Proceedings of the 13th Workshop on Building and Using Comparable Corpora, pages 65–68
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

BUCC2020: Bilingual Dictionary Induction using Cross-lingual Embedding

Sanjanasri JP, Vijay Krishna Menon, Soman K P
Center for Computational Engineering and Networking (CEN), Amrita School of Engineering,
Amrita Vishwa Vidyapeetham, Coimbatore- 641112, India
{p_sanjanashree, m_vijaykrishna}@cb.amrita.edu, kp_soman@amrita.edu

Abstract

This paper presents a deep learning system for the BUCC 2020 shared task: Bilingual dictionary induction from comparable corpora. We have submitted two runs for this shared Task, German (de) and English (en) language pair for "closed track" and Tamil (ta) and English (en) for the "open track". Our core approach focuses on quantifying the semantics of the language pairs, so that semantics of two different language pairs can be compared or transfer learned. With the advent of word embeddings, it is possible to quantify this. In this paper, we propose a deep learning approach which makes use of the supplied training data, to generate cross-lingual embedding. This is later used for inducting bilingual dictionaries from comparable corpora.

1. Introduction

In machine translation, the extraction of bilingual dictionaries from parallel corpora have been conducted very successfully. Theoretically, it is possible to extract multilingual lexical knowledge from comparable rather than from parallel corpora as the former is more abundant than the latter. To implement any machine learning tasks in Natural Language processing (NLP), it is necessary to quantify the semantics (meaning) of the word in a language. Representation of semantics of a word quantitatively is made possible with the evolution of word embeddings (Mikolov et al., 2013a);they are dense distributed vector representations of words. This numerical representation mimics the linguistic phenomena such as lexical, syntactic, morphological and other complex phenomena such as ambiguity, negation, lemmas, inference and so on. Contemporary vector training algorithms such as GloVe and Word2Vec (Pennington et al., 2014; Mikolov et al., 2013c) are more accurate in capturing word to word semantics than conventional vector space models such as Latent Semantic Analysis (LSA) (Deerwester et al., 1990) and perform better in almost all downstream tasks in NLP (Treviso et al., 2017; Bansal et al., 2014; Guo et al., 2014).

In this paper, we train a transfer learning model/Deep Neural Network(DNN) using pre-trained monolingual embeddings of the given bilingual dictionary. Source embedding is given to DNN, so it generates a target embedding. The generated embedding is compared with the original (monolingual) embedding to find the closest embedding. The word corresponding to the closest embedding is identified as the word translation of the given source word. Simply, we perform a reverse look up to identify the correct word translation from the original embedding given the transfer learned embedding.

Section 2 describes the systems that are experimented for this task. Section 3 gives the details of the data used for this experimentation. Section 4 gives insight about the computational complexity. Section 5 details the evaluation method carried out to justify the system. Section 6 gives the results of the systems. Section 7 gives some concluding inferences and remarks.

2. System Description

The main objective of this work is to develop an efficient and accurate transfer learning method for attaining *'cross-lingual'* word embeddings without the large monolingual and bilingual corpus. The system was developed in four stages; each improving the accuracy. The test data result submitted is run on the system that gave us the best accuracy. System one derives the translation matrix for the language pair using the standard method (direct linear mapping) (Mikolov et al., 2013b). Given pairs of word vectors in a source and target language $< x_i, y_i >_{i=1}^{n}$ respectively, we calculate the transformation matrix (W) between the two languages utilizing pseudo inverse $X^+ = (X^T X)^{-1} X^T$, as follows:

$$\begin{aligned} XW &= Y \\ W &= X^+ Y \end{aligned} \tag{1}$$

System two and three deploy deep learning network to learn the mapping between two different language embeddings. In this method we train a transfer learning model to generate cross-lingual embedding. Our method has obvious advantages over the bilingual embedding (Chandar et al., 2014; Gouws et al., 2015), because bilingual embeddings might compromise semantics in order to project each language (source and target) into the common vector space; the semantic properties pertaining to the language might be lost as the model considers only the common semantic features between the languages. Our method generates cross-lingual embedding by projecting the vectors of one language into another language space without compromising the actual semantics of both the languages. Also, to train an efficient bilingual embedding, it is necessary to have large bilingual resources. The transfer learning model can generate better cross-lingual embedding when trained with as minimum as 5000 dictionary words. System two is implemented on a Multi Layer Perceptron (MLP) and system three uses Convolutional Neural Networks (CNN).
System four is a mere extension of the CNN with a small topical modification. It fine tunes the pre-trained translational model (system 3) using neighbourhood relationships. The systems of each language pair are implemented

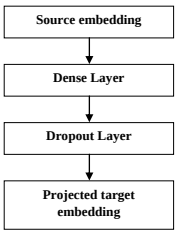

Figure 1: Architecture of MLP for learning the transfer model for cross-lingual embedding

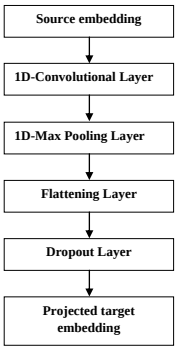

Figure 2: Architecture of CNN for learning the transfer model for cross-lingual embedding

as mentioned above, they are further trained over the monolingual embedding of bilingual word pairs of the respective languages.

2.1. Multi Layer Perceptron

The multi layer perceptrons (MLP) is a fully connected DNN that holds a special place in NLP for intuitive non-linear modeling. Our MLP topology possesses three dense layers, that uses Rectified Linear Unit (ReLU) as its activation. The dropout layer that follows immediate to every dense layer avoids overfitting in training. Cosine proximity is used as the loss function and RMSprop as optimizer. Figure 1 depicts the architecture of MLP.

2.2. 1D- Convolutional Neural Network

The architecture of CNN has five layers, a CNN layer followed by maxpooling, flatten layer, dropout layer and a dense layer. Rectified Linear Unit (ReLU) is used as activation function in each layer. Again, the cosine proximity and RMSprop is used as loss function and optimizer respectively for training. The CNN architecture is shown in Figure 2.

2.3. Fine-tuned Convolutional Neural Network (Fine-tuned CNN)

In this architecture of CNN, the translation model is trained on neighbourhood relationship of source language word pairs given the cosine similarity between the corresponding target language word pairs as labels. The core objective of this network focuses on fine tuning the previously learned translational model to improve on neighbourhood relations.

For training, embeddings of randomly chosen source language word pairs (wv_{s_i}, wv_{s_j}) from the dictionary is given as an inputs to model1 and model2. The model1 and model2 are identical copies of pre-trained translational model discussed in section 2.2. The outputs of model1 and model2, transfer learned/projected target language word vectors $(wv_{t_i^*}, wv_{t_j^*})$, is passed on to the dot layer, that computes the cosine proximity between the vectors. The cosine distance/output of the dot layer is passed on to dropout layer to avoid over fitting and finally passed on to dense layer, where linear activation is used. For back propagation, the cosine distance between the corresponding target language words (w_{t_i}, w_{t_j}) for the source language word (w_{s_i}, w_{s_j}) is given as labels, mean squared error and RMSprop is used as a loss and optimizer respectively. Please note, that, model1 and model2 are already trained and back propagating with the cosine similarity of the word pairs helps in better learning of the neighbourhood relations. The topology of this model is shown in Figure 3

Table 1: Description of Data

Language Pairs	Train (#. of word pairs)	Test (# of word pairs)
de - en	10095	6000
ta - en	21100	1999

3. Data

For "closed track", German (de) and English (en) language pairs, we used the FastText pre-trained embeddings of Wacky corpora (Conneau et al., 2017) and the given bilingual dictionary for training. For "open track", Tamil (ta) and English (en) language pairs, FastText pre-trained embeddings of crawled web corpus (Bojanowski et al., 2017; Pre-trained, 2019) and in-house dictionary is used. Details of the dataset used for the tasks is shown in Table 1

