Bilingual Word Embeddings with Bucketed CNN for Parallel Sentence Extraction

Jeenu Grover
IIT Kharagpur
India - 721302
groverjeenu@gmail.com

Pabitra Mitra
IIT Kharagpur
India - 721302
pabitra@cse.iitkgp.ernet.in

Abstract

We propose a novel model which can be used to align the sentences of two different languages using neural architectures. First, we train our model to get the bilingual word embeddings and then, we create a similarity matrix between the words of the two sentences. Because of different lengths of the sentences involved, we get a matrix of varying dimension. We dynamically pool the similarity matrix into a matrix of fixed dimension and use Convolutional Neural Network (CNN) to classify the sentences as aligned or not. To further improve upon this technique, we bucket the sentence pairs to be classified into different groups and train CNN's separately. Our approach not only solves sentence alignment problem but our model can be regarded as a generic bag-of-words similarity measure for monolingual or bilingual corpora.

1 Introduction

Parallel bilingual corpora are very crucial for building natural language processing systems, including machine translation, word disambiguation, and cross-language information retrieval. Machine translation tasks need a lot of parallel data for training purposes (Brown et al., 1993). Sentence alignment between two parallel monolingual corpora is used for getting the parallel data. But the present sentence alignment algorithms rely mainly on surface based properties of the sentence pairs and lexicon-based approaches (Varga et al., 2007; Moore, 2002; Gale and Church, 1993). A little work has been done using the neural network models for sentence alignment. We propose a novel approach based on neural networks for sentence alignment which performs exceedingly well as compared to standard alignment techniques which can not capture the semantics being conveyed by the text. Our model uses distributed word-embeddings which have been behind the success of many NLP applications in recent years. We then leverage upon the use of CNNs (Zou et al., 2013; Kusner et al., 2015; Norouzi et al., 2013) for capturing the word-overlapping and word-ordering features in similarity matrix for classification.

2 Model

The different aspects of our model are presented below.

2.1 Bilingual word embeddings

Bilingual Word embeddings (Bengio and Corrado, 2015; Luong et al., 2015) are a representation of the words of two languages in the same semantic space. The three ways of getting the bilingual word representations as mentioned by Luong et al. are: bilingual mapping (Mikolov et al., 2013a), monolingual adaptation (Zou et al., 2013) and bilingual training (AP et al., 2014; Pham et al., 2015).

The semantic relatedness of two words across different languages have been compared by using translation dictionaries for the purpose of sentence alignment tasks. But, this method of telling if the two words across languages are synonyms or not, is very discrete, because a word would be called a synonym only if it is present in list of top k synonyms of the other word. Words having a little similarity with each other would be totally ignored. Our approach intends to mitigate this difference by using bilingual word embeddings. Our method is not discrete, because even if the word is not present in the list of top synonyms for the

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics- Student Research Workshop, pages 11–16
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-3003

other, we would still get their similarity (albeit low) using the bilingual embeddings.

In our paper, we would be using bilingual training for getting word representations as proposed by (Luong et al., 2015). Bilingual training does not depend on independently trained word representation on monolingual corpora of either language. Instead, we learn monolingual and bilingual representations jointly on a parallel corpora.

2.2 Similarity Matrix

Two semantically similar sentences in same or different languages would use same or similar words albeit with a change of order introduced by various factors like language grammar, change of narrative, change of tense or even paraphrasing. We assume that if a sentence pair has high word-overlap, then it might be conveying the same semantics. Here, we are ignoring the cases in which same set of words may convey different semantics because parallel corpora contains a few such instances. Handling such instances would lead to minor changes in precision or recall due to fewer instances and is beyond the scope of this paper. Moreover, none of the existing approaches to sentence alignment deal with such cases.

To capture word-overlap, some alignment techniques use measures like TF-IDF similarity (Fung and Cheung, 2004) or even the basic Jaccard similarity along with translation dictionaries. In our approach, we generate a similarity matrix for each sentence pair, where entries in rows are corresponding to the words of the sentence of one language in their order of occurrence. Same way, columns denote the entries for the words of sentence of second language. The entry $S(i, j)$ of S denotes a similarity measure between the words w_i of s_1 and $w_j^{'}$ of s_2. If we find a word in the sentence which is not present in the corresponding vocabulary, we simply omit it. The different similarity measures that we used include cosine similarity and euclidean distance between the embeddings.

