with the attention weighted representation of the input words h^*, before being fed into the final multi-layer perceptron for classification.

5.2.6 Cross Family Pre-training

The three different language families with which we work have completely different origins and are placed across different regions geographically. However, it would be interesting to note if any notion of language evolution is still shared amongst these independently evolved language families. We try to test this hypothesis through the joint learning of the model for these families. To implement this, we instantiate the network with the combined character vocabulary of the two language families. We then train the model till the loss saturates on one language family. This is followed by the training of the model on the second language family, while using the learned weights from the pre-training as the initialization.

5.3 Experiments

We conducted several types of evaluations on our models covering a variety of test for thorough analysis of its performance. In the subsections below we describe the results from these different tests. The wordlist datasets used in our evaluation can be divided on the basis of languages or concepts for testing and training. We also conducted tests on how cross-family pre-training helps different models and also how the different levels of transcription affects the learning and performance of the models.

5.3.1 Evaluation Metric

We report the *F-score* and the area under the Precision-Recall curve (*AUC*) as a measure of performance for all our models. *F-score* is computed as the harmonic mean of the *precision* and *recall*[1], when the classifier confidence threshold is 0.5. Since the dataset is heavily biased and contains a majority of negative samples (As can be seen in Tables 5.1 and 5.4), *accuracy* is not a good measure of performance to compare the models.

5.3.2 Cross Language Evaluation

In the cross language evaluation test, we fixed a random set of 70% of the languages as the training set of languages and the remaining as the testing set, Then we took words from languages in the training set of languages and all concepts, to form the training samples and similarly for the testing samples. It must be noted that all samples were created by taking words from the same concept .ie. for each word pair in the training and testing set, be it positive or negative, belongs to the same concept.

The training and testing set size details for the different datasets formed using cross language evaluation test can be found in Table 5.1. The results for the cross language evaluation tests are listed in Table 5.2.

It is observed that the Recurrent Co-attention model (denoted as *CoAtt*) performs significantly better than the CNN [21] and the Subsequence [20] models for the Indo-European and the Austronesian datasets. For the Mayan dataset, the *CoAtt* model does not learn very

[1]Precision and Recall is computed on positive labels. Precision = TP/(TP+FP), Recall = TP/(TP+FN)

	Indo-European		Austronesian		Mayan	
	Total	Positive	Total	Positive	Total	Positive
Training Samples	218,429	56,678	333,626	96,356	25,473	9,614
Testing Samples	9,894	2,188	20,799	5,296	1,458	441

Table 5.1: Data size for Cross Language Evaluation

Model	Indo-European		Austronesian		Mayan	
	F-Score	*AUC*	*F-Score*	*AUC*	*F-Score*	*AUC*
Gap-weighted Subsequence	59.0	75.5	58.8	68.9	71.8	81.8
PhoneticCNN	73.7	86.1	54.6	68.0	72.8	85.0
PhoneticCNN + Language Features	62.2	85.4	46.8	67.0	66.4	84.0
CharacterCNN	75.3	85.3	62.2	71.6	75.9	85.7
CharacterCNN + Language Features	70.7	82.6	61.4	70.1	61.1	82.2
CoAtt	83.8	89.2	69.0	77.5	67.1	67.7
CoAtt + Concept Features	**83.5**	90.5	**68.9**	**77.9**	76.2	84.2
CoAtt + Pre-training (Austro)	83.2	**90.6**	-	-	**80.4**	**88.3**
CoAtt + Pre-training (IELex)	-	-	-	-	79.6	85.2

Table 5.2: Cross Language Evaluation Results

well and in fact performs worse than even the subsequence model. This can be due to the small size of the Mayan dataset, which is not sufficient for training the *CoAtt* network.

Since the mode of evaluation is cross-language, it is intuitive that the additional *Language Features* will not contribute to the performance of the model since the languages of the training and testing set do not overlap and hence the relevant language feature weights for the test set are never learnt. That is, any information about the affinity or interaction between the languages in the training set is not useful for the languages in the testing set, as they are not common. This can clearly be observed in the performance of the CNN models, whos performance deteriorates by a margin with the addition of the *Language features*. Hence, we do not use *Language features* with our model for cross-language evaluation.

