Mining Big Data - Intermediate Report

Investigating Paragraph Vectors

a1632538 Zachary Forman
a1646930 James Caddy

September 11, 2016

Introduction

This document describes our progress towards achieving our project goals. We have three primary project goals, the first and second of which our current work has focused on:

1. Replicate the results claimed by Le and Mikolov in their 2014 paper [1].
2. Compare `doc2vec` backed with k-nearest-neighbours with standard approaches to determining document similarity [2].
3. Apply `doc2vec` to source code, and explore how it distributes code segments.

Current Progress

Work on the IMDB dataset

Some exploratory analysis and testing, using gensim [3] as an implementation of `doc2vec`, and the tests have been run as per the paper [1]. So far the results are disappointing. Despite several attempts to replicate the paper's experiment, and obtain a working sentiment classifier, so far, we have not come close to achieving the accuracy described in the paper, or achieved by other approaches. So far, only an accuracy of 68% has been obtained, in contrast with the 92% claimed by Le and Mikolov.

Work on the Wikipedia dataset

We again used gensim [3] to run `doc2vec` on the wikipedia corpus [4]. Even with only 2 training iterations on the corpus (compared with a desired 10-20), this took over 100 CPU hours. From the word and paragraph vectors trained on this data, we qualitatively evaluated the algorithm's performance on three axes:

1. Did the learned word vectors capture semantic meaning well?
2. Did the learned document vectors capture semantic meaning well?
3. Could the document vectors be used for similarity analysis?

Current Experimental Progress

All source code necessary to reproduce our results is available at `https://github.com/mbd-doc2vec-team/mbd-doc2vec/`, the IMDB dataset is available as referenced in Maas *et al.* 's work [5], and the wikipedia dataset is available at [4].

Experiments on the IMDB Dataset

The approach taken followed the method described in the paper as best as possible. Gensim [3] was used for the `doc2vec` implementation, with parameters as in Table 1 and Keras [6] backed with Theano was chosen for the neural network library, since neither were specified in the paper. It is unlikely that either of these choices were the cause for low performance, and more likely that our implementation was flawed. Both of these libraries were relatively easy to use which was good.
While the paper gives clear descriptions of most components and parameters in the classifier architecture, critical details are overlooked and explained poorly. Replicating the results without clear descriptions or values falls to trial and error, where any number of confounding variables could be in effect. One such ambiguity is the dimension of the document vectors, which can have large effects on accuracy.

Experiments on the Wikipedia Dataset

Similarly to the IMDB dataset, we trained `doc2vec` on the Wikipedia dataset. Due to the sheer size of the dataset, this was a lengthy process, with a single iteration taking in excess of 50 hours of CPU time. As a result (and due to numerous issues with disk space and memory consumption leading to crashes during model serialization) we only trained for 2 iterations of the Wikipedia dataset. We used the parameters in Table 1 for training. Once we had trained word and document vectors, we performed some qualitative analysis.

First, we compared the distribution of vectors in the trained Wikipedia model and the trained IMDB model. We did this by finding the vectors closest to a given seed word. Generally the wikipedia dataset, despite the smaller number of iterations, had better performance, as can be seen in Tables 2 and 3. This can likely be attributed to the broader text corpus offered by Wikipedia, with concepts such as Apple as a company as opposed to apple as a fruit being unavailable in the movie-focused IMDB source.

Secondly, we evaluated the distribution of vectors internally by finding the nearest neighbors to some specific documents. An example of this can be seen in Table 4, where we find that the closest documents to the wikipedia page for `xkcd` include other web-comics 'MS Paint Adventures' and 'Penny Arcade'.

We then performed some preliminary experiments to see how relevant `word2vec` can be for determining document similarity. We found that the similarity of the learned vectors is robust under changes to the document - even with 80% of the tokens in a document removed, the similarity between the modified document and the original was over 0.8, as can be seen in Table 5.

Finally, we plotted the word vectors for some subsets of articles (since there are far too many to plot them all) using t-SNE [7]. A selection of these plots can be seen in the appendices (Figures 1-4). The salient feature of these plots was that the encapsulated conceptual relationships that have meaning to humans - for example, in Figure 2, it is clear that related concepts are tightly clustered. This reflects the algorithm's success in learning document level concepts.

