Mining Big Data - Final Report

Investigating Paragraph Vectors

a1632538 Zachary Forman
a1646930 James Caddy

October 30, 2016

Introduction

We live in a world where enormous amounts of textual data exists, and even more is generated each day. This data presents both a major opportunity and major challenges, being plentiful and rich in information, but also high dimensional and loosely structured. We can view text as consisting of words, and can describe any collection of words as a document. In the past, textual documents are frequently viewed as just a collection of words, the so-called “Bag of Words” model [1]. Each word in such a model is typically just a number - the semantic connotations of words and documents discarded for the sake of efficiency.
In 2013, Mikolov *et al.* presented the `word2vec` algorithm [2]. This approach allowed for each word to be represented by a vector that contains a compact representation of the semantic meaning of that word. In 2014, Le and Mikolov extended this approach [3] with the `doc2vec` algorithm to allow learning vectors that summarize the semantic meaning of a document.
We seek to extend Le and Mikolov's work, reproducing their `doc2vec` results and further extending them to the domains of document similarity and understanding source code.

Contributions

This work makes the following contributions:

- Replication of a relatively recent paper [3] that makes some contentious claims [4].
- Comparison of `doc2vec`'s suitability as a document similarity measure as compared to Jaccard similarity.
- Application of the `doc2vec` algorithm to a novel domain – source code.

Background

Word Vectors

The first meaningful attempt at learning vector representations was Rumelhart *et al.* 's 1986 contribution [5]. This work, while relatively primitive, paved the way for more sophisticated models, such as Bengio *et al.* 's 2003 work [6]. This work presents a model that is extremely similar to the current state of the art: Learn vectors for words based on errors seen while trying to predict surrounding context.
Mikolov *et al.* present various enhancements to this technique [7, 2], improving both the speed of training the word vectors and the quality of the learned representations. The algorithm presented in these papers is known as `word2vec`. Mikolov *et al.* show that these learnt word vectors not only retain semantic information, but also relational information; for example, the result of $\text{vec(king)} - \text{vec(man)} + \text{vec(woman)} \approx \text{vec(queen)}$.

Paragraph Vectors

Document Similarity

Data Collection

Data Sources

We used three sources of data. Firstly, we use Maas' IMDB set [8] for verification and English language documents which can be found at `http://ai.stanford.edu/~amaas/data/sentiment/`. Secondly, we used The Wikimedia Foundation's English Wikipedia data dump [9] as a source for English language documents. We downloaded the full wiki dump from `https://`

`dumps.wikimedia.org/enwiki/20160820/`. Lastly, we used The Chromium Project's Chromium repository [10] as a source for source code documents. We cloned the repository at revision 4fe31bb06cf458234d7017950a8b2b82427487c8.

Data Scale

The IMDB dataset contains a total of 100,000 files. Half are labelled as either a positive or negative reviews, and the remainder are unlabelled. It is 346MB in size and contains 23 million words, from a vocabulary of 90 thousand words.
The English Wikipedia Dataset is a single XML file, 13GB compressed and 55GB uncompressed. It contains 16 million pages, however this includes redirects and meta-pages such as categories and portals, and therefore only 5 million of these are useful as documents. Excluding markup, this dataset contains approximately 4 billion words.
The Chromium codebase is 2.6GB in size, and contains 245 thousand files. Of these, 23 thousand are C and C++ files, which contain 4.8 million lines of code and 608 thousand lines of comments.

Data Format

The fundamental requirements for the `doc2vec` [3] algorithm to work are that each document can be associated with the words in the document, and that words in each document are just a series of words, i.e. no excess punctuation or markup.
This requirement can easily be met by storing our set of documents as a newline separated list of space separated tokens. The following example shows how a simple document set might be transformed into this file format:

{"My cow was sick, but now it is better.", "Apples are nice.", "Amazing results!"}

Convert all text to lowercase and remove all punctuation:

{"my cow was sick but now it is better", "apples are nice", "amazing results"}

Concatenate documents, separated by newlines:

```
my cow was sick but now it is better
apples are nice
amazing results
```

This format is simple to produce, simple to parse, and - given that the `doc2vec` algorithm needs to process every word anyway - is efficient to read. Perhaps more importantly, especially for large datasets like the Chromium dataset, this storage format has no storage requirements in excess of just the raw text in the files.

