Document Similarity using Feed Forward Neural Networks
CS224D Final Project Writeup

Jackson Poulos
Stanford University
jpoulos@stanford.edu

Leonard Bronner
Stanford University
lbronner@stanford.edu

Abstract

Throughout this course we have learnt a lot about similarity and differences between words and phrases. However, for many natural language processing tasks this is not informative enough, often it is more useful and important to find similarities and differences between entire documents. Thus, for our final project, we decided that we wanted to use deep learning to compare documents and decide whether they were related to one another or not. When deciding how to approach this task, it quickly became clear that the difference between two documents is more than the sum of differences between their word vector representations. Having learnt about Siamese Neural Networks during the midterm, and having done more research into their usefulness, we decided our best option would be to use existing neural network architecture to build our own Siamese Network. We would then run through different types of document vectors, in order to see whether we could learn anything meaningful. After a lot of trial and error, our final idea was, given two documents, we could compare them using what our network had learnt, and the two documents would be classified as related or unrelated. Having tried a number of combinations of similarity metrics and document vector representations, we strayed from the siamese model. Instead, we moved a step back to a normal feed forward network, trying out different combinations of non-linear layers. We have come to the conclusion, that neural networks are in fact very good at distinguishing similar documents, and are able to learn the difference between related and unrelated ones - they easily contend with non-deep learning methods.

1 Introduction

1.1 Problem

As we learnt in this class, word vectors are great ways to learn semantic similarities and differences betweens words. Word vectors can then be learnt in combination (or seperately) from different tasks such as sentiment analysis or named entity recognition. While these problems are incredibly useful and interesting, they often focused on shorter pieces (such as words, phrases or sentences) **and** they also only evaluated one document at a time. However, information does not come in isolation. A different problem from the ones we tackled in class, is how to compare two documents. While word vectors allows us to compare two words, and sentences can be compared through the sum of differences in word vectors, we thought it may be interesting to see whether it would be possible to generalize this problem. Given two random documents, we wanted to be able to learn whether these documents are related to one another or not.

1.2 Approach

We thought the easist approach would be to use news articles as documents to classify for a number of different reasons. Practically speaking, news articles are often not too long, which meant that similarity and differences can easily be observed. Concurrently, we had access to a hand labeled data of news articles. Furthermore, we understand that relations between documents is extremely important in the news world. Many content producers, and content distributers of news are looking for ways to enhance their recommendation engine for articles and so we can see an active utility to our work in this class in a field that interests both of us. In order to gage whether to documents were in fact related to eath other or not, we decided to use the subjective idea of relevancy. We wanted to model a subjsect matters expert in the field of politics for matching source articles to commentaries on that article.

2 Background Related Work

In order to first move away from word vectors, we first wanted to train our document vectors the same way as was standard (Le & Mikolov, 2014) using sentence and paragraph vectors. However, we quickly found this model to be to complex to learn for this task (we wish to continue this work over the summer to see if we can generate any meaningful progress there) and so we decided to move to back to mean word vectors over entire documents. This is standard for context for context prediction. Thus, our approach to document vectors was "using a weighted average of all the word s in teh document' as suggested in the Le & Mikolov paper. And which has been shown to be succesful in other areas and experiments (Zanzotto et al. 2010). One thing to hnote is that we weight each word equally, it is possible that weighting words differently would improve results.

In order to move forward with thet type of neural network to learn our similarity and different metrics from we looked to other uses of Siemese Neural Networks. We saw there had been some meaningful progress in that area when it came image comparison (Chopra et al. 2005, Bromley et al. 1993) and so we saw there being potential to generalize this for natural language processing too.

3 Approach

3.1 Document Vector Representation

As stated above, we used mean word vectors as our document vector representations. Initially learnt from a corpus of 50000 political news articles. Our document vectors had dimension 400. This means that our word vector space is entirely focused on news, in fact on US national politics. Therefore, it might make for intersting futher research to look at the document vectors in a larger, more general space. However, we hypothesize that it will be even more difficult to distinguish differences in documents if we tried this.

The documents came from aproximately 4300 news articles which were scraped from the internet. For each orginal document, we then had 10 pieces of commentary that were either relevant or not-relevant (as annotated by a subject matter expert in the field of politics. We wanted to use this annotation and our word vectors to find differences and similarities between the articles through our neural network.