Future Work

IMDB Dataset

Further clarification of the architecture would improve the performance far better than continued tuning. Unless there is any insight disclosed in discussion around the paper, contacting the author of the paper seems like the only way this might be achieved. Which is unfortunate, given the author has been so far unwilling to help others in discussion groups. In order to obtain higher accuracy, there is the possibility that running the training for longer could help. Both the embedding and the classifier could benefit from this, however over-fitting the data is a concern. Currently, achieving an accuracy within 10% of that claimed by Le and Mikolov [1] seems unlikely, and as such, serves as a goal for our next experiments.

Wikipedia Dataset

The most pressing, and most time-consuming, future work for the Wikipedia dataset is to train the model for more generations. While the word vectors and document vectors were somewhat coherent, there was clearly noise caused by under-fitting. While training for 20 generations will take 1000 CPU hours, the benefits of training for a significant number of iterations should be worthwhile.

After this, we will compare the similarity metric that can be obtained using `doc2vec` with those obtained from the W-shingles method, particularly with intent to quantitatively compare the speed of finding similar documents, and qualitatively assess how well the similarity metric compares to a human notion of similarity.

Chromium Dataset

We seek to apply the `doc2vec` algorithm to learn 'code' vectors, describing both comments and code attached to it. We aim to discover if there is any human-recognizable significance to the learnt vectors. Given the success shown in the the visualization of the Wikipedia dataset, hopefully we can produce graphics that reveal what code elements are related.

References

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

[2] Andrei Z Broder. "On the resemblance and containment of documents". In: *Compression and Complexity of Sequences 1997. Proceedings*. IEEE. 1997, pp. 21–29.

[3] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. `http://is.muni.cz/publication/884893/en`. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[4] 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`.

[5] 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`.

[6] *Keras: Deep Learning library for Theano and TensorFlow*. URL: `https://keras.io/`.

[7] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

Appendices

Parameter	IMBD model	Wikipedia model
Vector size	200	200
Context distance	5	8
Training iterations	20	2

Table 1: Parameters used for `doc2vec` for each experiment

Wikipedia model	IMBD model
apricot	kahuna
cherry	mines
onion	interpreters
mango	explosive
persimmon	imperialist
indigo	catholic
hazelnut	conservatory
pomegranate	pilling
yahoo	craftsmanship
ibm	casing

Table 2: Words similar to 'apple'

Wikipedia model	IMBD model
screwball	screwball
slapstick	slapstick
erotic	romance
modernist	stand-up
realist	slap-stick
satirical	physical
absurdist	teen
bromantic	sophisticated
improvisational	erotic
surreal	frothy

Table 3: Words similar to 'romantic'

Page name	Cosine similarity
randall munroe	0.517820239067
ms paint adventures	0.443829506636
time (xkcd)	0.440876543522
penny arcade	0.424748897552
alexey pajitnov	0.424251556396
ctrl+alt+del (webcomic)	0.423852056265

Table 4: Wikipedia pages most similar to XKCD

Percentage of words remaining	Cosine similarity of document vectors
100	1.0
90	0.972760783379
80	0.938948015357
70	0.920902002202
60	0.906302172798
50	0.884556154362
40	0.853949113559
30	0.83972823229
20	0.814944547022
10	0.73780806255

Table 5: Cosine similarity of a document with a copy of itself with some proportion of tokens randomly removed