4. Computational Complexity

To induce a word translation for the source word, we perform a reverse look up of the transfer learned target vector with the original target monolingual embedding. Given a set of source and target word $< w_{s_i}, w_{t_i} >$ and their corresponding embeddings (original monolingual embeddings) $< wv_{s_i}, wv_{t_i} >$ and transfer learned target embedding$< wv_{t_i^*} >$. For every query source word w_{s_i}, the correct target word w_{t_i} is identified by locating the target embedding wv_{t_i} that is the closest neighbour to the transfer learned/projected target word embedding $wv_{t_i^*}$, where cosine similarity is computed as a measure between the embedding.

However, performing the reverse lookup is computationally intensive. For instance, the embedding size of each test data (German (de) and Tamil(ta)) is $\in \mathbb{R}^{2000\times300}$ and English pre-trained Wacky and Crawled web corpus is approximately 2 billion words. Henceforth, the size of original

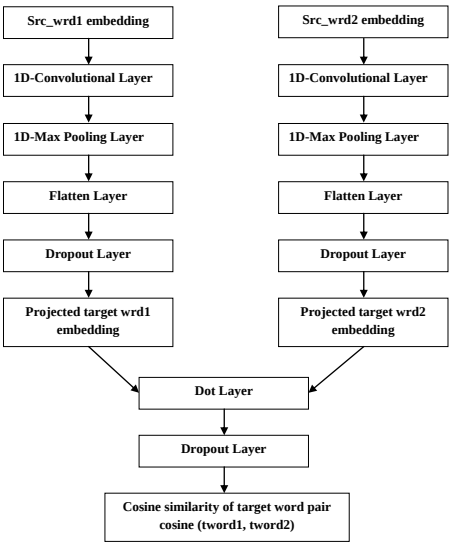

Figure 3: Architecture of CNN for learning the transfer model based on neighbourhood relations for cross-lingual embedding

embedding is $\in \mathbb{R}^{2E9\times300}$. The word vectors are of double data type (8 bytes). The cartesian product of the original embedding and transfer learned test embedding would sum upto size of $\in \mathbb{R}^{4E12\times300}$ (approximately, four trillion). Computing such huge dataset takes months for a normal computer system to compute. This complex computation is deployed to the cluster using Apache Spark® Framework. The word pairs are filtered based on cosine similarity. The figure 4 shows the architecture.

5. Evaluation Tasks

We know that word embeddings translate semantic relationships to spatial distances, in a good word embedding model the semantically related word pairs in a languages are expected to have closer spatial distance (higher similarity score) in their respective embeddings. We use this linguistic aspect to evaluate our cross-lingual word embeddings. Here, we treat the original (monolingually trained) embedding as our ground truth and compare the global neighbourhood behavior of the generated embeddding. Algorithm 1 explains this. The original (monolingual pretrained) and transfer learned embedding are represented as $OrigVec$ and $TransVec$; N represents the size of the test set. The similarity metric between two words vectors a and b is computed using cosine distance as given in Equation 2.

$$cos(a,b) = \frac{a^T b}{||a||.||b||} \tag{2}$$

Algorithm 1: Algorithm for computing percentage accuracy for global neighbourhood behaviour of the transfer learned embeddings

Input: Input:$OrigVec, TransVec$
Output: Output: $Accuracy$

```
k, i ← 0
for i < N do
    for j < N do
        CosOrigVec[k] =
        cos(OrigVec[i], OrigVec[j])
        CosTransVec[k] =
        cos(TransVec[i], TransVec[j])
        k = k + 1
    end for
end for
sum, i ← 0
for i < N * N do
    grad = CosOrigVec[i] − CosTransVec[i]
    tmp = grad * grad
    sum = sum + tmp
end for
RMSE = sqrt(sum/(N * N))
PerErr = (RMSE/2) * 100
Accuracy = 100 − PerErr
```