On the other hand, the *Concept features* discussed earlier, are found to be useful in improving the performance of the *CoAtt* especially on the Mayan dataset. Using the extra information about the word meaning from the of input pair from the *Concept features*, the *CoAtt* model is able to cross the baseline performance on the Mayan dataset.

The best boost however for the *CoAtt* model on the Mayan dataset comes from pre-training the model on the Austronesian and the Indo-European datasets. Pre-training the

model on the other language families gives a good initialisation point to start training for the Mayan dataset. Using this method, the *CoAtt* model is thus able to beat the performance of best CNN model (*CharCNN*) on the Mayan dataset. This also provides evidence to our hypothesis that the *CoAtt* was not able to learn on the Mayan dataset simply because of the lack of enough data to train the network, but pre-training the model on other language families helped to show the true potential of the model on the dataset.

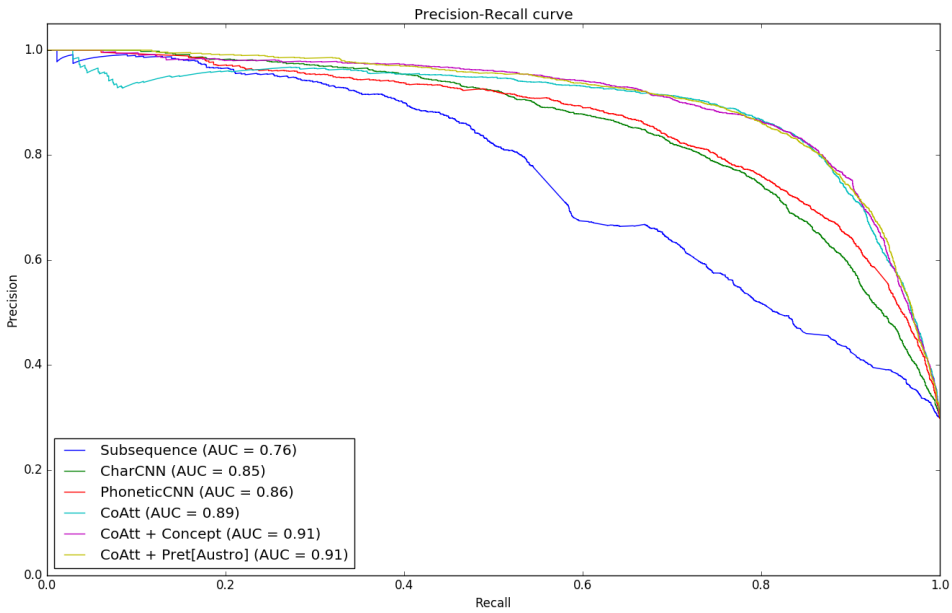

Figure 5-2: PR Curve for Cross Language Evaluation on Indo-European dataset

The *CoAtt* model was also tested with the 2 different transcriptions for the IELex dataset, IPA and ASJP. The results for the same are list in Table 5.3. It is observed that the model has almost similar performance in both the transcriptions, however the further analysis (shown later) reveals the different performance over different samples.

Model	**Indo-European (ASJP)**		**Indo-European (IPA)**	
	F-Score	*AUC*	*F-Score*	*AUC*
CoAtt	83.8	89.2	82.2	89.1
CoAtt + Concept Features	83.5	90.5	82.1	90.7

