# Learning Bilingual Word Embeddings using Word-Alignments

**Rricha Jalota**

March 15, 2021

## 1 Introduction

This project is an amalgamation of two topics that we were taught in our coursework - Word Alignments and Distributional Semantics. The idea is to train bilingual word embeddings using the co-occurrence information obtained from word alignments. I found this topic particularly interesting because I wanted to investigate if the results from word aligners could be used alongside vector space models to learn a meaningful representation of two languages (eg. English and German) in a common vector space. Since English is the most popular language in NLP Research, NLP frameworks and tools are either not available for other (low-resource) languages or if they do exist, they do not work as good. Hence, the task of learning multilingual embeddings provides an opportunity for transfer learning i.e. models trained on one language can also be used across other languages. The goal of this project is to learn bilingual representations in vector space.
To obtain bilingual word-embeddings, I followed three steps: 1) fetched word-alignments using a word-aligner - FastAlign [1]; 2) trained monolingual word embeddings using the word2vec skipgram approach [2] for the source (EN) and target languages (DE); 3) mapped the monolingual word embeddings to a common vector space using the alignment-based projection approach by Guo et al. [3]. All three of these steps have been elucidated in section 3. The bilingual embeddings are then evaluated on two intrinsic tasks: word similarity and analogy. For both of these tasks, custom-built datasets were used (see section 5 for details).

The rest of this paper is structured as follows: Section 2 describes the datasets used for training bilingual word embeddings and the preprocessing steps. Section 3 explains the approach in detail. Section 4 briefs about the experimental settings to reproduce the results. Finally, in section 5 and section 6, the approach is evaluated and discussed, respectively.

## 2 Dataset

To train bilingual word embeddings, first and foremost, a parallel corpora is required. Therefore, I use the EuroParl Corpus [4], which has been extracted from the proceedings of the European Parliament. To be precise, I use the

TMX-formatted EuroParl corpus from OPUS [5] because TMX files only contain unique translation units. This makes it easier to prepare input data for word aligners.
The next step is to clean the TMX-formatted data to To this end, TMX-formatted EuroParl data is first transformed to a tab-separated text file using an open source library, `tmxt`[1]. For the scope of this project, the generated tsv file is first trimmed to 100,000 sentences (using the unix command `head -n100000 <filename> > <out_file>`). Then, all the blank lines, punctuation and digits (non-word characters) are removed and the text is lower-cased[2] Following this, (1) input data is created for fastAlign; and (2) distinct parallel corpora are created for training monolingual word vectors.
Note: After preprocessing, the dataset is reduced to 99,954 sentences. For training fastAlign, this entire dataset of 99,954 sentences is used. However, for learning word embeddings, only the first 5000 sentences (at most) are considered.

## 3 Approach

To train bilingual word embeddings, I follow the monolingual mapping approach that was proposed by Guo et al. [3]. As stated by Ruder et al. in [6], monolingual mapping refers to the set of methods that first independently train monolingual word representations on a large monolingual corpora. Then a transformation function is applied (or learned) to map representations in one language to the representations in another. Such methods use a set of source-target word pairs as anchors for learning the mapping. This method, therefore, has three stages which are described in the following subsections.

### 3.1 Learning Word Alignments

To learn word-alignments from a sentence-aligned parallel corpora, containing sentences from SRC ($S$) to TRG ($T$) language[3], the fastAlign [1] word aligner is used in its default settings. From the obtained the word-alignments, an alignment matrix $A$ containing the co-occurrence count of target-to-source words is created. Thus, the alignment matrix is of shape $(N_T, N_S)$, where $N_T$ and $N_S$ are vocabulary sizes of TRG and SRC languages, respectively. (More details on the alignment matrix are given in subsection 3.3).

### 3.2 Training Monolingual Embeddings using Word2Vec Skipgram Model

For training monolingual embedding spaces, the word2vec variant, skipgram is used. The goal of skipgram is to predict context words within a certain range, given a target word. Figure 1 depicts the architecture of the skipgram model. To be precise, a simple neural network model with a single hidden layer is trained

[1]https://github.com/sortiz/tmxt

[2]I also tried stemming the words during preprocessing (using snowball stemmer) to reduce the size of the vocabularies of the two languages (en-de). However, the results were not as good and the t-sne plots were difficult to interpret.