6. Results

The percentage accuracy of the test data on transfer learned model of each language pairs, German-English (de-en) and Tamil-English (ta-en), tested over various systems is shown in Table 2. In fine-tuned CNN network (CNN+NN), the dictionary is inducted by passing test data to model1 and the output of model1 is calculated for percentage accuracy on global neighbourhood. From the results in Table 2, it is evident that CNN+NN network outperforms the other three models in each language pair. Henceforth, the final result submitted for the shared task is run on CNN+NN network

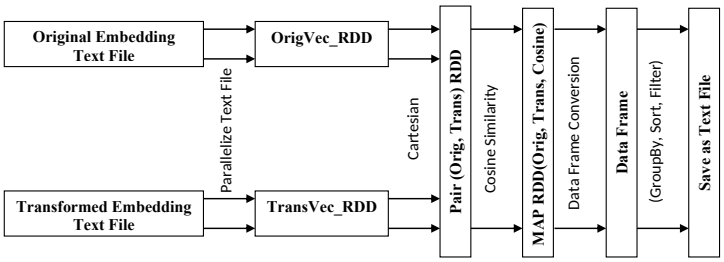

Figure 4: Block diagram for reverse look up of dictionary using Apache Spark® Framework

model.

Table 2: Percentage Accuracy of transfer model of various systems

Models	Language pairs	
	de - en	ta - en
Linear Mapping	73.01	76.05
MLP	80.67	85.52
CNN	85.16	90.33
CNN+NN	***89.91***	***93.65***

7. Conclusion

In this paper, we were able to generate bilingual dictionary for language pairs, German-English (de-en) and Tamil-English (ta-en) by using *'cross-lingual'* embeddings (vectors in separate space, mapped) that is trained on neighbourhood relationship between source language word pairs. As word embedding has no ground truth to evaluate the cross-lingual embedding, we also proposed an evaluation method to validate the model.

For *'de-en'* and *'ta-en'* language pairs, the model is trained with 10095 and 21100 FastText pre-trained monolingual embedding of bilingual words. We started with linear mapping system, as the results were not satisfactory, we moved on to deep learning network. In deep network, CNN gave better accuracy than MLP. Hence, the CNN network was further fine-tuned with a neighbourhood information of source language. This gave the best accuracy among every other systems. Henceforth, test data was run on this system. The core system generates the transfer learned/projected target embedding for the given source embedding. The generated target embedding is compared with the original monolingual target embedding to find the correct target word translation for the source word. To do this reverse lookup process, Apache Spark® Scala language APIs is utilized to manage the computational complexity and speed up.

Bibliographical References

Bansal, M., Gimpel, K., and Livescu, K. (2014). Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Chandar, A. P. S., Lauly, S., Larochelle, H., Khapra, M. M., Ravindran, B., Raykar, V. C., and Saha, A. (2014). An autoencoder approach to learning bilingual word representations. *CoRR*, abs/1402.1454.

Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H. (2017). Word translation without parallel data. *CoRR*, abs/1710.04087.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407.

Gouws, S., Bengio, Y., and Corrado, G. (2015). Bilbowa: Fast bilingual distributed representations without word alignments. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 748–756.

Guo, J., Che, W., Wang, H., and Liu, T. (2014). Revisiting embedding features for simple semi-supervised learning. In *EMNLP*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mikolov, T., Le, Q. V., and Sutskever, I. (2013b). Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013c). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Pre-trained, E. (2019). https://github.com/Kyubyong/wordvectors, June.

Treviso, M. V., Shulby, C. D., and Aluísio, S. M. (2017). Evaluating word embeddings for sentence boundary detection in speech transcripts. *CoRR*, abs/1708.04704.