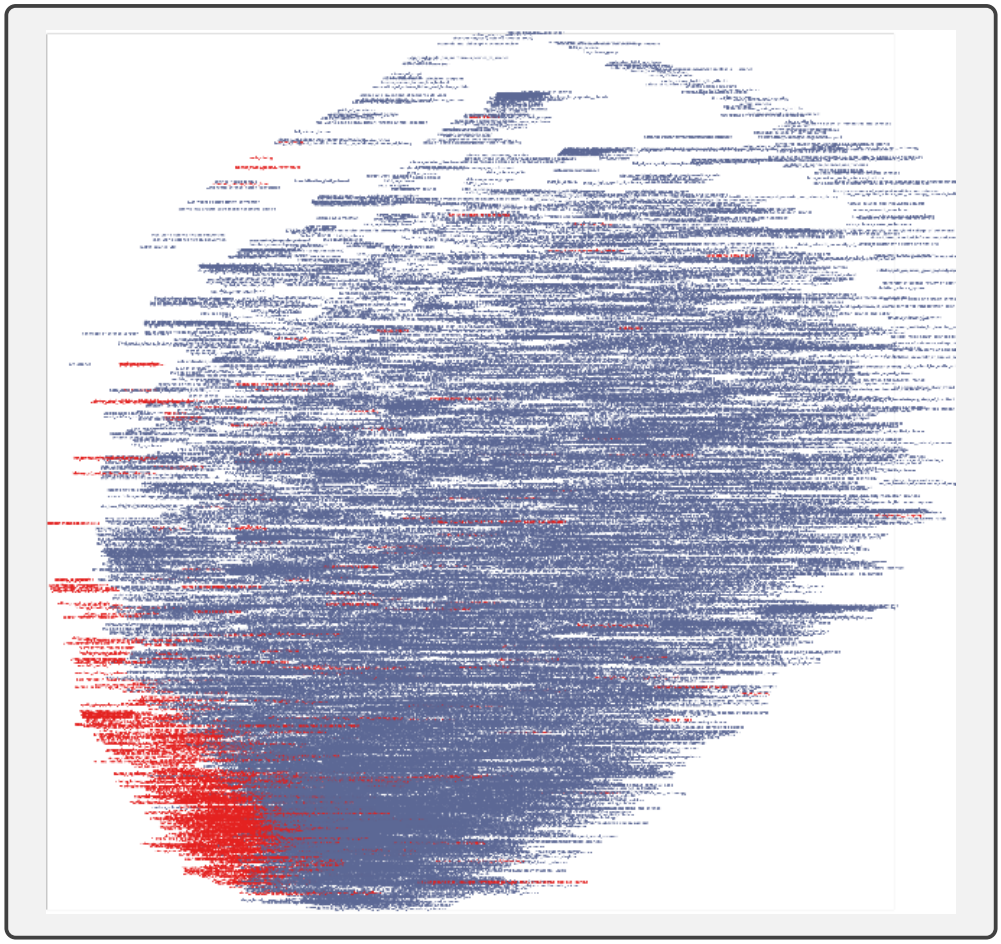


Figure 1: t-sne [7] used to visualize mining (red) and science (blue) wikipedia pages. Concepts such as 'School of Geological Sciences' are nearer the meeting point of the groups.

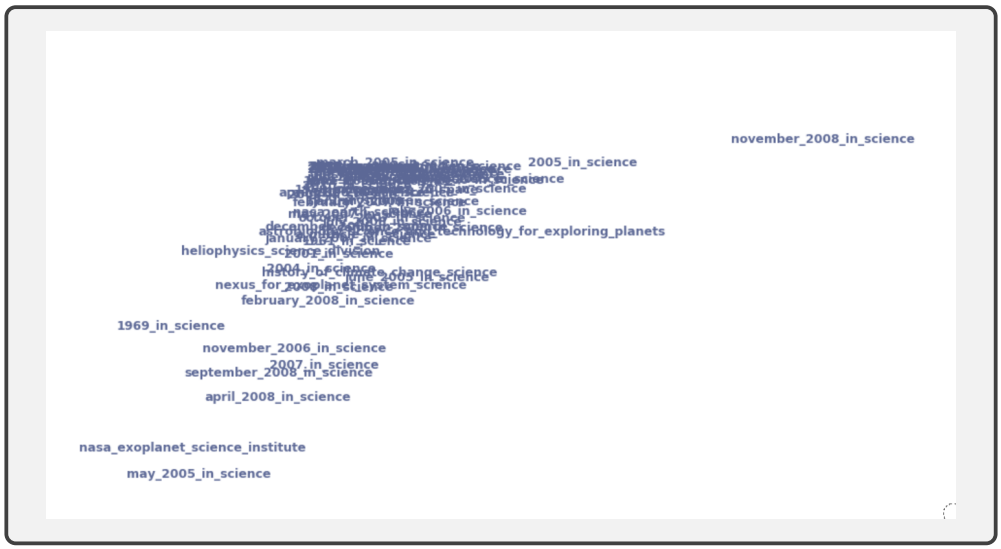


Figure 2: Some conceptual clusters from Figure 1.