[3]In my experiments, SRC language is English and TRG language is German.

to perform a certain task. The task here, is to predict the words that occur within a certain range before and after the current word. [2]. As reported by the authors in the original paper, increasing the range improves the quality of the resulting word vectors, but it also increases the computational complexity. Therefore, in all of my experiments, a context window of 3 is chosen.

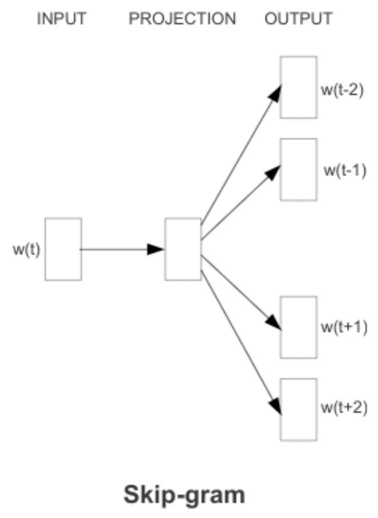
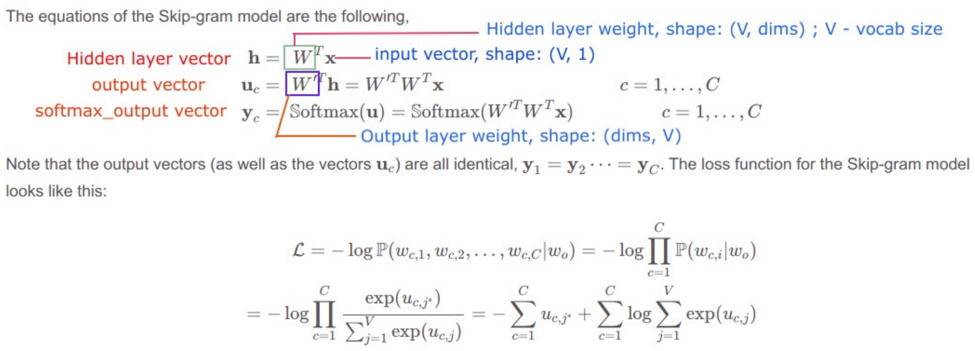


FIGURE 1: Skipgram architecture - Given a current wprd, the goal is to predict context words within a certain range. Source: [2]

### 3.2.1 Skipgram Implementation Steps

The following steps are carried out in the given order to train a skipgram model on a monolingual corpora.

- **Data preparation:** Given that the monolingual data (for both the languages) is already preprocessed (see section 2), the input text is first tokenized to fetch the vocabulary of the given dataset. Optionally, there is also a possibility to exclude those words from the vocab that do not frequently occur in the text. Next, each word is assigned an integer value (word-to-id mapping) based on its position in the word vocab list. Finally, the training data is generated for the skip-gram model.For each target word `word[i]`, training pairs, `(word[i], word[i-window_size]),..,(word[i], word[i-1]),..,(word[i], word[i+window_size])` are created. This implies, for 5000 sentences, if the average sentence length is 10 words, then with a context window (range) of 3, a total of $5000 * 10 * 6 = 300,000$ training pairs would be generated.
- **Model Training:** As a neural network cannot directly read words, each word is mapped to a one-hot encoded vector. Therefore, the input to the neural network is a one-hot encoded matrix of shape $(N, vocab\_size)$


The equations of the Skip-gram model are the following,

$$\mathbf{h} = \boxed{W^T}\mathbf{x}$$

$$\mathbf{u}_c = \boxed{W'^T}\mathbf{h} = W'^T W^T \mathbf{x} \qquad c = 1, \dots, C$$

$$\mathbf{y}_c = \text{Softmax}(\mathbf{u}) = \text{Softmax}(W'^T W^T \mathbf{x}) \qquad c = 1, \dots, C$$

Note that the output vectors (as well as the vectors $\mathbf{u}_c$) are all identical, $\mathbf{y}_1 = \mathbf{y}_2 \cdots = \mathbf{y}_C$. The loss function for the Skip-gram model looks like this:

$$\mathcal{L} = -\log \mathbb{P}(w_{c,1}, w_{c,2}, \dots, w_{c,C} | w_o) = -\log \prod_{c=1}^{C} \mathbb{P}(w_{c,i} | w_o)$$

$$= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j^*})}{\sum_{j=1}^{V} \exp(u_{c,j})} = -\sum_{c=1}^{C} u_{c,j^*} + \sum_{c=1}^{C} \log \sum_{j=1}^{V} \exp(u_{c,j})$$