Table 5.3: Cross Language Evaluation : Transcription Tests

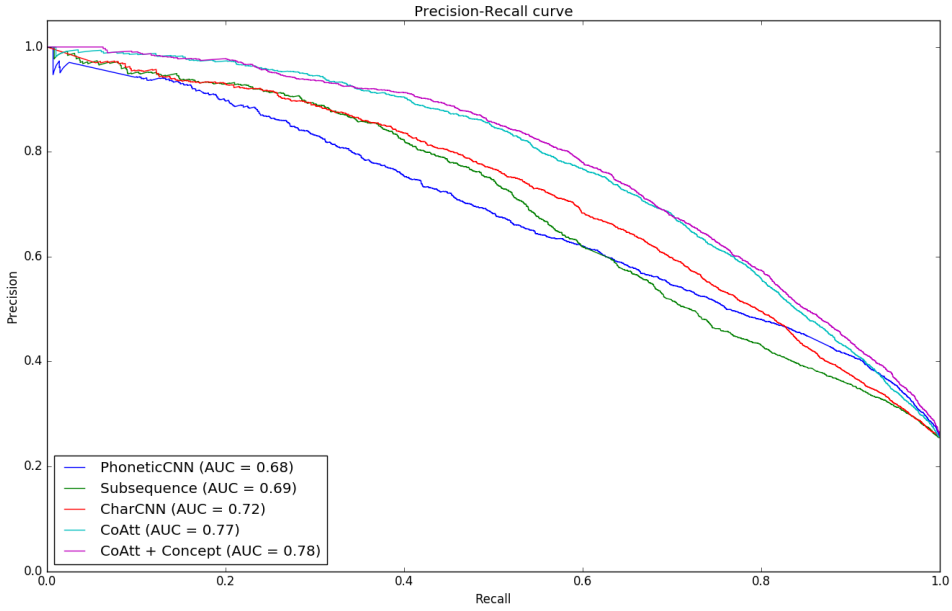

Figure 5-3: PR Curve for Cross Language Evaluation on Austro dataset

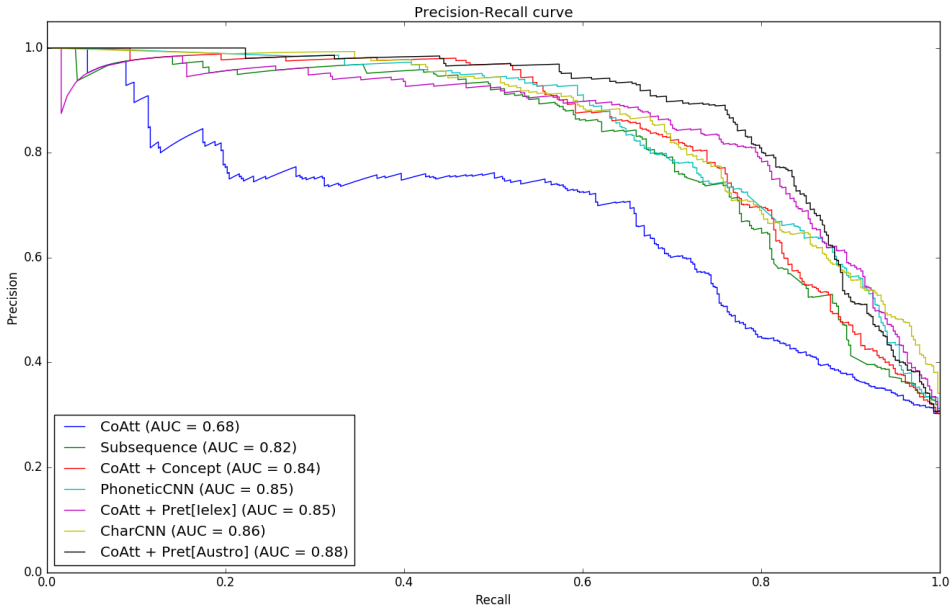

Figure 5-4: PR Curve for Cross Language Evaluation on Mayan dataset

	Indo-European		Austronesian		Mayan	
	Total	Positive	Total	Positive	Total	Positive
Training Samples	223,666	61,856	375,693	126,081	28,222	10.482
Testing Samples	103,092	21,547	150,248	41,595	12,344	4,297

Table 5.4: Data size for Cross Concept Evaluation

Model	Indo-European		Austronesian		Mayan	
	F-Score	*AUC*	*F-Score*	*AUC*	*F-Score*	*AUC*
Gap-weighted Subsequence	51.6	62.0	53.1	64.5	61.0	75.4
PhoneticCNN + Language Features	**66.4**	**73.2**	57.8	66.6	**80.6**	88.1
CharacterCNN + Language Features	63.5	70.5	**60.9**	**70.2**	79.6	**89.1**
CoAtt	64.8	69.8	57.1	61.0	70.5	74.8
CoAtt + Language Features	65.6	70.8	57.3	62.0	69.6	71.9
CoAtt+ Concept Features	64.1	70.6	58.0	63.1	71.9	78.6
CoAtt + Pre-training (Austro)	65.8	71.0	**-**	**-**	71.1	78.4
CoAtt + Pre-training (IELex)	-	**-**	-	-	71.2	79.0

Table 5.5: Cross Concept Evaluation Results

5.3.3 Cross Concept Evaluation

The cross concept evaluation, was conducted similar to [21], wherein the first 70% of the concepts were taken as training concepts and the remaining concepts as testing. The training and testing pair samples were then created from each set by using words from the same concept. The training and testing set size details formed using cross concept evaluation test can be found in Table 5.4. The results for the cross concept evaluation tests are listed in Table 5.5.

