Discovering Cognates Using LSTM Networks

Anonymous EMNLP submission

Abstract

In this paper, we present a deep learning model for the 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 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 '*Nacht*', both meaning *Night* and English '*Hound*' and German '*Hund*', meaning *Dog* are cognates whose origin can be traced back to Proto-Germanic.

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 (**?**) and improving statistical machine translation models (**?**). 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 (e.g. orthographic or phonetic similarity). (Hauer and Kondrak, 2011) (Inkpen et al., 2005) (List et al., 2016). 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 paper shows that an end-to-end recurrent neural network based model is able to outperform both surface similarity as well as a recent CNN based Siamese-style 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 model described in (Rama, 2015) and its variants.

We also compare our model's performance with existing supervised approaches, including the sub-

sequence model described in (Rama, 2015) (and its variants). Even if these models are provided with richer character representations in the form of IPA encoding, they are still unable to identify cognate pairs from very broad cognate classes. On the other hand, the LSTM model, even when pretrained with data from a completely unrelated language family has a much lower error rate. *Some lines need to be added here about the result pairs*

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 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). The word lists contain information about a particular word, its language family and a cognate class ID. Words are provided for a number of languages for a set of concepts that are from the basic vocabulary.

We make use of three datasets in our work. The first of these is the IELex Database, which contains cognacy judgements from the Indo European language family, curated by Michael Dunn . 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 Indo European database has the largest number of languages: 139 as compared to Mayan, which is a much smaller language family consisting of only 30. The Austronesian dataset contains the largest number of cognate classes as compared to the other two.

For the purposes of this task, we form word pairs using words from the same concept and such a pair is assigned a positive cognate label if its cognate class ids match. In this way, we get 525, 941 word pairs from Austronesian, 326, 758 pairs from Indo-European and 63, 028 pairs from Mayan.

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 Related Work

Orthographic features based classifier : Hauer and Kondrak (2011) 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. The also use features that encode the degree of affinity between pairs of languages.

Gap-weighted common subsequences : Rama (2015) 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 subsequence based features outperform orthographic word similarity measures.

Siamese ConvNet model : In a recent work, T. Rama introduces CNN based siamese-style model (Rama, 2016) for the task. The model is inspired by image-similarity CNN models. Instead of CNNs, we propose the use of LSTM networks. As these 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 more 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 (Yin et al., 2017).

4 Gap-weighted subsequence model

Before coming to the LSTM based model, we performed a thorough analysis on the performance of

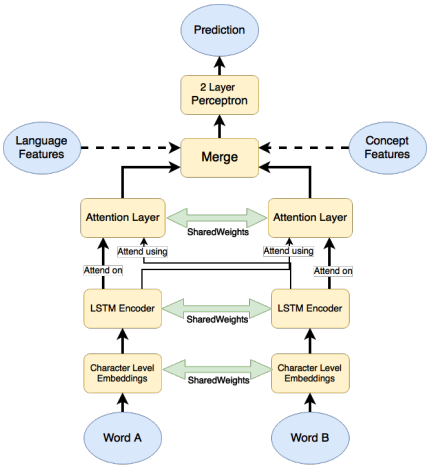

Figure 1: Recurrent Co-Attention Network for Cognate Discovery

the gap-weighted subsequence model. There were two fundamental drawbacks in using the common-subsequence model (Referred to as *Multiplicative* model).

Firstly, since the model only looked at common subsequences, it ignored vital information about closely related subsequences such as *'fa'* in FATHER and *'pa'* in PATER which are cognate words. Secondly, this also resulted in very sparse vectors as the feature space size was huge. To overcome these problems, we tried smoothing techniques which resulted in models Hy-Avg, Hy-Norm and Additive.

4.1 Error Analysis

4.2 Modified model

5 Approach

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 Rocktäschel et al. (2016) for the task of recognising textual entailment in natural language sentences. The network used is illustrated in Figure 1. 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 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.

5.1 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.

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 useful for the task in hand.

5.2 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 $h_j \in \mathbb{R}^{n_h}$.

5.3 Attention

Attention neural networks have been used extensively in tasks like machine translation `cite`, image captioning `cite`, summarisation `cite` and visual question answering 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.

5.4 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.

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.

5.5 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.

5.6 Cross Family Pre-training

The three different language families with which we work have completely different origins and are placed across different regions geographically and in time. 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. Particularly since the size of the Mayan dataset available is quite small, it would be interesting to note if some information or feature learning from the other language families could prove beneficial for it. 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 initialisation.

6 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 cross-family evaluation tests. Finally we conducted tests on how the different levels of transcription affects the learning and performance of the models.

6.1 Evaluation Metric

We report the *F-score* as a measure of performance for all our models. *F-score* is computed as the harmonic mean of the *precision* and *recall*[1]. Since the dataset is heavily biased and contains a majority of negative samples (As can be seen in Tables 1 and 3), *accuracy* is not a good measure of performance to compare the models.

6.2 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 6. The results for the cross language evaluation tests are listed in Table 6.

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

[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 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 2: Cross Language Evaluation Results

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.

6.3 Cross Concept Evaluation

For the cross concept evaluation test, 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 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 6.2. The results for the cross concept evaluation tests are listed in Table 6.2.

It is observed that the *CoAtt* model gives an almost equal performance for the Indo-European and the Austronesian datasets as compared to the CNN models. In fact, it can be seen that within the CNN models, *PhoneticCNN + Lang* model performs better on Indo-European whereas *CharCNN + Lang* performs better on Austronesian dataset. The *CoAtt* model has a performance equivalent to the best CNN model for each dataset. It is also observed that the *CoAtt* model again does not learn very well for the Mayan dataset. 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 can be thought of as 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 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.

7 Analysis

7.1 Concept Wise Performance

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.

	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 3: 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 4: Cross Concept Evaluation Results

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

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

Table 6: *CoAtt* model using IPA vs ASJP transcription on various concepts (F-Score)

7.2 Transcription Tests

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.

7.3 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.

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 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.

8 Conclusion

Acknowledgments

Do not number the acknowledgment section.

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 *Proceedings of 5th International Joint Conference on Natural Language Processing*. Citeseer, pages 865–873.

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.

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)*. Berlin, pages 599–605. http://anthology.aclweb.org/P16-2097.

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. http://www.aclweb.org/anthology/D14-1162.

Taraka Rama. 2015. Automatic cognate identification with gap-weighted string subsequences. In *HLT-NAACL*. 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*.

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*. ACM, pages 553–562.

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.

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