Hidden layer weight, shape: (V, dims) ; V - vocab size
Hidden layer vector
input vector, shape: (V, 1)
output vector
softmax_output vector
Output layer weight, shape: (dims, V)

FIGURE 2: Equations of the skipgram model. Source.

where $N$ is the number of fed training examples (a.k.a. batch size) and $vocab\_size$ is the size of the vocabulary of the monolingual dataset[4]. The training vector pairs are then passed through a dense single-layer neural network (forward-pass). Error is then calculated as the sum of the difference between $softmaxed\_output\_vector$ and each of the context word vectors. Based on this error, the weights for the hidden and output layer are adjusted via backpropogation. Finally, loss is computed and this process is iterated again for the given number of epochs to minimize the loss. Figure 2 formally shows the equations used to train the model. The word embeddings are basically the weights $W$ of the hidden layer $h$ that are learned during training. For more details, please refer to this blog post, wherein every step has been explained with examples.

## 3.3 Alignment-based Projection

Quoting from [3],

> let $A^{T|S} = (w_i^T, w_j^S, c_{i,j})$ $i = 1, 2, ..., N_T; j = 1, 2, ..., N_S$ be the alignment dictionary, where $c_{i,j}$ is the number of times $i^{th}$ TRG word $w_i^T$ is aligned to the $j^{th}$ SRC word $w_j^S$. $N_S$ and $N_T$ are vocabulary sizes. The shorthand $(i, j) \in A^{T|S}$ is used to denote a word pair in $A^{T|S}$.
>
> The projection can then be formalized as the weighted average of the embeddings of the translation words, i.e.:
>
> $$v(w_i^T) = \sum_{(i,j) \in A^{T|S}} \frac{c_{i,j}}{c_i} \cdot v(w_j^S) \tag{1}$$
>
> where $c_i = \sum_j c_{i,j}$ and $v(w)$ is the embedding for word $w$.

Therefore, we can derive the embeddings for the target words using just the embeddings of the aligned source words. The projection method described in equation (1) suffers from a limitation; it only assigns embeddings to those TRG

[4]first 5000 sentences from monolingual (en/de) EuroParl corpus.

words that have alignments with the SRC words. However, since in my experiments, the dataset used for training word embeddings is much smaller than the one used for training word alignments, not all words in the alignment dictionary are assigned a word embedding and hence, support for words outside the alignment dictionary/mapping is out of scope for this project.

Now, once we have the derived embeddings for the target words, next step is to combine these embeddings with the source embeddings to get a single bilingual representation of the aligned source and target words. However, we are now faced with another problem. How to represent words that are same in both the languages i.e. how to deal with homonyms/heteronyms that are shared across two languages? For example, *rot-de vs rot-en, will-de vs will-en, so-de vs so-en, etc.* In some cases, such word pairs almost have the same meaning, however, they can also mean very different. Like, *rot* in English means 'to deteriorate' while *rot* in German is used for the 'color red'. One simple solution (and the one used) is to average out the embeddings of such word-pairs and get a unique vector representation for every word in the bilingual vocab. The downside of this approach, however, is its inability to capture a meaningful representation of the words that carry the same name but distinct meanings.

## 4 Experiments

For training monolingual word embeddings, two sets of experiments were carried out:

1. on a corpus of 1000 sentences with the following hyperparameters: context window of size 3, 50 epochs, batch_size 500, minimum word frequency 3.

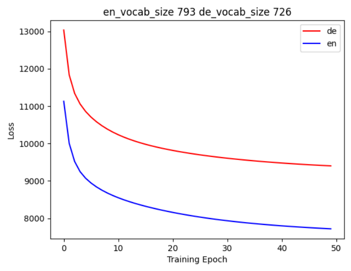
2. on a corpus of 5000 sentences with the following hyperparameters: context window of size 3, 100 epochs, batch_size 1000, minimum word frequency 5.

Both the English and German monolingual embeddings were trained on their respective corporas. The loss vs epoch plots for both the experiments are shown in Figure 3. As can be seen, for the larger dataset, even though the vocabulary for German is larger than that of English, the training loss for German is much lower. Therefore, German word embeddings are used to learn a bilingual vector representation for the aligned English-German words.