It is observed that the *CoAtt* model gives an almost equal performance for the Indo-European and Austronesian datasets as compared to the CNN models. For Mayan, the *CoAtt* model does not learn very well due to the small size of the data. Even though it is able to beat the Subsequence model in terms of F-score, it is still behind the CNN models by more than 10 points. The cross-concept evaluation test is a more rigorous test for cognate detection as the models have not seen any of the similar word structures during training. Words coming from different concepts would have different sequence structures altogether. Different concepts come from different parts of speech and may contain different levels of variability which is tough for the model to detect if it has not observed it during training.

5.3.4 Analysis

For the concept wise performance of the *CoAtt* models, it is observed that the performance is more uniform throughout the concepts as compared to more varied distribution of the subsequence model. For concepts like WHAT, WHO, WHERE, HOW, THERE where the subsequence model performed poorly, the *CoAtt* model is able to acheive high scores. The *CoAtt* model performs poorly on a few selected concepts like AT, IF, IN, BECAUSE, GIVE. By looking at the samples, it is found that these concepts are heavily biased by negative samples and contain only a handful of positive cognate pair examples. In fact the subsequence model could not perform at all on these concepts as the highly biased data is coupled with almost no overlap of subsequences.

Concept	**CoAtt**	**Subseq**	**Concept**	**CoAtt**	**Subseq**
WHAT	*0.91*	*0.04*	BECAUSE	*0.28*	*0*
WHO	*0.85*	*0.05*	IN	*0.35*	*0*
WHERE	*0.90*	*0.16*	IF	*0.31*	*0*
HOW	*0.90*	*0.17*	AT	*0.25*	*0*
THERE	*0.95*	*0.19*	GIVE	*0.45*	*0.35*

Table 5.6: *CoAtt* vs *Subseq* model on various concepts (F-Score)

To compare the IPA model with the ASJP models on the IELex dataset, it is found that the IPA model performs poorly on concepts like GUTS, SWIM, WIPE, WHITE, SING where the ASJP model gives good results. On the other hand, the IPA model performs better on concepts like FIRE, SLEEP, PULL, SAY and SMOOTH. By looking at specific examples, we find that for concepts like FIRE and PULL, the ASJP model gives many false positives which can be a fault due to the coarser representation of the ASJP character.

Concept	**IPA**	**ASJP**	**Concept**	**IPA**	**ASJP**
GUTS	*0.28*	*0.58*	FIRE	*0.62*	*0.33*
SWIM	*0.47*	0.93	SLEEP	*0.73*	0.39
WHITE	*0.54*	*0.75*	PULL	*0.83*	*0.50*
WIPE	*0.55*	*0.72*	SMOOTH	*0.83*	*0.50*
SING	*0.56*	*0.88*	SAY	*0.90*	*0.51*

Table 5.7: *CoAtt* model using IPA vs ASJP transcription on various concepts (F-Score)

ASJP		IPA	
Word 1	Word 2	Word 1	Word 2
swim	sinda	swɪm	ˈsɪn̪da
swim	zwem3n	swɪm	zwɛmən
swim	svim3n	swɪm	ʃvɪmən
swem3	sinda	ˈsʊθmːə	ˈsɪn̪da
swem3	zwem3n	ˈsʊθmːə	zwɛmən
swem3	svim3n	ˈsʊθmːə	ʃvɪmən
sinda	zwem3n	ˈsɪn̪da	zwɛmən
sinda	svim3n	ˈsɪn̪da	ʃvɪmən
zwem3n	sima	zwɛmən	ˈsimːa
sima	svim3n	ˈsimːa	ʃvɪmən

Figure 5-5: Sample test word pairs from the concept SWIM

Figure 5-5 shows few test sample pairs from the concept SWIM. It is observed that for all of these samples, the IPA model falsely predicts negatives whereas the ASJP model is correctly able to predict them all as cognates. Here perhaps the very find representation in IPA throws the model's judgment off and its not able to pick up the correspondence correctly. It is also interesting to note that for all of these samples, adding the *concept feature* of SWIM to the *CoAtt + Concept Features* model makes it predict all of them as cognates. Thus, perhaps adding the concept features signals the model to relax the degree of overlapping of phonemes and become less strict in predicting cognates.