2.3 Bucketing and Dynamic Poling

Last step gave us the similarity matrix for the sentences s_1 and s_2, but the size of the matrix is variable owing to different sentence lengths. We would pool this similarity matrix dynamically to convert it into a matrix of fixed dimension. But we were a bit skeptical of this step as even sentences with very short and very long lengths would be pooled to the same dimension. To overcome it, we bucketed the sentences into different sentence length ranges. Bucketing is done on the basis of mean length of the two sentences in the pair to be classified.

Thus, we trained different classifiers for each range of the buckets. The main limitation of this method is that the effective training data reduces for each of the classifiers. This may degrade the performance of the model as compared to a single classifier which has all the data available for training. If parallel annotated data is available in abundance, then this method would work better than a single classifier.

To convert matrix to a fixed size representation, we pool it dynamically to a matrix of fixed dimension as mentioned in Socher et al. We divide each dimension of 2D matrix into dim chunks of $\lfloor \frac{len}{dim} \rfloor$ size, where dim is the bucket size and len is the length of dimension. If the length len of any dimension is lesser than dim, we duplicate the matrix entries along that dimension till len becomes greater than or equal to dim. If there are l left-over entries where $l = len - dim * \lfloor \frac{len}{dim} \rfloor$, we distribute them to the last l chunks. We do it for both the dimensions.

Now, for a particular chunk, we can pool the entries in it using methods like average, minimum and maximum pooling. When we take cosine similarity between words as the matrix entries, we take max-pooling and with euclidean distance, we take min-pooling. This is because we do not want to fade the effect of two words with high similarity entries in the same chunk and the mentioned pooling methods for each similarity measure, take care of it.

2.4 Convolution Neural Network

CNN's have been found to be performing well where we need to capture the spatial properties of the input in the neural network (Krizhevsky et al., 2012; Kim, 2014). The intention behind using CNN's on matrix rather than training a simple neural classifier on input of flattened data is that the similarity matrices not only contain the similarity scores between words but they also capture the word-order. Thus a matching phrase would appear as a high intensity diagonal streak in the similarity matrix (Refer figure 4). A single 1D vector would loose such visual word-ordering features of the similarity matrix. We also report the

performance of a multilayer perceptron classifier as compared to CNN's, justifying our choice.

3 Experiments

3.1 Data

We performed our experiment on the sentence alignment dataset (Zweigenbaum et al., 2016) provided by shared task of 10^{th} BUCC workshop[1]. We performed experiments on the English-German (en-de) dataset. The dataset consists of 399,337 sentences of monolingual English Corpora and 413,869 sentences of monolingual German corpora. The gold alignment has been provided with the data. The gold alignment between the two languages consists of 9580 matching sentence pairs.

3.2 Sampling

As described above, the sizes of monolingual corpora are large as compared to actual aligned sentences. The total possible sentence pairs are 165 billion which contains just 9580 positive sentence pairs, creating a huge class imbalance. It would be difficult to train or even store 165 billion sentence pairs. Moreover, large presence of negative data in the corpora would also make the classifiers less sensitive towards positive data. To overcome these difficulties, we sampled negative examples from the data. Table 1 gives the sizes of datasets after sampling.

3.3 Training of Word Embeddings

We used the English German word embeddings as mentioned in the paper by Luong et al.[2] The dimension of all embeddings used in our experiments is 128. The embeddings are trained on parallel Europarl v7 corpus between German (de) and English (en) (Koehn, 2005), which consists of 1.9M parallel sentences. The training settings as described in Mikolov et al. and Luong et al. were used. The hyper-parameter α is 1 and β is 4 in our experiments as described in Luong et al. The vocabulary size for en is 40,054 and for de is 95,195.

