Discovering Cognates Using LSTM Networks

Shantanu Kumar and **Ashwini Vaidya** and **Sumeet Agarwal**
Indian Institute of Technology Delhi
{ee1130798, ird11278, sumeet}@iitd.ac.in

Abstract

In this paper, we present a deep learning (DL) model for the task of pairwise cognate prediction. We use a character level model with recurrent neural network architecture and attention. We compare the performance of our model with previous approaches on various language families. We are able to show that our model performs better than non-DL methods which exploit surface similarity measures as well as a recent convolutional neural network (CNN) based model for the task. We also employ our model specifically to the domain of discovering cognates from Hindi-Marathi to assist the task of lexical resource creation.

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 word '*Nacht*', both meaning *Night* and English '*Hound*' and German '*Hund*', meaning *Dog* are cognates whose origin can be traced back to Proto-Germanic. Cognate words are not simply the translations of each other in any two languages, but are historically known to have a common origin. For example, the English word '*Hound*' and the Spanish word '*Perro*' both mean *Dog* but are not cognates.

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. Cognate information has also been used in several downstream NLP tasks, like sentence alignment in bitexts (Simard et al., 1993) and improving statistical machine translation models (Kondrak et al., 2003). Additionally, it has been proposed that cognates can be used to share lexical resources among languages that are closely related (Singh and Surana, 2007).

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 such as orthographic or phonetic similarity (Hauer and Kondrak, 2011; Inkpen et al., 2005; List et al., 2016). These are used as features for a classifier to identify cognacy between a given word-pair. Surface similarity measures miss out on capturing generalizations beyond string similarity, as cognate words are not always revealingly similar.

Rama (2015) attempt to identify cognates by looking at the common subsequences present in the candidate word pair. For a cognate pair like the English '*Wheel*' and the Sanskrit '*Chakra*', such an approach fails as they have nothing in common with each other orthographically. In fact, even for a pair like English '*Father*' and Latin '*Pater*', a common subsequence approach completely ignores the similarity between the '*Fa*' and '*Pa*' phonemes, which is a possible indication of cognacy between the pair. Thus, there is a need of 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.

By using DL based models, the need for external feature engineering is circumvented as the system learns to find hidden representations of the input depending on the task in hand. Our paper presents an end-to-end character-level recurrent neural network (RNN) based model that is adapted from a model used on a similar word-level task called RTE (Rocktäschel et al., 2016). Our model is able to outperform both the common subsequence model (Rama, 2015) as well as a recent CNN-based model (Rama, 2016) 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 (Bahdanau et al., 2014), language modelling (Mikolov et al., 2010), information retrieval (Sordoni et al., 2015) and RTE (Bowman et al., 2015). 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. We compare our model against existing supervised approaches. We test the importance of the attention layer and the role of word semantics in the task, along with how properties of the dataset like size and transcription affects the models.

The task of discovering cognates can 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 unlabelled corpus of aligned texts to find cognate pairs.

2 Problem Statement

The task of cognate identification will make use of word lists of different language families taken 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). Table 1 shows a part of a word list that is used for

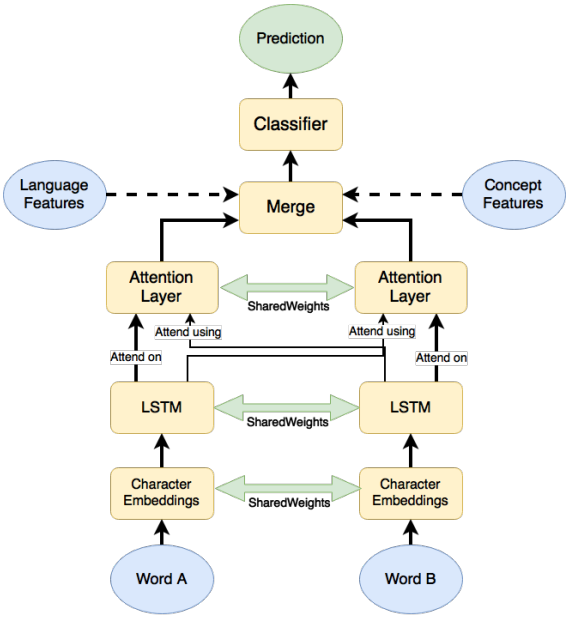

Figure 1: Recurrent Co-Attention Network for Cognate Discovery

the task. Each cell in the table contains a lexical item belonging to a particular language and a particular concept, along with its cognate class ID. If two words have the same cognate class ID, then they are identified as cognates.