5.3.5 Hindi-Marathi Extractions

Finally we applied the *CoAtt* model to the domain of Hindi-Marathi. The model was trained on the IELex dataset with IPA transcription with a character vocabulary of around 150 phonetic characters. The model was trained in a cross-language evaluation style. It should be noted that the IELex database contains instances from Marathi, but it does not directly contain instances from Hindi. However, it does contain words from Urdu and Bhojpuri (Bihari) which are also languages closely related to Hindi and share many words of the vocabulary with Hindi.

We used the TDIL sentence-aligned corpus. The corpus contains sentences from Hindi-Marathi that are POS tagged and transcribed in Devanagari. We specifically extracted word pairs from each sentence with the NOUN and VERB tags. Since the sentences are not

word aligned, we extracted candidate word pairs for testing by choosing the first word with the same tag in either sentence as the candidate pair. The words were converted from Devanagari to IPA using a rule-based system and finally fed into the model. We extracted 16K pairs from Nouns and 9K pairs from Verbs.

Some randomly sampled extractions from the test samples are shown in the figures. On first observation it seems that the model is doing a fair job of aligning similar word pairs that are possibly cognates. We plan to do further manual evaluation of these extraction. The model is able to find word pairs with a common stem without the need of lemmetization. In the case of verbs, it can be observed that the model is able to see through the inflections on the verbs to predict the pairs with similar stems as cognates. Figure 5-8 is most illustrative of the ability of the model. It presents few exceptional cases where the word pairs predicted as cognates are not straightforwardly similar.

HINDI	MARATHI	SCORE
फूलों	फुलांनी	0.985515
फूलों	फुलांचे	0.985508
फूलों	फुलांच्या	0.985507
फ्लाइंग	फ्लाईग	0.985507
लाइन	लाइनची	0.985505
कस्बा	कसबा	0.985502
बाइक	बाईक	0.985439
फूलों	फ़ुलांसठी	0.985436
शांति	``शांती	0.985435
शांति	शांती	0.985435
भागों	भागांवर	0.985435
बसें	बसेंस	0.985429
रोजाना	रोज	0.985425
रेखाओं	रेषांनी	0.985424
पर्वत	पर्वत	0.985225
स्थान	स्थानी	0.985224
शासन	शासनाच्या	0.98522
वर्षों	वर्षांत	0.985219
गलियों	गल्ल्यांच्या	0.985218
फाटक	फाटकात	0.985217

(a) Positive Label

HINDI	MARATHI	SCORE
चोटियाँ	समोर	0.0086563
रूम	खोलीतील	0.0086572
छंतोली	पालखीवर	0.0086578
यात्रा	लोक	0.008658
मुमकनि	लोकसंख्या	0.008658
कुदरत	नसिर्ग	0.008662
मलि	क्षेत्रास	0.0086626
गहराइयों	तलाव	0.0086626
नगिम	महामंडळाच्या	0.0086626
नाम	पॅलेस	0.0086633
इमारतें	नवाबांनी	0.0086637
प्रकृति	नसिर्ग	0.0086646
कलि	मशदि	0.0086649
मूरति	रुपे	0.0086649
तड़के	२७ला	0.008665
रोज	तलाव	0.0086651
तोप	महाराजा	0.0086651
टूरिस्ट	पर्यटनस्थळात	0.0086654

(b) Negative Label

Figure 5-6: Hindi-Marathi Noun Pairs

HINDI	MARATHI	SCORE
बाँध	बांधू	0.985483
बनवा	बनवून	0.985482
बाँध	बांधून	0.985473
बँधे	बांधून	0.98547
बनाने	बांधण्यास	0.985444
बना	बनवा	0.985438
बनाना	बनवणे	0.985435
भर	भरून	0.985432
बना	बनवत	0.985429
स्थति	साकार	0.985426
बनने	बनण्यास	0.985423
बनाने	बनवू	0.985421
बना	बनवतो	0.985417
भगिोकर	भजिवून	0.985415
बनवाने	बनवण्याच्या	0.985414
बनाए	बनवण्यात	0.985356
रुकने	राहण्याची	0.985355
बन	बनत	0.985351
बनते	बनत	0.98535