3.4 Similarity Matrix and Bucketing

We split the sentence pairs in data into training, validation and testing set having a ratio of 6:1:3 respectively, giving data splits of size 18654, 3411 and 9581 respectively. The bucketed data ranges for each data split are shown in Table1. There were no sentence pairs in our dataset with mean length of pair below 5 or above 25. More over, the splits for range (5,8] and (20,25] do not contain enough data for training, so we exclude them from the experiments. Thus, out of total 8 buckets in Table 1, we ran the experiments for only 5 buckets as well as all non-bucketed data. We report our results using cosine similarity, as it performed better than using euclidean distance as similarity measure.

B size	Train	Valid	Test	Par	Ratio
[0,5]	0	0	0	4431	0
(5,8]	76	15	43	10281	7.392
(8,10]	1024	189	469	15681	65.302
(10,12]	3561	630	1829	22281	159.822
(12, 15]	8585	1630	4409	34431	249.339
(15,18]	4862	867	2602	49281	98.659
(18,20]	452	78	222	60681	7.449
(20,25]	4	2	7	94431	0.0424
Total	18654	3411	9581	34431	**541.779**

Table 1: Table showing bucket-ranges and the number of sentence pairs in train, validation and test set. Number of parameters (Par) and Ratio of Training data to Parameter size ($Ratio = Train \div Par$, in 10^{-3} units) are shown for each bucket

3.5 Model Parameters

The neural network architecture is described below:

- Convolution Layer 1 with Relu Activation: We used a 3D convolution volume of area 3×3 and depth 12 on input of size $dim \times dim \times 1$ where dim is the bucket size. The strides of 1 unit were used in each direction. The zero padding was done to keep output height and width same as input. The convolution layer was followed by a layer of Rectified Linear Units (Relu) to introduce the non-linearity in the model
- Max-Pooling Layer 1: We used max pooling on the output of the previous layer to reduce its size by half. This was done using strides of 2 units for both height and width.
- Convolution Layer 2 with Relu : It uses a 3D volume of area 3×3 and depth 16. Rest all properties are same as Convolution Layer 1.
- Max-Pooling Layer 2: It again reduces the

[1] https://comparable.limsi.fr/bucc2017/bucc2017-task.html

[2] The code and pre-trained embeddings can be downloaded from http://stanford.edu/~lmthang/bivec

size of input by half using strides of 2 units in both directions. The output of this layer is flattened from 3D to 1D to pass it to next layer as input.

- Fully Connected Layer 1 (FC1) with Relu: This layer maps the flattened input to a hidden layer with 200 units. Relu Activation was used here on the output of hidden units.

- Dropout Layer: We used a layer with dropout probability of 0.2 to prevent over-fitting in the model.

- Full Connected layer 2: This layer maps the output of previous layer with 200 hidden units into a single output, which is further passed to sigmoid activation unit. The value of the sigmoid unit is the predicted output.

The model was trained using 20 epochs with batch-size of 5. The loss function is the mean squared error between actual and predicted output. The Adam optimizer (Kingma and Ba, 2014) was used for stochastic optimization for backpropagation. The learning rate parameter (eta) is 0.0005. The motivation behind using Relu as activation function is to overcome gradient decays, which hinder the training of the neural networks. All the hyper-parameters were tuned by random search in hyper-parameter space and testing on the validation dataset. The total number of parameters to be trained in the model are $150 \times dim^2 + 681$, where dim is the bucket size. This gives us number of parameters ranging from 15,681 for lowest size bucket with $dim = 10$ and 60,681 for largest bucket with $dim = 20$.