The task of pairwise cognate prediction can thus be more formally defined as given two words from a word list belonging to different languages and the same concept, to predict whether the words in the pair are cognates. Thus, a model for this task would take in as input a candidate pair of words and produce as output a single value classifying the pair as cognate or non-cognate.

3 Model

The overall model used in our system is called the Recurrent Co-Attention Model (*CoAtt*). It is adapted from the word-by-word attention model used by (Rocktäschel et al., 2016) for the task of recognising textual entailment (RTE) in natural language sentences. Just as the RTE task involves understanding the semantics of a sentence which is hidden behind sequence of words, the cognate identification task also requires information beyond surface character similarity, which was the motivation to adapt this particular model for our task. The network is

		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 1: Sample Word List from the Indo-European Dataset

illustrated in Figure 1. We have converted the RTE model into a siamese-style network that encodes a word pair in parallel 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 the subsections that follow. The encodings of both the words are merged and passed through a 2-layer neural network classifier 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.

3.1 Character Embeddings

The input words are first encoded into character level embeddings. Character embeddings are a form of distributional representation, where every character of the vocabulary is expressed as a vector in a vector space. This is done using a character level embedding matrix $E \in \mathbb{R}^{n_e \times |C|}$. Here n_e is the dimensionality of the 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.

There are two ways of initialising the character embedding matrix for training. The matrix can be *randomly initialised* by sampling values from a uniform distribution. In such a case, the embeddings are dependent heavily on the training and the random vectors assigned to each character can change during training to such values that are optimal for the task. The other method of initialising the embeddings matrix is by using *phonetic feature vectors* (PV). These phonetic vectors are manually defined binary vectors that are based on various linguistic properties of phonemes such as place of articulation (Dental, Nasal) and manner of articulation (Fricative, Voiced, Lateral).

3.2 LSTM

After the input words to the network are transformed using the character embedding matrix, we encode them with an LSTM. Given an input word $y = \{e_1, e_2, ..., e_n\}$, 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,

$$H = [e_t \;\; h_{t-1}] \tag{1}$$
$$i_t = \sigma(W^i H + b^i) \tag{2}$$
$$o_t = \sigma(W^o H + b^o) \tag{3}$$
$$f_t = \sigma(W^f H + b^f) \tag{4}$$
$$c_t = i_t * tanh(W^c H + b^c) + f_t * c_{t-1} \tag{5}$$
$$h_t = o_t * tanh(c_t) \tag{6}$$

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. [ ] is the concatenation operator and σ is the element-wise sigmoid opertor. The final output of the LSTM gives us a sequence $\{h_1, h_2, ..., h_n\}$ for each word, where $h_j \in \mathbb{R}^{n_h}$.

3.3 Attention Layer

Attention neural networks have been used extensively in tasks like machine translation (Luong et al., 2015), image captioning (Xu et al., 2015) and

visual question answering (Yang et al., 2016). At a high level, attention can be considered as a soft selection procedure, where given a sequence of inputs, one would like to focus or attend on the important part of the sequence with respect to a context. We use this procedure to enhance the representation of the character sequence of a word coming out of the LSTM, by giving it as context the second word. One can compare the attention mechanism with the common subsequence model. In a common subsequence model, one makes a hard selection procedure by using as features only the common subsequences of both the words. In the attention model, the network makes a soft selection while focusing on those parts of the sequence which are important with respect to the other word.

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 an attention-weighted representation $r \in \mathbb{R}^{n_h}$ of Y as,

$$M = tanh(W^y Y + W^h h \otimes e_L) \quad (7)$$

$$\alpha = softmax(w^T M) \quad (8)$$

$$r = Y\alpha_t^T \quad (9)$$

The outer product $W^h h \otimes e_L$ repeats the linearly transformed h as many times as there are characters in Y (L times). Using the mechanism followed by Rocktäschel et al. (2016) 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 $H = \{h_1, h_2, ..., h_N\} \in \mathbb{R}^{n_h \times N}$.

$$M_t = tanh(W^y Y + (W^h h_t + W^r r_{t-1}) \otimes e_L) \quad (10)$$

$$\alpha_t = softmax(w^T M_t) \quad (11)$$

$$r_t = Y\alpha_t^T + tanh(W^t r_{t-1}) \quad (12)$$

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 is the 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 r_{HY} and r_{YH} vectors. This method of making both the character sequences attend over each is called the Co-Attention mechanism.

