Fast Training of word2vec Representations Using N-gram Corpora
Filip Ginter, Jenna Kanerva
Department of Information Technology
University of Turku, Finland
ginter@cs.utu.fi, jmnybl@utu.fi

Abstract
In this paper, we study methods to train the popular word2vec vector space representation of the lexicon using only n-gram collections. By using the n-grams rather than the full text corpus we gain a substantial speed-up in training, as well as get the opportunity to train from corpora which would not be available otherwise.

1. Introduction
Among the notable recent developments in the research on continuous vector space representations of the lexicon is the introduction of the word2vec method of Mikolov et al. (2013a). The word2vec method is sufficiently efficient to be trained on corpora in the billion-token range and several models are publicly available that have been trained on corpora in the hundreds of billion token category.

However, the vast majority of researchers in the academia are unlikely to have access to a hundreds of billion word corpus and not necessarily the computational resources to train word2vec models on corpora this large either. The availability of such corpora is also often prevented by copyright protection. Fortunately, it is often legally possible to release n-gram collections from these otherwise closed text corpora. For example, in 2006 Google released the first n-gram collection (LDC2006T13) and in 2009 and 2012 they released collections of n-grams by publication year from their book corpus. For Finnish a tri-gram collection was recently released, based on a 5 billion word corpus from journals and other periodicals dating back to 1820.

In this work, our primary objective is to investigate whether and how these n-gram collections can be utilized to induce a vector space representation of the lexicon using the word2vec method. Besides being able to learn models based on corpora we otherwise would not have access to, we will also be interested in whether models of matching performance in down-stream applications could be trained more efficiently, which would in turn allow us to experiment with various training techniques more freely.

2. word2vec skip-gram architecture
The word2vec skip-gram method is a simplified neural network model trained by sliding a context window along the text and learning word representations by training the network to predict a single context word from the focus word at the middle of the sliding window. The network is trained using back-propagation and therefore its learning rate parameter \( \alpha \) as well as its gradual decrease over time need to be set appropriately. The choice to use the skip-gram training of word2vec was made because it lends itself very naturally to our purposes, as it only requires to be given a single focus-context word pair at a time and does not need the whole context to be available at once.

3. Training from n-gram collections
A single n-gram can be viewed as the left-hand half of a context of its right-most word, and the right-hand half of a context of its left-most word. Even though we irrecoverably lose the access to the complete context window, as we will meet the left-hand and right-hand halves as separate n-grams, this is in fact fully compatible with the skip-gram word2vec induction method, which considers only a single context word at a time. The skip-gram training focus-context word pairs for a 5-gram \( w_1 \ldots w_5 \) would thus be the eight pairs \((w_1, w_2), \ldots, (w_1, w_5), (w_5, w_1), \ldots, (w_5, w_4)\). One can also use the fact that for any pair \((w_i, w_j)\), a sliding window method would ultimately also visit \((w_j, w_i)\), and therefore, we can also extract the additional eight pairs \((w_2, w_3), \ldots, (w_3, w_5), (w_5, w_2), \ldots, (w_5, w_4)\), thus totaling 16 training examples from a single 5-gram.

The most direct interpretation of the count \( C \) associated with each n-gram would be to repeat the forward and back-propagation steps of the network \( C \) times, which would be prohibitively inefficient. One could also use the count \( C \) to adjust the learning rate \( \alpha \) of the back-propagation algorithm on a per-ngram basis, such that the smaller the count \( C \), the smaller the update of the weights becomes for pairs from this particular n-gram. And finally, we could simply ignore the n-gram counts and treat all n-grams as equal.

In what we think is one of the more surprising outcomes of this work, we could not find a way to implement the learning rate adjustment to give better results than simply ignoring the counts, which is the strategy we will use in the evaluation as well. We note, however, that the word2vec method downsamples common words, which to some extent achieves the same goal of decreasing the impact of extremely common words on training.

