Comparing Unsupervised Machine Translation Strategies using Word2Vec

II2202, Fall 2015

Bram Leenders
KTH, Royal Institute of Technology
Brinellvägen 8, 114 28 Stockholm
Sweden
b.c.leenders@gmail.com

Marc Romeyn
KTH, Royal Institute of Technology
Brinellvägen 8, 114 28 Stockholm
Sweden
marc.romeyn@gmail.com

Keywords
Word2Vec, Machine Translation

1. WORD2VEC
Word2vec is an algorithm that computes vector representations of words based on co-occurrences. The spatial distance between two word-vectors corresponds to word similarity. In order to achieve this there two different algorithms: skip-gram and continuous bag of words (CBOW). Skip-gram is the most popular choice because it scales better to bigger datasets and therefore also chosen for our research.

Skip-gram model is an method for learning distributed vector representations that capture a large number of syntactic and semantic word relationships [8]. Given a a word w, the skip-gram model predicts the n neighboring words.

Word2vec is an example of so-called "shallow" learning and can be trained as a simple neural network. This neural network has a single hidden layer with no non-linearities and no unsupervised pre-training of layers is needed [9].

Since word2vec is simplified, its scope of application is more limited than deep neural networks [**?**]. However, training is far more efficient due to the lower amount of layers and simpler functions.

2. MULTI-MODEL TRANSLATIONS
In this section, we explain a technique for translating words using seperate models for the input and output language. The techniques described here are originally published by Mikolov et al. [7].

2.1 Translation
Given a word in language A that we want to translate to language B, the first step is to find the vector representation of the word. We do this by looking up the word in a word2vec model trained on language A.

Translating is now a matter of mapping vector representations from model A to corresponding vectors in model B. We do this by multiplying the vector with a translation matrix, which gives an expected vector in model B.

The last step is to convert the "translated" vector representation back to a word. We do this by looking up which word in language B which has the vector representation closest to the translated vector. The criterium used for this is simply nearest neighbour with a Euclidean distance.

2.2 Training the Models
This way of translation requires three models: word2vec models for both language A and B and a translation matrix from A to B.

The two language models are trained using the default word2vec algorithm. We refer to section 1 and previous papers for the specifics on this [6, 8].

The translation matrix is trained by solving the following expression:

$$\underset{T}{\operatorname{argmin}} \Sigma_{i=1}^{n} ||T \cdot a_i - b_i||^2$$

where T is an n by n matrix, a_i and b_i are n dimensional vector representations of words in languages A and B, such that b_i is the translation of a_i.

The quality of the translation largely depends on the accuracy of the mapping from vectors in model A to model B. Training language models using word2vec is unsupervised, and can therefore use hundreds of gigabytes or even terabytes of training data. Training the translation matrix is a supervised process (you have to provide correct translations), which makes it impractical to provide more than a few thousand words.

An advantage of using this method, is that it not only provides a word translation but also gives a distance between the translated vector and its nearest neighbour. If this distance is large, it indicates a higher level of uncertainty in the matrix. As such, one can search for translations where the algorithm is unsure what to choose and provide targeted training data to improve a next iteration of the model.

2.3 Training models of different sizes
One of the parameters when training word2vec models is the number of dimensions to train on. Typically, the number of dimensions is between 100 and 400[6].

Large datasets contain many relations, and can be trained on high dimensions (400 or more), but small datasets (i.e.

under 50 million words) tend to be trained with lower dimensionality. However, these numbers are not set in stone: a study by Pennington et al. [3] indicates that 300 dimensions sufficiently capture relations -for their dataset. Since training complexity scales linearly with the number of dimensions, for very large datasets a high dimensionality might even simply be too costly.

Having discussed the cost/benefit of higher dimensions, we remark that translations with multiple models also allows for models trained with a different number of dimensions. If both models have the same dimensions, the translation matrix will be square. If the dimensions are different, one can either use dimensionality reduction on the larger matrix, or use a non-square translation matrix to scale down the dimensionality.

3. SINGLE-MODEL TRANSLATIONS

3.1 Translation using Relations

As a simplest form of translation, one could say that "king" is to "koning" (the Dutch word for king) as "queen" is to "koninging" (Dutch for queen). Intuitively, this means we consider relations between two languages, instead of relations within a single language.

Thus, if we train a single model with two languages in it, one could expect to find these relations and use them to translate Dutch to English and vice versa.

This method is so simplistic, we do not expect it to give good translations. There is no guarantee the trained model models the Dutch-English relation in a consistent manner. A proper English sentence will not contain the word "koning", since it is not an English word. As such, it will be difficult (if not impossible) for word2vec to learn that "koning" and "king" are related.

Despite our doubts about its effectiveness, we choose to include this method as a baseline for performance and simplicity. Any more complex method performing not at least as good as this is not worth the additional complexity.

3.2 Translation matrix in single space

A mix between the two methods described above is using a single model trained on multiple languages (as in the previous section), but using a translation matrix (as in section 2).

This method would allow the language model to be trained without having to take into account the languages it is being trained on. As such, one could feed it text without determining the language that text is written in. This is a significant advantage for applications where the input is unlabelled.

4. EXPERIMENTS

4.1 Datasets

The datasets used to train the word2vec models are freely available online. We used the wikipedia datasets containing a snapshot of all articles on the English and Dutch Wikipedia and a copy of all Reddit comments.