3.4 Language & Concept Features

It is known that some languages are more closely related to each other as compared to others. For example from Table 1 one can see that *Hindi* is more related to *Marathi* than to *French*. That is a candidate word pair with words from *Hindi* and *Marathi* is more likely to be a cognate pair as compared to a word pair with words from *Hindi* and *French*. This information about language relatedness can be exploited by using as features a 2-hot encoding vector that represents the respective languages of the two input words. During training, the network can use these features to learn automatically which language pairs are closely related from the data.

Similar to language information, we hypothesise that information about the semantics of the input words can also be useful features for the classifier. The word semantics inherently contain information like the part-of-speech (POS) category of the word. Our initial exploration of the data showed that certain POS categories such as question words or prepositions tend to have greater divergence in their cognate classes as compared to nouns or adjectives. We use the GloVe word embedding (Pennington et al., 2014) for the English concept of the input word pair as the concept feature vector. 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.

4 Experiments

In the subsections below we describe the datasets that we used and the comparisons we made with our models. This is followed by the three experiments we conducted with the datasets.

4.1 Datasets

We make use of three datasets in our work which come from three language families. These families make a good test set as they vary widely in terms

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 2: Statistics about the different language families

	Indo-European		Austronesian		Mayan	
	Total	*Positive*	*Total*	*Positive*	*Total*	*Positive*
Cross Language Evaluation						
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
Cross Concept Evaluation						
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 3: Number of word pairs obtained for both modes of evaluation from different language families

of the number of languages, concepts and cognate classes. The first and primary dataset that we use is the IELex Database, which contains cognacy judgements from the Indo-European language family. The dataset is curated by Michael Dunn[1]. Second, we include a dataset taken from the Austronesian Basic Vocabulary project (Greenhill et al., 2008), and a third dataset from the Mayan family (Wichmann and Holman, 2008).

There are several differences in transcription in each of these datasets. While Indo-European is available in IPA, ASJP and a coarse 'Romanized' IPA encoding, the Mayan database is available in the ASJP format (similar to a Romanized IPA) (Brown et al., 2008) and the Austronesian has been semi-automatically converted to ASJP (Rama, 2016). We use subsets of the original databases due to lack of availability of uniform transcription.

The Indo-European database contains words from 52 languages for over 200 concepts, while the Austronesian contains words from 100 languages and as many concepts, as can be seen in Table 2. The Mayan dataset is comparatively very small with only 100 concepts from 30 languages. The Austronesian dataset also contains the largest number of cognate classes as compared to the other two. The number of sample pairs obtained from each dataset are mentioned in Table 3. The small size of the Mayan dataset especially poses a challenge for training the deep learning models which is addressed in the later sections.

4.2 Evaluation

There are two methods of evaluation that we follow in our experiments, namely the **cross-language** evaluation and **cross-concept** evaluation. In cross-language evaluation, the training and testing sample pairs are created using exclusive sets of languages, whereas in cross-concept they come from exclusive set of concepts. This can be done by dividing the word list in Table 1 on the basis of rows or columns, respectively for cross-language or cross-concept, into training and testing sets and then forming the sample pairs from them. Both words in a sample pair always belong to the same concept or meaning. A sample pair is assigned a positive cognate label if their cognate class ids match.

We report the *F-score* and the area under the PR curve (*AUC*) as a measure of performance for all the models. *F-score* is computed as the harmonic mean of the *precision* and *recall*[2]. Since the dataset is heavily biased and contains a majority of negative

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

[2] Precision and Recall is computed on positive labels at 0.5 threshold. Precision = TP/(TP+FP), Recall = TP/(TP+FN), TP: True Positives, FP: False Positives, FN: False Negatives

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
Phonetic CNN	73.7	86.1	54.6	68.0	72.8	85.0
Character CNN	75.3	85.3	62.2	71.6	75.9	85.7
LSTM + No Attention	56.7	59.0	51.2	55.2	60.6	67.1
LSTM + Uniform Attention	52.8	59.4	49.8	52.7	60.8	66.1
Co-Attention Model	83.8	89.2	69.0	77.5	67.1	67.7
+ PV	85.1	92.4	70.2	79.3	63.6	71.3
+ PV + CF	**86.2**	**93.0**	**70.5**	**79.7**	**81.5**	**89.0**

Table 4: Cross Language Evaluation Results [PV: *Phonetic Feature Vectors*, CF: *Concept Features*]

cognate sample pairs, we do not use *accuracy* as a measure of performance.