Data Processing

To transform the data into the format previously described, some data processing is required. All programs that we used to transform the datasets are available at `https://github.com/mbd-doc2vec-team/mbd-doc2vec/tree/master/data-processing`.

IMDB Dataset

The IMDB dataset needed minimal processing. The bulk of the work had already been performed by the authors of the dataset. Beyond what had already been done, punctuation, symbols and elements of markup were removed, and each document concatenated into a file as discussed in the previous section.

Wikipedia Dataset

The Wikipedia dataset required extensive pre-processing, with each document needing to be parsed from the XML, and the plain text extracted from the wiki markup. Additionally, "fake" documents (e.g. portals, category pages) were removed. Each wiki page was then tokenized and stored in a file as previously stated.

Chromium Dataset

The Chromium codebase required extensive pre-processing as well, with each C++ file being tokenized and having unicode characters (e.g. ¿) removed. Unlike the other two datasets, because punctuation and uppercase/lowercase distinctions are semantically significant in code, we retained punctuation and the original casing in tokens. Each comment and string literal was then tokenized as well, with punctuation being removed from these tokens only, and the resulting tokens being stored as discussed in the previous section. For an example of this process, the document

```
// Frobulates the bar, in an efficient manner.
void Frobulate(Bar &bar);
```

would be represented by a line in the file as follows

```
Frobulates the bar in an efficient manner void Frobulate ( Bar & bar ) ;
```

Experiments

IMDB Dataset

The experimentation was undertaken with the method as follows.

Preprocessing

- The entire set of 100,000 documents were processed as described in the previous sections. The reason for concatenating all documents into a single file is that the dataset is small and can comfortably fit within memory, so it does not suffer from this simplification.

Training

- The co-author, Mikolov, provided some guidance and recommened the use of gensim's skip-gram model, which is trained for 20 iterations on the entire corpus of reviews.
- After each iteration, the order of the documents is randomised.

Evaluation

- The training document vectors are paired with their labels and a logistic regression is calculated to fit.
- Prediction of sentiment using the testing document vectors takes place and the accuracy of the prediction versus the actual labels is reported.

Wikipedia Dataset

Chromium Dataset

Results

IMDB Dataset

Through experimentation and switching from a neural network based approach to a logistic regression, the final best accuracy achieved whilst attempting to replicate the results was 89.4%.

Being unable to replicate the results of Le and Mikolov in [3], some research into the discussion was conducted. Many others were trying their hardest to replicate these results, but ultimately the co-author Mikolov, who was not involved in the experimentation found the issue. This was reported in [11], where Mikolov states the 92.6% accuracy is achievable using hierarchical softmax "only when the training and test data are not shuffled. Thus, we consider this result to be invalid".

Wikipedia Dataset

Chromium Dataset

Conclusion

Future Work

References

[1] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. "Understanding bag-of-words model: a statistical framework". In: *International Journal of Machine Learning and Cybernetics* 1.1-4 (2010), pp. 43–52.

[2] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[3] Quoc V Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents." In: *ICML*. Vol. 14. 2014, pp. 1188–1196.

[4] *Google Groups*. URL: `https://groups.google.com/d/msg/word2vec-toolkit/q49firnoqro/bp--14e4unwj`.

[5] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Cognitive modeling* 5.3 (1988), p. 1.

[6] Yoshua Bengio et al. "A neural probabilistic language model". In: *journal of machine learning research* 3.Feb (2003), pp. 1137–1155.

[7] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[8] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[9] Meta. *Data dump torrents — Meta, discussion about Wikimedia projects*. [Online; accessed 9-August-2016]. 2016. URL: https://meta.wikimedia.org/w/index.php?title=Data_dump_torrents.

[10] *Git repositories on chromium*. URL: https://chromium.googlesource.com/.

[11] Grégoire Mesnil et al. "Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews". In: *arXiv preprint arXiv:1412.5335* (2014).