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 modeling (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 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 based model (Rama, 2015).

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.

2 Datasets

The task of cognate identification will make use of word lists of different language families taken from the basic vocabulary e.g. 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). A word list can be considered as a table where the different rows and columns represent different languages and concepts respectively. Each cell in the table contains a lexical item along with its cognate class ID which helps to determine if two words are cognates.

We make use of three datasets in our work which come from three different language families. These families make a good test set as they vary widely in terms 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 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) (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 IELex database contains words from 52 languages for over 200 concepts, while the Austronesian contains words from 100 languages and as many concepts. 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 samples obtained from each dataset are mentioned in Tables 1 and 4. The small size of the Mayan dataset especially poses a challenge for training the deep learning networks which is addressed in the later sections.

We also use the TDIL Hindi-Marathi sentence-aligned corpus as the large unlabeled data for our final model. This dataset provides a large part of the vocabulary from the both the languages to search for cognates.

3 Approach

The overall model used in our system is called the Recurrent Co-Attention Model (*CoAtt*). It is adapted

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

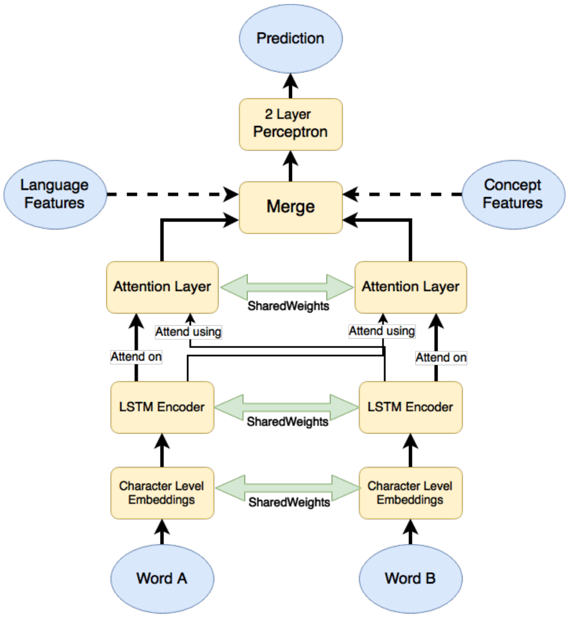

Figure 1: Recurrent Co-Attention Network for Cognate Discovery

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 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 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.

(Rama, 2016) 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 tuned for the task in hand. Thus, we use Rama's manually defined embeddings as an initialisation for the character embeddings and let these weights be trained during learning. It is found that such a method of embedding initialisation results in a better traning of the model as compared to a random initialisation.

3.2 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,

$$H = [e_t h_{t-1}] \quad (1)$$

$$i_t = \sigma(W^i H + b^i) \quad (2)$$

$$o_t = \sigma(W^o H + b^o) \quad (3)$$

$$f_t = \sigma(W^f H + b^f) \quad (4)$$

$$c_t = i_t * tanh(W^c H + b^c) + f_t * c_{t-1} \quad (5)$$

$$h_t = o_t * tanh(c_t) \quad (6)$$

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. The final output of the LSTM gives us a sequence $\{h_1, h_2, ..., h_n\}$ for every word, where $h_j \in \mathbb{R}^{n_h}$.

3.3 Attention

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). 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 characted encoding of the second word, which helps to generate a weighted representation of the first word that includes its important segments with respect to its similarity with 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,

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

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

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

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 $H = \{h_1, h_2, ..., h_N\} \in \mathbb{R}^{n_h \times N}$.

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

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

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

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 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.

$$r^* = [r_{HY} r_{YH}] \quad (13)$$

This method of making both the sequences attend over each is called the *Co-Attention* model.

3.4 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. (Rama, 2016) 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.

Since we would primarily work in a Cross Language mode of evaluation, where the training and testing languages come from exclusive sets, it is intuitive that such *Language Features* will not help to improve of the model in such a setting. Since the languages of the training and testing set do not overlap, the relevant language feature weights for the test set are never learnt. Any information about the affinity or interaction between the languages in the training set is not useful for the languages in the testing set.

The *Language Features* are thus only used during the Cross Concept mode of evaluation, where there is an overlap in the training and testing languages. We follow the same approach used by (Rama, 2016) 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 multi-layer perceptron for classification.

3.5 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 (Pennington et al., 2014). 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.

4 Experiments

We primarily follow a Cross Language evaluation procedure, where the training and testing sample pairs are created using exclusive sets of languages. A random set of 70% of the languages is set as the training set of languages and the rest as testing set. Both words in a sample pair belongs to the same concept or meaning. A word pair is assigned a positive cognate label if their cognate class ids match. The number of sample pairs obtained for training and testing from the different datasets formed using cross language evaluation test can be found in Table 1.

We conducted ablation tests to study the contribution of different features and conducted tests on how the different levels of transcription affects the performance of the models. In the subsections below we describe the results from these different tests.

4.1 Evaluation Metric

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

4.2 Baseline Models

We compare our model against the following baseline models.