4.3 Baseline Models

We compare our model against a model based on surface similarity (subsequence model) and two CNN based DL models as described below:

Gap-weighted Subsequences : This model refers to the common subsequence model (Rama, 2015) mentioned earlier. The author uses a string kernel based approach wherein he defines a vector for a word pair using all common subsequences between them and weighting the subsequence by their gaps in the strings. The results reported for the subsequence model were found by implementing the model using the paper as the original code was not available.

Phonetic CNN & Character CNN : These models are variations of the siamese-style CNN-based models (Rama, 2016). The models are inspired from convolutional networks used for image-similarity tasks. The *Phonetic CNN* model uses the manually defined phonetic feature vectors as character embeddings in the network (but they are fixed during training), whereas the *Character CNN* model uses a 1-hot encoding to represent the different characters. The results reported for these models were found by rerunning the original code from the author on the prepared datasets[3].

LSTM + No Attention & LSTM + Uniform Attention : We also introduced two sanity-check baseline models to test the attention layer of the *CoAtt* model. The *LSTM + No Attention* model removes the Attention layer from the *CoAtt* model, while the *LSTM + Uniform Attention* model does a simple average rather than a weighted average in the attention layer.

4.4 Experiment 1: Cross Language Evaluation

As can be observed in Table 4, the *CoAtt* model performs significantly better than the baseline models (both CNN and subsequence-based). The *LSTM + No Attention* and *LSTM + Uniform Attention* models reflect the importance of the attention layer. The soft selection procedure of attention is able to highlight the important features of either word automatically for the classification task.

The additional features added to the *CoAtt* model help to improve its performance further. Initialising the character embeddings with the manually defined phonetic vectors (*+ PV* models) increases the *AUC* by around 3%. Further, addition of the *Concept features* discussed earlier, is also found to be useful (*+ CF* model). The concept features can intuitively help the model to indicate different thresholds to use depending on the kind of variation in cognates observed for that concept.

The Mayan language family is linguistically and geographically less diverse than the other datasets (see also Table 2). The addition of *Concept features* significantly improves the *CoAtt* model on the Mayan dataset. This shows the usefulness of semantic information for cognacy detection when less data is available. For Indo-European and Austronesian, on the other hand, the *CoAtt* performs more effec-

[3]It can be noted that there is a difference in the reported F-score of the CNN models as compared to the original paper Rama (2016). This is because we report the f-score with respect to the positive labels only, whereas the original paper reported the average f-scores of positive and negative labels (Observed from the implementation in author's code)

Model	Mayan	
	F-Score	*AUC*
Gap-Weighted Subsequence	71.8	81.8
Phonetic CNN	72.8	85.0
Character CNN	75.9	85.7
Co-Attention Model	67.1	67.7
+ PV	63.6	71.3
+ PV + PreT (Indo-European)	82.5	90.6
+ PV + PreT (Austronesian)	**83.5**	**91.2**

Table 5: Cross Language Evaluation Results for Mayan Dataset with Pre-Training
[PV: *Phonetic Feature Vectors*, PreT: *Pre-Training on another dataset*]

tively than the subsequence models or the Phonetic or Character CNN, even without the concept features.

Table 4 shows that the *CoAtt* model does not train well on the Mayan dataset directly. It is found that the loss does not decrease a lot during training as compared to the other models. This poor performance on the Mayan dataset is associated with its small size, and relatively fewer number of languages and this does not prove sufficient for training the *CoAtt* network. We justify this hypothesis in the following section with the *Cross-Family Pre-training* experiment.

4.5 Experiment 2: Cross-Family Pre-training

The three different language families with which we work have completely different origins and are placed across different regions geographically. We test whether any notion of language evolution might be shared amongst these independently evolved families. This is done through the joint learning of models. The network is instantiated with the combined character vocabulary of two datasets. Then the model is trained on one dataset till the loss saturated. This is followed by the training on a second dataset, starting from the weights learned from the pre-training.

It is found that such a joint-training procedure helps the *CoAtt* model on the Mayan dataset significantly. The pre-training procedure is able to provide a good initialisation point to start training on the Mayan dataset. The pre-trained models perform better than the baseline models (*PreT* models in Table 5). This shows that pre-training *CoAtt* is helpful and also points to the fact that some regular sound changes could potentially be shared across language families (keeping in mind that the datasets are coarsely transcribed, with a few similarities that may be coincidental).