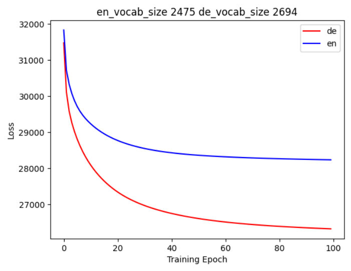
**To retrieve bilingual embeddings, instead of projecting target words to source vector space, as given in equation 1, the opposite is learned. That is, English source words are projected to the target vector space using an alignment matrix $A^{S|T} = (w_i^S, w_j^T, c_{i,j})$ $i = 1, 2, ..., N_S; j = 1, 2, ..., N_T$. Here $c_{i,j}$ is the number of times $i^{th}$ SRC word $w_i^S$ is aligned to the $j^{th}$ TRG word $w_j^T$ and the following equation is used for projection:**

$$v(w_i^S) = \sum_{(i,j) \in A^{S|T}} \frac{c_{i,j}}{c_i} \cdot v(w_j^T) \qquad (2)$$

The obtained bilingual word embeddings are then evaluated.



(A) 1000 sentences, 50 iterations



(B) 5000 sentences, 100 iterations

Figure 3: Loss vs Epoch plots

## 5 Evaluation

### 5.1 Task 1: Semantic Word Similarity

As stated in [7], this task is based on the idea that the distances between words in a vector space can be evaluated via heuristic-based human judgements on the actual semantic distances between the two words. Therefore, a human assesor is given a set of word pairs and is asked to assess the degree of similarity between [0,1] for each pair. The semantic distances between these pairs are also fetched from the vector embedding space, and then these two sets of similarity (or distance) scores are compared based on their correlation with each other. In particular, Spearman's rank correlation coefficient is computed. The higher the coefficient, the better is the quality of the word embeddings.

To carry out this evaluation, a custom dataset of 20 word pairs is built based on the words in the vocabulary of first 5000 sentences of the EuroParl corpora. For this task, evaluation is also done on the `WordSim280` corpus from **Deep Semantic Analogies Dataset** [8]. This file consists of 280 translated word pairs from WordSim353 dataset [9]. The pairs have been re-rated by 10 assessors under the same conditions as in WordSim353. The results are shown in Table 1

| Evaluation Dataset | Weights-corpusSize | OOV Pairs | Pearson Coefficient |
|---|---|---|---|
| WordSim280 | bilingual-5000 | 247 | 0.17 |
| WordSim280 | bilingual-1000 | 279 | - |
| WordSim280 | de-5000 | 260 | 0.26 |
| WordSim280 | en-5000 | 280 | - |
| Custom20 | bilingual-1000 | 15 | 0.57 |
| Custom20 | bilingual-5000 | 1 | -0.14 |

Table 1: Semantic Word Similarity Evaluation

As can be seen from the table, for the bilingual word embeddings that are generated from the dataset of 5000 sentences, a pearson correlation score of -0.14 is achieved on the custom dataset and a score of 0.17 is achieved on the WordSim280 dataset, which means there is only a small correlation between the two sets of similarity scores[5] and thus, the quality of the word embeddings can be significantly improved. Furthermore, due to out-of-vocabulary words, one cannot make a fair comparison among the different trained embeddings. Therefore, even though German embeddings achieve a fairly good correlation score, we cannot say, they are better than bilingual embeddings because out of 280, 260 words are out-of-vocabulary.

However, it is still interesting to see that for such a prototypical implementation, the word embeddings show at least some semantic properties.

### 5.2 Task 2: Analogy Task

For this classic word-embedding evaluation task [2], a small dataset containing 11 analogy relationships ($A : B :: C : D$) is created. This dataset is created based on the vocabulary of the 5000 sentences from English and German corpora such that words on the left side of :: belong to one language (German) and words on the right belong to another (English). For example, `man`:`frau`::`men`:`women`. None of the bilingual word embeddings could perform the analogy task and the score was 0 out of 11.

### 5.3 Visualizations

Finally, all the trained embeddings - both monolingual and bilingual are visualized using t-sne. From the plots, one can see that a few word clusters do make some sense. For brevity, only the t-sne plots for embeddings trained on 5000 sentences are shown in figures 4 5 6. All other t-sne plots (i.e. for embeddings trained on 1000 sentences) are, however, present in the code repository.

