Comparing Unsupervised Machine Translation Strategies with 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. INTRODUCTION
The Internet offers a vast amount of natural language that can be used in natural language processing, for a very low price. A study by Buck et al. [2] estimated that each of the top 10 most frequently used languages on the internet has at least 250 GiB worth of text publically available online. For English (the most frequent), they even found 23 TiB of text.

Such amounts of text have a big potential to be used for the training of language models, but only if building these models can be done in an unsupervised fashion. Manually curating, marking and tagging text is far too much work to be feasible. With unsupervised algorithms, however, the structure in language can be exploited to let computers build language models.

This research will focus on unsupervised training of computer models to translate words. Specifically, we focus on the use of word2vec [5] to provide translations.

WE COULD EXPAND A BIT HERE, AND REVISIT AFTER WRITING MORE OF THE OTHER PARTS

1.1 Background and Related Work
Since the introduction of word2vec [5, 7] in 2013, the algorithm has seen a wide variety of usecases. In the initial paper [5], Mikolov et. al describe interesting relations between vectors corresponding to words. A famous example of how word2vec models relations between words as mathematical equations is $king - man + woman = queen$. The sematic relationships between man/woman and king/queen are preserved in the transformation of words to vectors, and can be expressed with basic algebra.

Subsequent papers have improved the algorithm both in terms of accuracy ([4]), performance, parallelization and extended the initial scope of applications. A good example of the latter is a paper by Boycheva [1], which uses word2vec outside the natural language processing (NLP) domain but to generate playlists. Based on a set of playlists, their word2vec-based algorithm can suggest new playlists with artists that go well together.

One of the applications of word2vec inside the NLP domain, is exploiting similarities in languages for assistance in machine translation [8]. Mikolov et al. [6] released a subsequent paper on word2vec, in which they describe similarities between models of different languages. An example they give, is how the usage of the numbers one to five in English is very similar to the usage in Spanish, and likewise for the names of animals. Figure 1 shows a graphical representation of word vectors in English and Spanish.

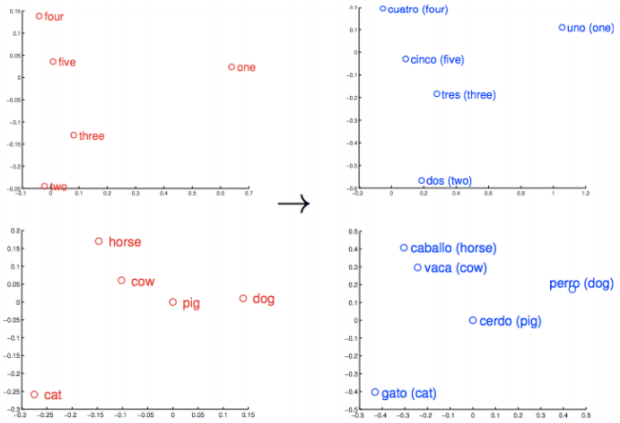

Figure 1: Vector representations of English and Spanish words, after dimensionality reduction and rotation. Notice the high level of similarity between both languages. Reprinted from Mikolov et al. [6]

The similarities between languages can be used to predict translations for words without any human interaction or labeled input data. Using only unsupervised machine learning techniques, a computer could learn how to translate English to for instance Spanish and vice versa. The only requirement is a large amount of text in both languages to train the word2vec models on.

In this research, we will focus on this specific application of word2vec: using similarities in languages to provide translations of words.

It is important to note that word2vec only uses information

of co-occurrences to model words. It does not learn grammatical concepts other than by statistical analysis. This limits our translation to single words; although the translator might be able to translate each word individually, it cannot learn that each finite verb must has a subject, that "we" is plural, etc. It will learn that "swim" is to "swimming" as "walk" is to "walking", but will not know that "swimming" is a gerund. Note that word2vec can be extended to sentences or whole documents as proposed by Le et al. [3] but this will be out of scope for our research.

1.2 Our Contribution

This research aims to improve the capabilities of machines to learn translations with minimal human intervention. Previous research [6, 8] has already shown potential for word2vec in the context of automatic translation, as discussed in section 1.1.

However, we found no practical implementations using Word2Vec and no further research on different setups for word2vec based machine translation.

1.3 Outline

BRIEFLY EXPLAIN STRUCTURE OF PAPER

2. RELATIONS BETWEEN LANGUAGES

EXPLAIN HOW KING IS TO KONING AS QUEEN IS TO KONINGIN

:) is to :(as happy is to sad

... etc.

3. 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. [6].

3.1 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.

3.2 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 previous papers for the specifics on this [5, 7].

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 potential bad translations and provide targeted training data to improve a next iteration of the model.

4. SINGLE-MODEL TRANSLATIONS

EXPLAIN HOW YOU TRAIN A SINGLE MODEL AND LOOK FOR LANGUAGE PATTERNS WITHIN THAT SINGLE MODEL

5. EXPERIMENTS

5.1 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 5.1.

Name	Language	Items	Words
Reddit comments [3]	English	1,325,482,268	38,177,224,313
English Wikipedia [4]	English	4,929,936	1,707,791,444
Dutch Wikipedia [5]	Dutch	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.

[1] `https://github.com/bcleenders/autoTranslate`
[2] `https://github.com/bwbaugh/wikipedia-extractor`

5.2 Tests

MARC: EXPLAIN HOW WE EVALUATE THE OUTPUT

6. RESULTS

NO RESULTS YET

What did our experiments tell us?

Acknowledgements

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

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

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

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

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

[7] 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.

[8] 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.

[3] `http://academictorrents.com/details/7690f71ea949b868080401c749e878f98de34d3d`

[4] `https://dumps.wikimedia.org/enwiki/20150901/`

[5] `https://dumps.wikimedia.org/nlwiki/20150901/`