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 reimplementing 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 CNN networks used for image-similarity tasks. The *Phonetic CNN* model uses the manually defined (fixed) character embeddings in the network, 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 base-

[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

[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. 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)

	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 1: Data size for Cross Language Evaluation

Model	**Indo-European**	
	F-Score	*AUC*
Gap-Weighted Subsequence	59.0	75.5
PhoneticCNN	73.7	86.1
CharacterCNN	75.3	85.3
LSTM + No Attention	56.7	59.0
LSTM + Uniform Attention	52.8	59.4
Co-Attention Model	83.8	89.2
+ IE	85.1	92.4
+ IE + CF	**86.2**	**93.0**

Table 2: Cross Language Evaluation Results for Indo-European Dataset
[IE: *Initialised Embeddings*, CF: *Concept Features*]

line models to test the attention layer of the *CoAtt* model. The *LSTM + No Attention* model removes the Co-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.

4.3 Experiments with Indo-European

As can be observed in Table 4.1, the *CoAtt* model performs significantly better than the CNN and the subsequence based models. The *LSTM + No Attention* and *LSTM + Uniform Attention* models reflect the importance of the attention layer adapted from the RTE model in the network, as without it the model does not perform very good. EXPAND

A few additional features added to the *CoAtt* model helps to improve it even further. Initialising the character embeddings with the manually defined vectors (+ *IE* models) increases the *AUC* by around 3%. Further, addition of the *Concept features* discussed earlier, is also found to be useful (+ *CF* model). EXPAND

4.3.1 Transcription Tests

4.4 Experiments with Multiple Datasets

Table 3 lists the results of the models on these datasets. We observe a similar trend for the models on the Austronesian and Mayan datasets as well. However, the *CoAtt* model does not train well on the Mayan dataset directly. This poor performance on the Mayan dataset is associated with its small size. The Mayan dataset being significantly smaller than the other datasets, does not prove sufficient for training the *CoAtt* network. We justify this hypothesis subsequently with the *Cross-Family Pretraining* experiment. The *Concept features* are again found useful to improve the *CoAtt* model, especially on the Mayan dataset, where the extra information about the meaning of input word pair helps the model to cross the baseline results.

Cross Family Pre-training Experiments

The three different language families with which we work have completely different origins and are placed across different regions geographically. We test if any notion of language evolution is still 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 sig-

Model	Austronesian		Mayan	
	F-Score	*AUC*	*F-Score*	*AUC*
Gap-Weighted Subsequence	58.8	68.9	71.8	81.8
PhoneticCNN	54.6	68.0	72.8	85.0
CharacterCNN	62.2	71.6	75.9	85.7
Co-Attention Model	69.0	77.5	67.1	67.7
+ IE	70.2	79.3	63.6	71.3
+ IE + CF	**70.5**	**79.7**	81.5	89.0
+ IE + PreT (Indo-European)	-	-	82.5	90.6
+ IE + PreT (Austronesian)	-	-	**83.5**	**91.2**

Table 3: Cross Language Evaluation Results for Austronesian and Mayan Datasets
[IE: *Initialised Embeddings*, CF: *Concept Features*, PreT: *Pre-Training on another 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 4: Data size for Cross Concept Evaluation

nificantly. The pretraining procedure is able to provide a good initialisation point to start training on the Mayan dataset. The pretrained models perform significantly better than the baseline models (*PreT* models in Table 3). This also provides evidence to support our hypothesis that the *CoAtt* was not able to learn on the Mayan dataset because of 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.

4.5 Cross Concept Evaluation

We also conducted cross concept evaluation experiments, where the training and testing word pairs were formed using exclusive sets of *concepts* or *meanings*. For this, we followed the same scheme as done by (Rama, 2016), wherein we took the first 70% of the concepts as training concepts and the remaining concepts as testing concepts. The training and testing set size details formed using cross concept evaluation test can be found in Table 4. The results for the cross concept evaluation tests are listed in Table 4.4.

It is observed that the *CoAtt* model is able to reach close to the performance of the CNN based models. With the initialised embeddings and extra *Language features*, the model performs slightly better than the baselines. `EXPAND`

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 absolutely different concepts. Words coming from different concepts would have different sequence structures altogether and for a model to predict cognate similarity in such a case would definitely have to exploit phoneme similarity information in the context of cognates.

5 Analysis

5.1 Character Embeddings

5.2 Concept Wise Performance

In this analysis of the models, we looked at the performance of various models over the individual concepts in the test set samples. It is observed that the performance of *CoAtt* 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

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
+ PreT (Austronesian)	65.8	71.0
+ IE + CF	69.0	74.9
+ IE + LF	**69.1**	**75.0**

Table 5: Cross Concept Evaluation Results for Indo-European
[IE: *Initialised Embeddings*, CF: *Concept Features*, LF: *Language Features*, PreT: *Pre-Training on another dataset*]

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

Table 6: *CoAtt* vs *Subseq* model on various concepts (F-Score)

5.3 Hindi-Marathi Domain Adaptation

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.

On first observation it seems that the model is doing 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 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.

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.

References

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

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 Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

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 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.

Johann-Mattis List, Philippe Lopez, and Eric Bapteste. 2016. 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.

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 the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, May 31–June 5, 2015 Denver, Colorado, USA*, pages 1227–1231.

Taraka Rama. 2016. 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.

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* *the Centre for Advanced Studies on Collaborative research: distributed computing-Volume 2*, 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.