4 Results and Analysis

To evaluate, we ran our algorithm on the *en-de Test set* mentioned beforehand. Our algorithm assigns a score to each sentence pair, denoting the probability of two sentences conveying same semantics. If the score is greater than a certain threshold th, we take it as a positive. Table 2 shows results for $th = 0.5$. When we performed experiments for all data (Total in Table 2), without bucketing, we chose $dim = 15$ as highest number of data entries fall in that bucket. We expected that bucketing data would yield better results compared to non-bucketing as each sentence pair would be pooled to a matrix of the dimension comparable to its actual length. But as seen in Table 2, the non-bucketed approach performs very well and in some cases, even better than a few buckets. This happens because the training data available for non-bucketed approach (18654 pairs) is atleast double of any of the buckets. *Ratio* column in Table 1 shows the ratio of Training data to number of parameters in the model, which is highest for non-bucketed data with $dim = 15$. If we had more training data in each bucket, then all the buckets might have achieved better performance than non-bucketed approach. *Macro* and *Micro* in Table 2 denote the Macro-average and Micro-average respectively for all the 5 buckets taken for experiment.

T Data	TP	FP	FN	P	R	F1
(8,10]	128	2	3	.9846	.9771	.9808
(10,12]	472	6	34	.9874	.9328	.9593
(12, 15]	1207	63	13	.9504	**.9893**	.9695
(15,18]	904	4	22	.9956	.9762	.9858
(18,20]	67	0	6	**1.0000**	.9178	.9571
Total*	2825	18	49	.9937	.9830	**.9883**
Macro	-	-	-	.9836	.9586	.9710
Micro	2778	75	78	.9737	.9726	.9731
Baseline	2221	92	653	.9603	.7728	.8564

Table 2: Table showing *Test dataset type (T Data), True Positives (TP), False Positives (FP), False Negatives (FN), Precision (P), Recall (R), and F1-score (F1).*Includes all non-bucketed data*

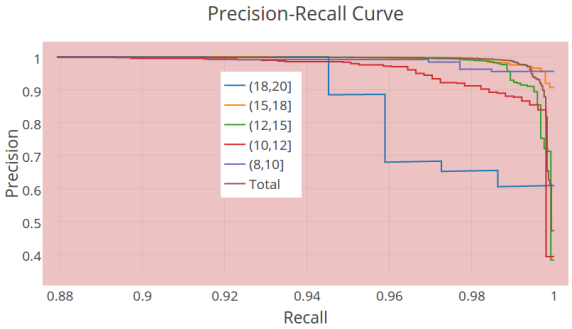

Figure 1: Precision-Recall Curve for all the buckets as well as total data. The different dimensions have been zoomed appropriately to show relevant parts of the plot.

We also used a multilayer perceptron classifier on non-bucketed data with flattened matrices as input (*Baseline* in Table 2), but that performed poorly with F1 score of 0.8564. Figure 2 and 3 depict the similarity matrices for True Positives and True Negatives. We can clearly observe some vi-

sual features of the similarity matrices, such as the presence of high intensity streaks along diagonal, which denote high similarity between the entries in close vicinity in one sentence to the entries in close vicinity in the other. This justifies our hypothesis that, unlike multilayer perceptron, CNN is able to capture the relations between word similarity and their ordering, which is represented by the matrices. Our method also performs much better than other baseline methods such as a classifier using sentence length features.

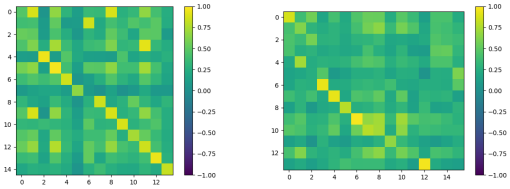

Figure 2: True Positive examples. Left image shows a sentence pair with high overlap and right image shows a sentence pair with low overlap.

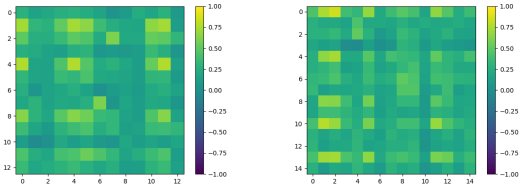

Figure 3: True Negative examples. Both the images are visually different from True Positives.

As an abstraction, our model can be viewed as a neural equivalent of bag-of-words similarity measures such as TF-IDF similarity or Jaccard similarity as this approach covers the word-overlap between two documents (sentences here). Moreover, rather just capturing the overlap, it also captures the order in which the words match in two documents. So, it can be dubbed as neural-bag-of-words like model which remembers matching order.