The Go program used for cleaning the Reddit data is published on GitHub[1]. The wikipedia data was parsed with a slightly modified version of Wikipedia Extractor[2], which is also available on our GitHub page.

The characteristics of the cleaned datasets can be seen in table 4.1.

Name	Items	Words
Reddit comments [3]	1,325,482,268	38,177,224,313
English Wikipedia [4]	4,929,936	1,707,791,444
Dutch Wikipedia [5]	1,831,031	209,095,532

Table 1: Dataset statistics. For Wikipedia, "Items" refers to the number of articles. For Reddit, it refers to the number of comments.

4.2 Tests

We tested each translation method in multiple configurations, most notably with different dimensions for the language models.

The accuracy of the translation is measured in percentages how often the algorithm gave the expected answer. To account for answers that are almost correct, we keep the following statistics:

- **Accuracy @1**: the algorithm gave the correct answer.
- **Accuracy @5**: the correct answer was in the top 5.
- **Accuracy @10**: the correct answer was in the top 10.

If the answer is in the top 5 it is likely to be a synonym or very related word, since word2vec tends to group words by meaning. For example, the translation may be "buy" or "acquire" where the correct translation is "purchase".

5. RESULTS

ADD MORE RESULTS (MATTER OF RUNNING MORE EXPERIMENTS, MOSTLY)

5.1 Multi-model Translations

Figure 1 shows the accuracy of the multi-model translations, both for models trained on 100 and 400 dimensions. We used the same training set of translations to train the matrix on, and both language models are trained on respectively the Dutch and English Wikipedia.

An interesting observation is that the accuracy at all levels (top 1, top 5 and top 10) starts off lower for models trained at 400 dimensions. However, at the biggest training set, each level is better than the corresponding level at 100 dimensions. This suggests a form of overfitting; the translation matrix might be overfitted on the translation samples.

[1] https://github.com/bcleenders/autoTranslate
[2] https://github.com/bwbaugh/wikipedia-extractor
[3] http://academictorrents.com/details/7690f71ea949b868080401c749e878f98de34d3d
[4] https://dumps.wikimedia.org/enwiki/20150901/
[5] https://dumps.wikimedia.org/nlwiki/20150901/

THERE'S ALSO A DIP TOWARDS THE END. CHECK IF THAT'S STILL THERE IF WE RUN IT WITH DIFFERENTLY SPLIT TEST/TRAINING DATA SETS.

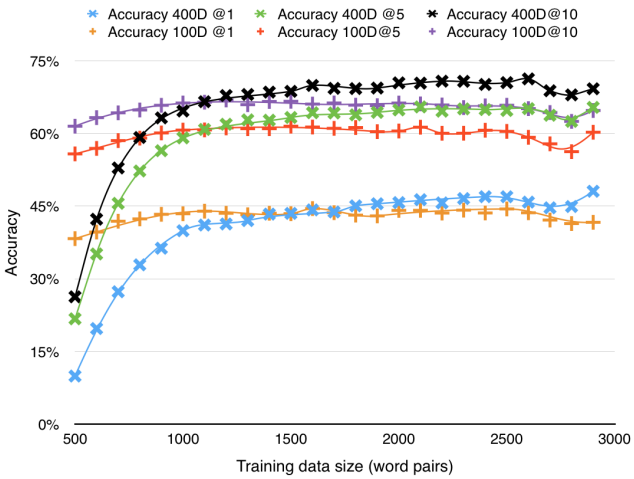

Figure 1: Accuracy for multi-model translations, trained on Dutch and English wiki. 'x' marks denote dimensionality 400, '+' marks dimensionalty 100.

5.2 Single-model Translations

5.2.1 Using Relations

5.2.2 Using Translation Matrix

REPEAT EXPERIMENT FROM MULTI MODEL SECTION, BUT USE BOTHWIKI 100 AND BOTHWIKI 400 AS LANGUAGE MODELS

6. DISCUSSION

6.1 Improvements

- Use labelled data (e.g. nl_koning, en_king). Optionally: use words that are the same (computer, Amsterdam, wc) to get "free" training data. One can suppose these are likely to be translations.
- Use word2phrase to combine "San Fransisco" into a single word.
- Handle ambiguous translations better (i.e. "bank" in Dutch can be either "couch" or "bank" in English). More research can be done in the exact implementation for this.

7. CONCLUSION

METHOD X IS BEST. REFER TO SECTION ?? FOR FURTHER IMPROVEMENTS.

Acknowledgements

The authors thank SICS Swedish ICT for the resources they provided.

8. REFERENCES

[1] N. Boycheva. Distributional similarity music recommendations versus spotify: A comparison based on user evaluation. 2015.

[2] C. Buck, K. Heafield, and B. van Ooyen. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, 2014.

[3] R. JeffreyPennington and C. Manning. Glove: Global vectors for word representation. 2014.

[4] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[5] O. Levy, Y. Goldberg, and I. Ramat-Gan. Linguistic regularities in sparse and explicit word representations. *CoNLL-2014*, page 171, 2014.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

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

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[9] H. Wang. Introduction to word2vec and its application to find predominant word senses. 2014.

[10] L. Wolf, Y. Hanani, K. Bar, and N. Dershowitz. Joint word2vec networks for bilingual semantic representations. *International Journal of Computational Linguistics and Applications*, 5(1):27–44, 2014.