## 6 Conclusion and Furture Work

To summarize, in this project, I trained bilingual word embeddings using a sentence-aligned English-German parallel corpora from EuroParl. First, word-alignments from English-German are retrieved via fastAlign and an alignment matrix containing co-occurence count from English to German words is obtained. Then, monolingual word embeddings are trained using Skipgram for the target language (German) using only the first 5000 sentences of the German monolingual corpus. Next, for the source words in the alignment matrix, a projection is learned to the target embedding space using equation 2 and finally, bilingual embeddings are generated. These embeddings are then evaluated on two intrinsic evaluation tasks - semantic word similarity and analogy. The results, as expected, show that there is a huge scope for improvement. The current approach suffers from the limitations of the word alignments, which could be erroneous and the lack of negative examples in the skipgram model training.
In future, it would be interesting to implement skip-gram with negative sampling and then compare its performance to the current results. More directions for future work include learning projection via linear transformation [10] and training a bilingual skip-gram model [11], followed by a thorough evaluation of how each of the approaches fair. If sufficient computation resources are available, it would also be interesting to use a larger corpus and retrieve better quality of

[5]Details on the interpretation of Pearson Correlation Coefficient can be found out here

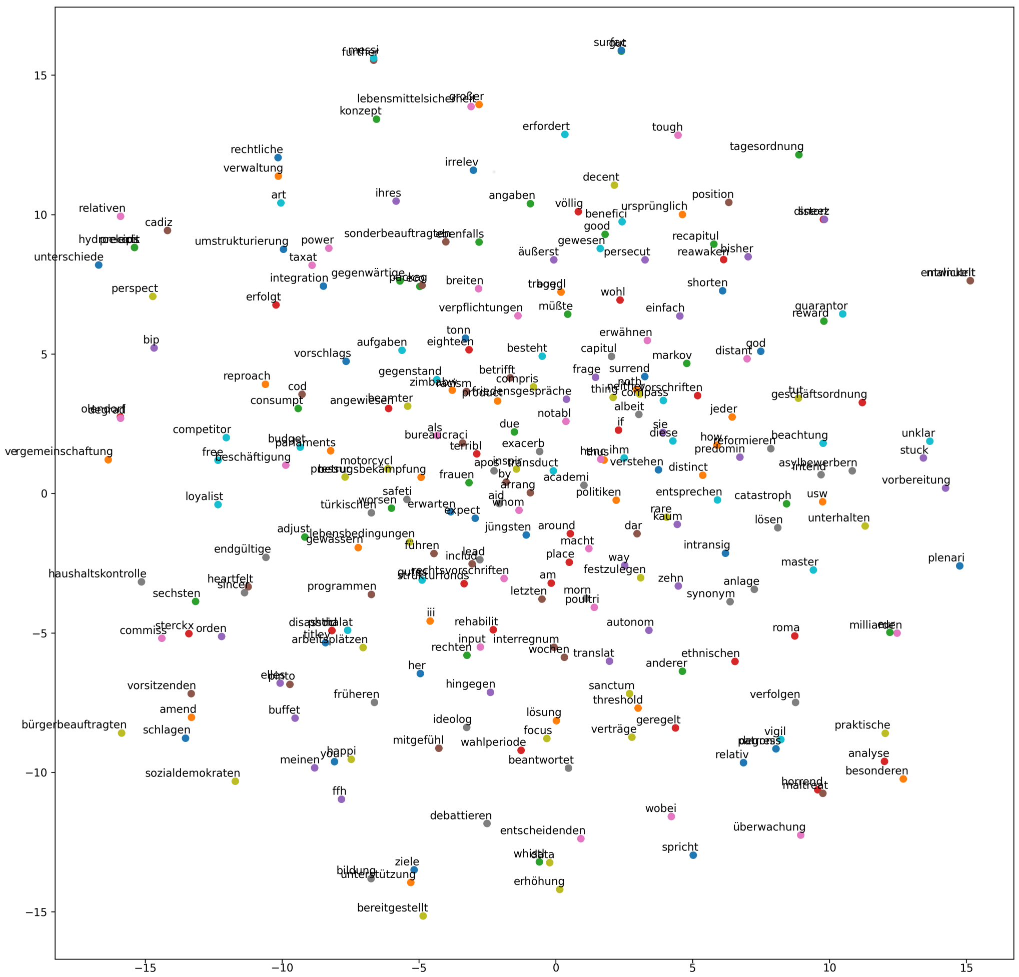


FIGURE 4: Bilingual embeddings

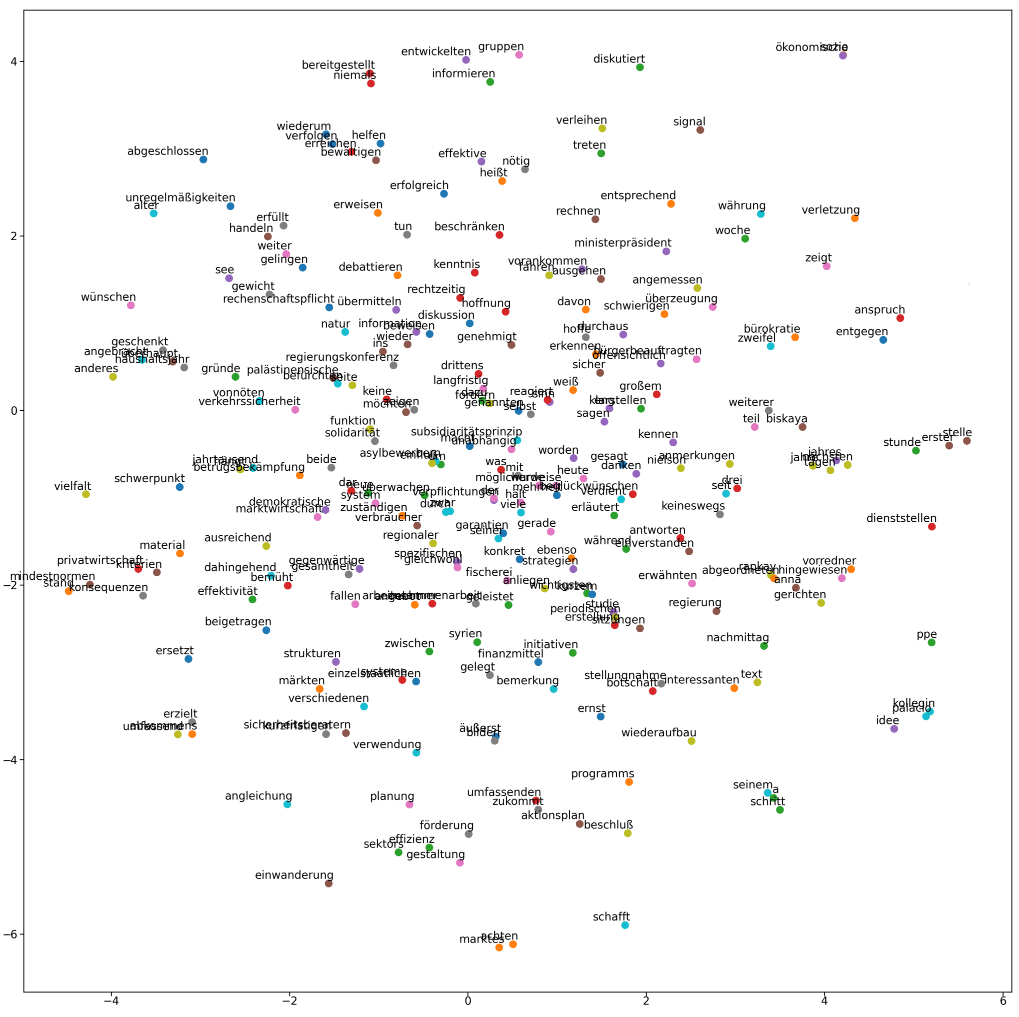


Figure 5: German embeddings

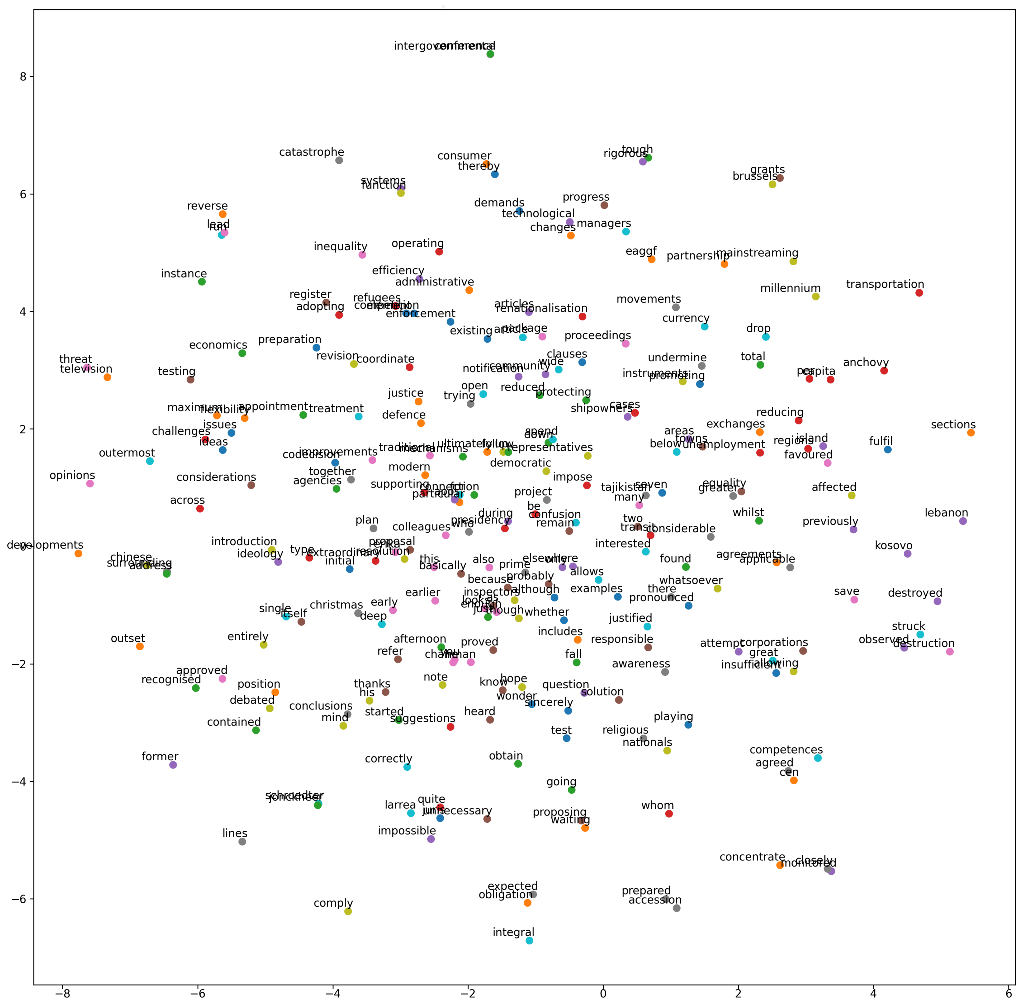


FIGURE 6:   English embeddings

embeddings. Finally, there is a need for more extensive evaluation (perhaps also on downstream tasks like classification) for both the monolingual and bilingual embeddings on a larger evaluation data.

## References

[1] C. Dyer, V. Chahuneau, and N. A. Smith, "A simple, fast, and effective reparameterization of ibm model 2," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 644–648.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[3] J. Guo, W. Che, D. Yarowsky, H. Wang, and T. Liu, "Cross-lingual dependency parsing based on distributed representations," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers.* The Association for Computer Linguistics, 2015, pp. 1234–1244. [Online]. Available: https://doi.org/10.3115/v1/p15-1119