(a) Positive Label

HINDI	MARATHI	SCORE
मलिेगी	करण्यास	0.0086538
पहुँचने	पोहोचल्यानंत	0.0086556
दौड़	पळू	0.0086578
रखते	केली	0.008658
मलिते	असतात	0.0086597
वर्णति	केले	0.0086604
फैली	करण्यासाठी	0.0086618
मलिाकर	करता	0.0086629
नजर	वसलेल्या	0.0086629
जमावड़ा	असली	0.0086631
गढे	केलेल्या	0.0086648
चाहें	पाहजि	0.0086649
मलिाकर	करुन	0.0086654
जमा	मळिवला	0.0086654
मलिेंगे	चाखण्यास	0.0086657
चलता	वाजता	0.0086665
खुलता	नघिायचे	0.0086666
चलकर	पोहचते	0.0086666
करें	मळिवावी	0.0086675

(b) Negative Label

Figure 5-7: Hindi-Marathi Verb Pairs

HINDI	MARATHI	SCORE
लौंग	लवंग	0.985484
भय	भीती	0.985461
रेखाओं	रेषांनी	0.985424
गलयिों	गल्ल्यांच्या	0.985218
रानी	राणीने	0.985215
लोगों	लोकांचा	0.985077
सुन्दरतम	उद्यानांपैकी	0.985051
मस्जदि	मशीद	0.985039
सींग	शगिी	0.984831
रानी	नैऋत्येच्या	0.984814
नाकवहिीन	नःसंदेह	0.984807
कविंदंती	दंतकथेनुसार	0.984309
मान्यता	माहेर	0.984165

Figure 5-8: Positive Label Noun Pairs Exceptional Cases

Chapter 6

Conclusion

The task of cognate discovery dwells into domain of finding rich hidden representation for words. It is found that simple surface similarity measures like common subsequence based features fail to capture the essence of phonological evolution and sound correspondences. Where there is large drift in the word structures and the characters of the words, these methods fail to capture any similarity between the words. Deep learning models like LSTMs are able to exploit such features to make better judgments on the prediction task.

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. The models generally perform better on Nouns and Adjective words and they also have 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. Introducing concept features to the models in the form of word embeddings is seen to help in improving the results. It is also found that joint training of the models with data from different language families is also useful.

By using deep learning models, the performance boosts are enough to test the model in an open domain. We applied our model to the Hindi-Marathi domain and found that the model is able to segregate the word pairs efficiently.

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*, 2014.

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

[3] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[4] Cecil H. Brown, Eric W Holman, Soren Wichmann, and Viveka Villupillai. Automated classification of the world's languages:a description of the method and preliminary results. *Language Typology and Universals*, (285-308), 2008.

[5] 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.

[6] S.J. Greenhill, R. Blust, and R.D. Gray. The austronesian basic vocabulary database: From bioinformatics to lexomics. *Evolutionary Bioinformatics*, 4:271–283, 2008.

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

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

[10] 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.

[11] 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.

[12] 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.

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

[14] Johann-Mattis List, Philippe Lopez, and Eric Bapteste. Using sequence similarity networks to identify partial cognates in multilingual wordlists. In *Proceedings of the Association of Computational Linguistics 2016 (Volume 2: Short Papers)*, pages 599–605, Berlin, 2016.

[15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[16] 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.

[17] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[18] 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.

[19] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[20] 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.

[21] Taraka Rama. Siamese convolutional networks for cognate identification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan*, pages 1018–1027, 2016.

[22] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. Reasoning about entailment with neural attention. In *ICLR*, 2016.

[23] 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.

[24] Anil Kumar Singh and Harshit Surana. Study of cognates among south asian languages for the purpose of building lexical resources. *Journal of Language Technology*, 2007.

[25] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM, 2015.

[26] 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.

[27] Soren Wichmann and Eric W Holman. Languages with longer words have more lexical change. In Lars Borin and Anju Saxena, editors, *Approaches to Measuring Linguistic Differences*. De Gruyter, 2008.

[28] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[29] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29, 2016.

[30] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.