4.6 Experiment 3: Cross Concept Evaluation

The cross-concept evaluation test can be thought of as a more rigorous test as the models have not seen any of the similar word structures during training. The testing sample words are from completely different concepts. Words coming from different concepts would have different sequence structures altogether. A model that predicts cognate similarity in such a case would have to exploit phonetic similarity information in the context of cognates.

The results for the cross concept evaluation tests are listed in Table 6. It is observed that the *CoAtt* model is able to reach close to the performance of the CNN based models. With the phonetic feature vectors and extra *Language features*, the model performs slightly better than the baselines. We note that the initialized embeddings consisting of phonetic feature vectors are useful for this task as compared to the randomly initialized cases.

The lower performance values on the cross-concept task as compared to the cross-language task can be associated due to the nature of this experiment. Even for a human historical linguist, this task is unnatural as traditionally a new word is assigned to a cognate class after it is compared against words of the same meaning that are known to be cognates. Therefore, there is sufficient information about that concept available to a human being that is performing this task.

Model	Indo-European	
	F-Score	*AUC*
Gap-weighted Subsequence	51.6	62.0
Phonetic CNN + LF	66.4	73.2
Character CNN + LF	63.5	70.5
Co-Attention Model	64.8	69.8
+ CF	64.1	70.6
+ LF	65.6	70.8
+ PV + CF	69.0	74.9
+ PV + LF	**69.1**	**75.0**

Table 6: Cross Concept Evaluation Results for Indo-European
[PV: *Phonetic Feature Vectors*, CF: *Concept Features*, LF: *Language Features*]

4.7 Hindi-Marathi Domain Experiment

We also applied our *CoAtt* model to the domain of Hindi-Marathi. The model used was trained on the Indo-European dataset with IPA transcription. It should be noted that the Indo-European 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 a Hindi-Marathi parallel corpus downloaded from TDIL. This dataset provides a large part of the vocabulary from the both the languages to search for cognates. 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.

Our model is does a fair job of aligning similar word pairs that are possibly cognates. We tested the performance of the model by randomly sampling 50 word pairs each from NOUNs and VERBs and manually annotating them. We found that our model gives an 80% accuracy on Verbs and 74% accuracy on Nouns. The model is able to find word pairs with a common stem without the need of lemmatization. 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.

This implies that the model can be used to find cognate pairs across closely related languages in order to share resources that are missing in one of the pairs. As an example, lexical resource sharing between Urdu and Hindi was carried out for verb subcategorization frames (Bhat et al., 2014).

5 Analysis

5.1 Concept Wise Performance

We observed the performance of the models over individual concepts in the test set samples for the Indo-European dataset. It is observed that the performance of *CoAtt* is more uniform throughout the concepts as compared to the more varied distribution of the other models. For concepts like WHAT, WHO, WHERE, HOW, THERE where both the subsequence model and the CNN model performed poorly, the *CoAtt* model is able to achieve high scores. The *CoAtt* model performs poorly on concepts like AT, IF, BECAUSE, GIVE. By looking at the samples, it is found that these concepts are very lexically divergent. They also have many cognate classes due to which the training data is heavily biased by negative samples and contain only a handful of positive cognate pair examples. The *CoAtt* model on these concepts is still able to perform almost as well as the CNN model, whereas the subsequence model does pretty bad due to the almost zero overlap of subsequences.

POS	CoAtt	CNN	Subseq
Noun	*0.80*	*0.74*	*0.66*
Pronoun	*0.73*	*0.52*	*0.22*
Verb	*0.82*	*0.76*	*0.62*
Adverb	*0.76*	*0.63*	*0.50*
Adjective	*0.80*	*0.65*	*0.60*
Preposition	*0.63*	*0.59*	*0.30*
Determiner	*0.85*	*0.74*	*0.38*

Table 7: F-Scores for various model on different POS categories [*CoAtt*: Co-attention model, *CNN*: Phonetic CNN model, *Subseq*: Gap-weighted subsequence model]