With or without the count-adjusted learning rate, the training then proceeds in the same manner as the word2vec skip-gram model, where the pairs of words extracted from each n-gram are used to train the network as if they were extracted from running text.

4. Evaluation
We evaluate the models on three different tasks on Finnish and English. The Finnish corpus comprises of 1.5B tokens extracted from the CommonCrawl Internet crawl dataset, totaling 264M unique 5-grams (Kanerva et al.,
Table 1: F$_1$-scores for the word similarity and semantic role labeling tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Finnish</th>
<th></th>
<th>English</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>base</td>
<td>n-gram</td>
<td>base1</td>
<td>base2</td>
</tr>
<tr>
<td>Wordsim</td>
<td>22.95</td>
<td>19.28</td>
<td>45.72</td>
<td>75.71</td>
</tr>
<tr>
<td>SRL</td>
<td>63.81</td>
<td>66.29</td>
<td>66.5</td>
<td>64.83</td>
</tr>
</tbody>
</table>

Table 2: Top-5 accuracy of Finnish→English (first number) and English→Finnish (second number) translation.

<table>
<thead>
<tr>
<th></th>
<th>Finnish</th>
<th></th>
<th>English</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>base1</td>
<td>n-gram</td>
<td>base2</td>
<td>n-gram</td>
</tr>
<tr>
<td>FIN-base</td>
<td>41.2-34.4</td>
<td>46.6-44.9</td>
<td>41.2-33.8</td>
<td></td>
</tr>
<tr>
<td>FIN-n-gram</td>
<td>40.5-39.0</td>
<td>48.7-48.2</td>
<td>42.2-37.4</td>
<td></td>
</tr>
</tbody>
</table>

Overall, we find — where comparable scores exist — that the n-gram based models are on a par with the full text baselines, while for the word similarity task, the baseline English models give a vastly superior performance. This we however think is likely due to the different underlying corpora (Google Books for n-grams versus English News and Wikipedia for the baselines). The possibility for substantially different scores even among models trained on full text is well demonstrated by the 30pp difference between the two English baselines. Further experiments are thus needed for English to establish comparable scores.

For the translation task, it is interesting to note that the top-5 accuracy results are well in line with those previously reported in (Mikolov et al., 2013b), who show top-5 accuracy ranging from 42% for the Czech→English pair to 52% for the Spanish→English pair. With models only trained on the n-gram collections, we obtain 42.2% for Finnish→English and 37.4% for the English→Finnish pair.

One of the primary advantages of training the models on n-gram collections is their relative compactness. This is especially prominent on the Google Books n-gram corpus. With a context span of ±4 words, the training on the n-gram corpus consists of 9.1B word pairs, while in the full text the corresponding count is a staggering 3.7T pairs. This amounts to over 400× speed-up in the training time, which is the difference between several hours and several months.

In practical terms, even the largest English n-gram models could be trained in 12 hours on a single computer.

5. Conclusions and future work

Overall, we find — where comparable scores exist — that the performance of the models trained on n-grams approaches models trained on full text. Considering the enormous reduction in training time, especially with the large English corpora where the training is two orders of magnitude faster when carried out on n-grams, we see this as a viable technique that will allow further experimentation and parameter optimization which would be otherwise prohibitive due to the computational costs involved.

As the future work (some which has already been carried out but did not fit into the extended abstract page limit), we will expand the experiments to include also syntactically informed word2vec models, utilizing the recently released syntactic n-gram corpora for English and Finnish (Goldberg and Orwant, 2013; Kanerva et al., 2014). The evaluation on English will be extended to provide fully comparable scores based on the same underlying corpus.

All source code necessary for replicating our results and training new models will be made publicly available at https://github.com/fginter/gensim.

Acknowledgments

This work has been supported by the Emil Aaltonen Foundation and Kone Foundation. Computational resources were provided by CSC – IT Center for Science. We would also like to thank Radim Řehůřek for making the gensim implementation available.
References