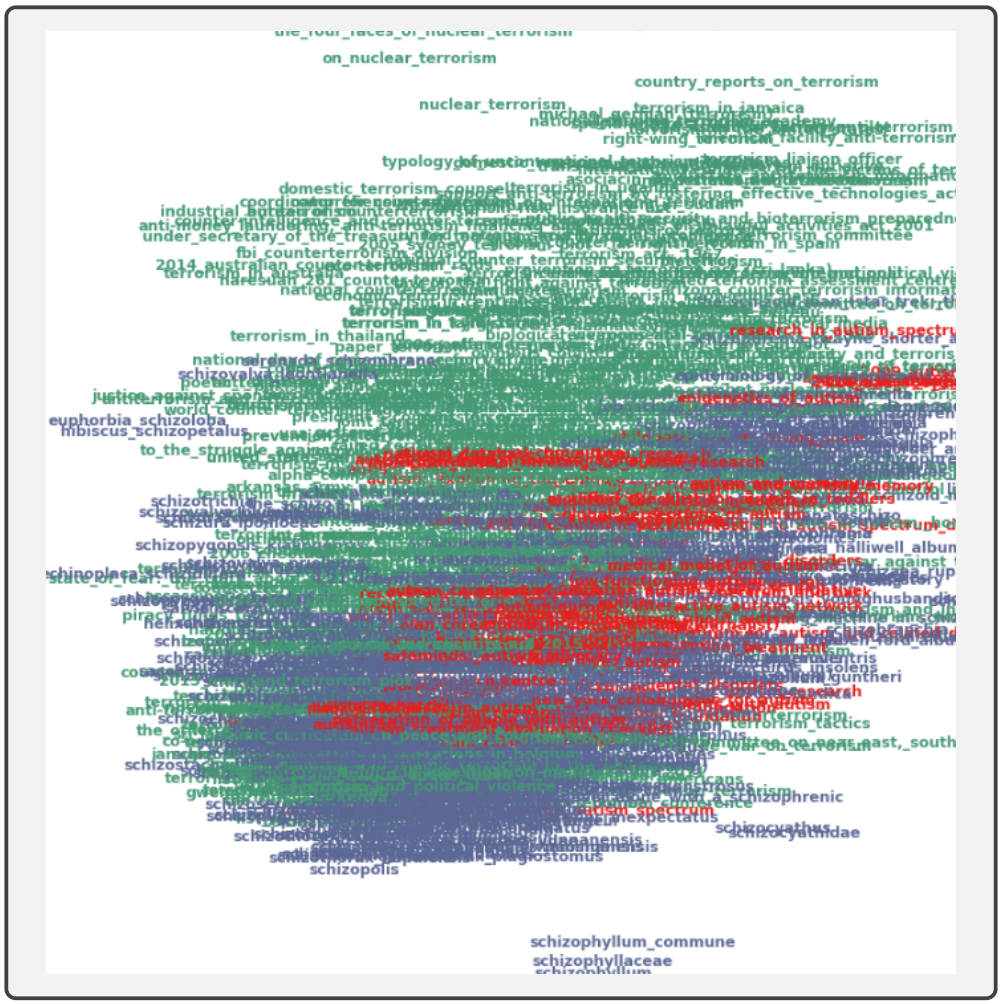
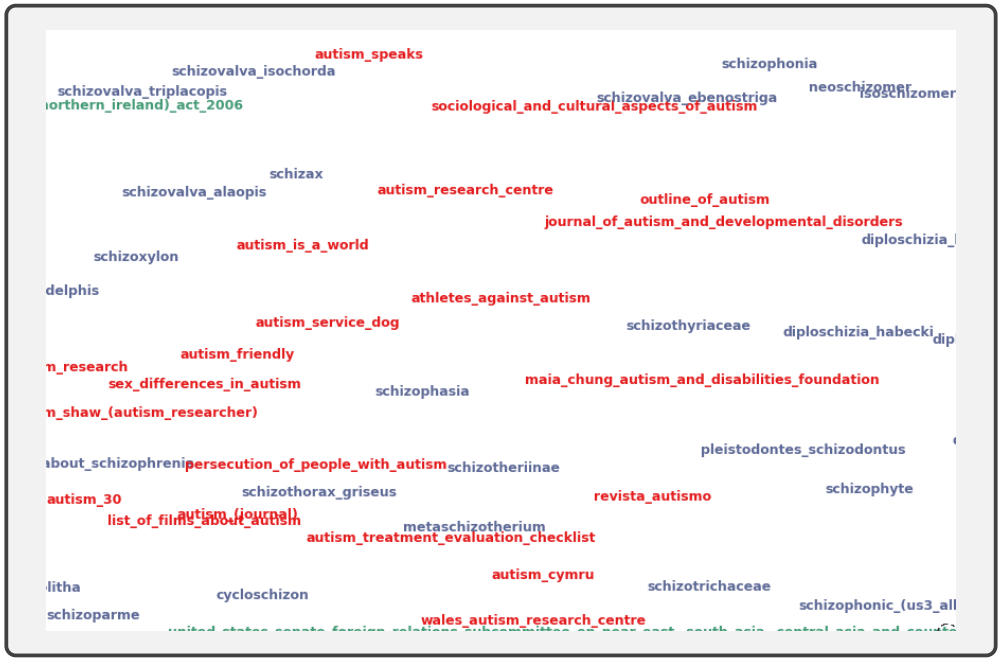