[4] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT summit*, vol. 5. Citeseer, 2005, pp. 79–86.

[5] J. Tiedemann, "Parallel data, tools and interfaces in opus," in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, N. C. C. Chair), K. Choukri, T. Declerck, M. U. Dogan, B. Maegaard, J. Mariani, J. Odijk, and S. Piperidis, Eds. Istanbul, Turkey: European Language Resources Association (ELRA), may 2012.

[6] S. Ruder, I. Vulić, and A. Søgaard, "A survey of cross-lingual word embedding models," *Journal of Artificial Intelligence Research*, vol. 65, pp. 569–631, 2019.

[7] A. Bakarov, "A survey of word embeddings evaluation methods," *arXiv preprint arXiv:1801.09536*, 2018.

[8] S. S. Maximilian Köper, Christian Scheible, "Multilingual reliability and "semantic" structure of continuous word spaces," in *Proceedings of the 11th International Conference on Computational Semantics (IWCS 2015) – Short Papers*, London, UK, 2015.

[9] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 406–414.

[10] T. Mikolov, Q. V. Le, and I. Sutskever, "Exploiting similarities among languages for machine translation," *arXiv preprint arXiv:1309.4168*, 2013.

[11] H. Pham, M.-T. Luong, and C. D. Manning, "Learning distributed representations for multilingual text sequences," in *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, 2015, pp. 88–94.