Concept	CoAtt	CNN	Subseq
WHAT	*0.97*	*0.39*	*0.20*
THERE	*0.86*	*0.78*	*0.17*
HOW	*0.83*	*0.27*	*0.16*
WHERE	*0.80*	*0.43*	*0.05*
WHO	*0.78*	*0.45*	*0.04*
IN	*0.70*	*0.64*	*0.17*
GIVE	*0.56*	*0.56*	*0.31*
AT	*0.50*	*0.50*	*0.29*
IF	*0.46*	*0.33*	*0.00*
BECAUSE	*0.33*	*0.00*	*0.00*

Table 8: F-Scores for various model on different concepts [*CoAtt*: Co-attention model, *CNN*: Phonetic CNN model, *Subseq*: Gap-weighted subsequence model]

5.2 Transcription Tests

The Indo-European dataset, as we had mentioned earlier, is available in two different transcriptions. The dataset is transcribed in ASJP like the other datasets and is also transcribed in IPA, which is a much finer representation as compared to ASJP. Table 9 shows the results of the *CoAtt* model on the Indo-European dataset in either transcription. Note that neither of these models include the phonetic features vectors since we only have these vectors for the ASJP character vocabulary and not the IPA. It is found that the overall performance is not really affected by the difference in transcription. Thus, the finer IPA transcription does not give any immediate added advantage over using the ASJP transcription.

However, there is a difference in the performance of the model in either transcription on different concepts. As an example, Table 10 shows few test word pairs from the concept SWIM for different language

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

Table 9: Cross Language Evaluation Transcription Tests [CF: *Concept Features*]

ASJP		IPA	
Word A	*Word B*	*Word A*	*Word B*
swim	sinda	swɪm	’sɪn̥da
swim	zwem3n	swɪm	zwɛmən
swim	svim3n	swɪm	ʃvɪmən
swim	swem3	swɪm	’sʊɸmːə
swim	sima	swɪm	’simːa

Table 10: Sample word pairs from the concept SWIM from Indo-European dataset. All samples are cognate pairs.

pairs in either transcription. All these pairs are true cognates. 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 the very fine 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 in the *CoAtt + CF* model makes it predict all of the samples as cognates. Thus, perhaps adding the concept features signals the model to relax the degree of overlapping of phonemes for this case and become less strict in predicting cognates.

6 Discussion

There are several interesting insights that we have observed from our experiments. Primarily we find that the co-attention model is effective at the character level for the cognate classification task. The model was successfully adapted from the word level model used for RTE and worked well in the domain of the character level task. The attention layer especially is very important for the model, as it found that the LSTM by itself is not able to successfully extract enough information from the words for cognate identification. We found that the size and prop-

erties of the dataset also effects the training and performance of the models. For a small dataset like Mayan, joint learning of models proves beneficial as information drawn across datasets overpowers the lack of data. We find that additional features that provide information about the word semantics are also useful for improving the models. There is a distribution of the variation in the evolution of words over the word semantics, which can be exploited by such models. The transcription of the data into a finer character vocabulary does not have any added advantage in the overall performance, but it is observed that there are cases where the coarser representation helps the model to realise correspondence.

7 Conclusion

The task of cognate discovery dwells into the 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. 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.

References

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

Riyaz Ahmad Bhat, Naman Jain, Ashwini Vaidya, Martha Palmer, Tafseer Ahmed Khan, Dipti Misra Sharma, and James Babani. 2014. Adapting predicate frames for urdu propbanking. In *Proceedings of the EMNLP 2014 Workshop on Language Technology for Closely Related Languages and Language Variants*, pages 47–55. ACL.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on EMNLP*. ACL.

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

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

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

Diana Inkpen, Oana Frunza, and Grzegorz Kondrak. 2005. 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.

Grzegorz Kondrak, Daniel Marcu, and Kevin Knight. 2003. Cognates can improve statistical translation models. In *Proceedings of HLT-NAACL 2003–short papers- Volume 2*, NAACL-Short '03, pages 46–48. ACL.

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

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

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

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

Taraka Rama. 2015. Automatic cognate identification with gap-weighted string subsequences. In *Proceedings of the 2015 Conference of NAACL: Human Language Technologies*, pages 1227–1231.

Taraka Rama. 2016. Siamese convolutional networks for cognate identification. In *Proceedings of COLING 2016*, pages 1018–1027.

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

Michel Simard, George F Foster, and Pierre Isabelle. 1993. Using cognates to align sentences in bilingual corpora. In *Proceedings of the 1993 Conference of CASCON*, pages 1071–1082. IBM Press.

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

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. 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.

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

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

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