Figure 3: t-sne [7] used to visualize autism (red), terrorism (blue) and schizo.* (blue) Wikipedia pages. Note how autism and schizophrenia are clustered together, in the top right.



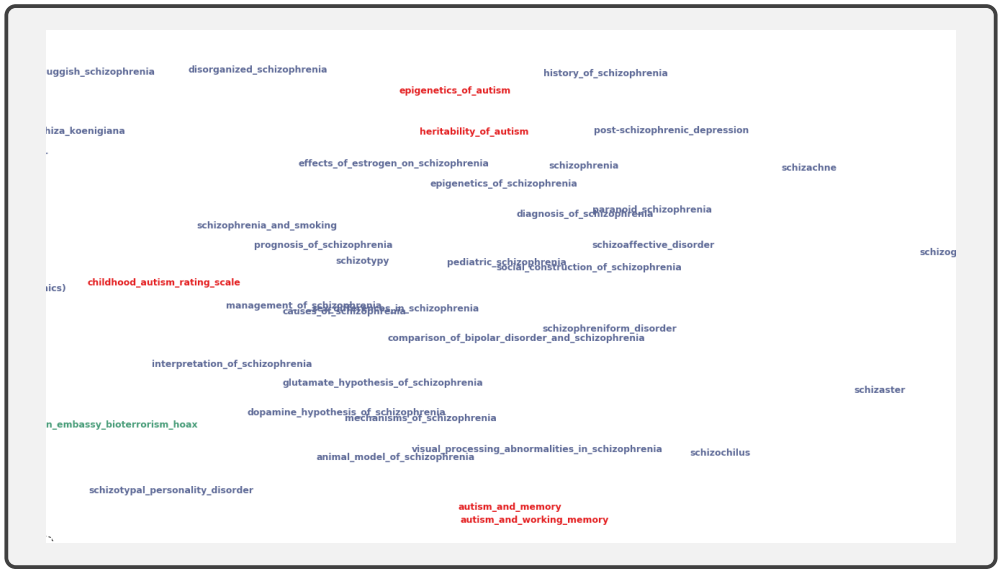


Figure 4: Some conceptual clusters from Figure 3.