5 Future Work

We have used the dynamically pooled similarity matrices with CNN for the purpose of sentence alignment. But as already mentioned, our approach specifies a general way of obtaining the similarity between two texts whether they belong to same or different languages. The sentences belonging to the same language can be handled in the same way, but only monolingual word embeddings would suffice for that purpose.

A unique feature of our similarity measure is that we get the similarity between two texts without mapping them to their respective vectors in the vector space. We can interpret it like a kernel function which gives dot product $\phi(x).\phi(y)$ between two entities without actually transforming x or y to $\phi(x)$ or $\phi(y)$ respectively. Also, unlike TF-IDF, where each vector is of the size of the vocabulary, our similarity approach takes only dim^2 space per sentence pair, which is much lesser than the former.

Our current approach assumes that the two texts are of comparable length, because that is generally the case for aligned sentences and that's why we took the dynamically pooled matrices with both dimensions of same size. But, if we want to use our approach for information retrieval purposes, then the size of document D would be much larger than size of the query q. In that case, we would have to take rectangular dynamically pooled matrices with appropriate dimensions. We would like to study the efficacy of our approach in all such scenarios.

Also, since our approach can tell the parts of the document it is matching, unlike TF-IDF, we can use it to assign different scores for matching phrases in different parts of the document. For example, to search for a query on a webpage which has an article and comments from the readers (just like a blog), our approach can be trained to give more importance to the matches in the article as compared to the reader comments, thus leading to a better information retrieval approach. In the future, we would also like to study how different properties like time and space complexity of our approach scale for large dataset. We would also like to explore the applications of our approach for tasks like cross-lingual as well as monolingual information retrieval, query expansion, cross-lingual recommender systems etc.

6 Conclusion

The novelty of our approach lies in using neural word embeddings in bilingual semantic space along with CNN to capture the sentence similarity and we have achieved very good results over the dataset by BUCC. Our model provides a new equivalent of bag-of-words similarity measures which is also aware of the matching order. The

architecture proposed by our algorithm is not just for this specific task but can be used for a number of other tasks like Information Retrieval in monolingual as well as bilingual corpora, query expansion for cross-lingual search etc. We would like to study the different properties and explore the applications of our approach in future.

7 Acknowledgements

We would like to thank all the anonymous reviewers for their valuable feedback.

References

Sarath Chandar AP, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*. pages 1853–1861.

Yoshua Bengio and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments .

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics* 19(2):263–311.

Pascale Fung and Percy Cheung. 2004. Mining very-non-parallel corpora: Parallel sentence and lexicon extraction via bootstrapping and e. In *EMNLP*. Citeseer, pages 57–63.

William A Gale and Kenneth W Church. 1993. A program for aligning sentences in bilingual corpora. *Computational linguistics* 19(1):75–102.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*. volume 5, pages 79–86.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. pages 1097–1105.

Matt J Kusner, Yu Sun, Nicholas I Kolkin, Kilian Q Weinberger, et al. 2015. From word embeddings to document distances. In *ICML*. volume 15, pages 957–966.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. pages 151–159.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Robert C Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Conference of the Association for Machine Translation in the Americas*. Springer, pages 135–144.

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. 2013. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650* .

Hieu Pham, Minh-Thang Luong, and Christopher D Manning. 2015. Learning distributed representations for multilingual text sequences. In *Proceedings of NAACL-HLT*. pages 88–94.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*. volume 24, pages 801–809.

Dániel Varga, Péter Halácsy, András Kornai, Viktor Nagy, László Németh, and Viktor Trón. 2007. Parallel corpora for medium density languages. *AMSTERDAM STUDIES IN THE THEORY AND HISTORY OF LINGUISTIC SCIENCE SERIES 4* 292:247.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*. pages 1393–1398.

Pierre Zweigenbaum, Serge Sharoff, and Reinhard Rapp. 2016. Towards preparation of the second bucc shared task: Detecting parallel sentences in comparable corpora. In *Ninth Workshop on Building and Using Comparable Corpora*. page 38.