3.2 Siamese Neural Network

Here, we were heavily inspired by the Siamese Neural Network. This means that our initial setup looked lie this:
The document input vectors $x_1, x_2 \in \mathbb{R}^{400}$, with shared parameters $W \in \mathbb{R}^{1000\times 400}$ and $b \in \mathbb{R}^{1000}$. We had a single input layer with each input:

$$h_1 = \sigma(Wx_1 + b)$$

$$h_2 = \sigma(Wx_2 + b)$$

To evalualte the distance between the two activations h_1 and h_2, we used experimented with both Cosine Similarity and Euclidean distance as a simiarity metric. We used Euclidean distance because it was used in the midterm, however, intuitively we hypothesized that cosine similarity may be better due to the difference in length of the documents (words repeat more often in longer documents and so the magnitude mean word vectors - document vectors - could be distorted).
Thus, we had:

$$J_1 = \frac{1}{2}||h_1 - h_2||^2$$

or

$$J_2 = \frac{h_1 \dot{h}_2}{||v_1||||v_2||}$$

We will not write out all the math here (with derivatives and all), but just to show we used the following derivatives for our similarity metric:

$$J_1 : 2 * (h_1 - h_2)$$

$$J_2 : \frac{h_2||h_1|| * ||h_2|| - \frac{(h_1 \dot{h}_2) h_1 * ||v_2||}{||v_1||}}{||v_1||^2 * ||v_2||^2} s$$

Through this neural network, we wanted to learn be able to identify articles that are relevant to each other through the similarity and/or difference of ther representation that had gone through a non-linear input layer. We could see this further being extended with multiple non-linear layers to test whether this improves accuracy.

3.3 Feed Forward Neural Network

However, as mentioned above, after a lot of experimentation with Siamese Neural Networks, using a cosine similarity metric, we found the results to be lower than we wanted. Thus we took a step back and decided to try an "easier" and less technologically advanced approach. Instead we took the difference of the document vectors we produced and ran them through the following feed forward neural network:
$v = v_1 - v_2 \in \mathbb{R}^{400}$, $W_1 \in \mathbb{R}^{200 \times 400}$, $b_1 \in \mathbb{R}^{200}$, $W_2 \in \mathbb{R}^{2 \times 200}$, $b_2 \in \mathbb{R}^2$:

$$inp = W_1 v + b_1$$

$$h_1 = tanh(W_2 inp + b_2)$$

$$outp = softmax(h_1)$$

We also tried running it with a second hidden layer:

$$h_2 = \sigma(W_3 h_1 + b_3)$$

We experimented with many structures and nonlinearities, and the above gave the consistently best performance.

3.4 Implementation

We made use of PyBrain to create and train our neural networks, with the addition of some custom modules. See the supplementary material for examples.

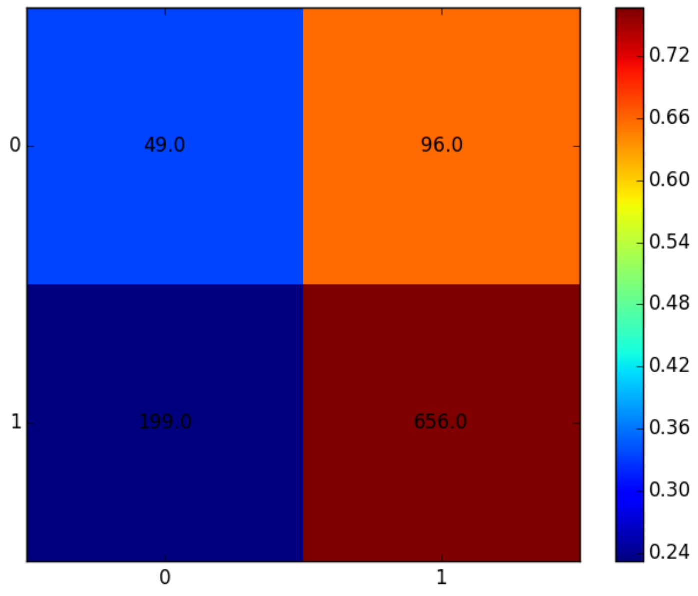
4 Experiment

The experiment is actually a simple binary classififcation task. We want to classify a pair or documents as either relevant to each other or not. This means that the non-linearity is used to see non-linear relationships among the input vectors.
As stated above, the dataset is over 4000 news articles and labeled commentaries to these articles that are either relevant or non-relevant to the initial article. Initially, the articles had been matched through a recommendation engine using a tdidf score, powered by a random forest, and was then hand labeled by as actually relevant or not by a political expert.
We split the data into a train/test split of 75/25.
As this experiment is a binary classification and we have labeled data, we are going to use area under the receiver operating characteristic (roc) curve. We could also use F1 score, however, we are more interested in minimizing false positives and don't care much about false negatives (recall in F1). This is because the used case for our model is to generate a list of articles that are related to our current article. Thus, if our model misses a few, it is less of a problem than if it thinks that two unrelated articles are in fact related. However, for internal measures we will also generate a F1 score to see how we compare to other classifcation tasks. Currently, the random forest model has an auc of 0.87 on the ROC curve, so this is the number that our model needs to beat for us to say that we have suceeded in our experiment.

4.1 Results

4.1.1 Siamese Neural Network

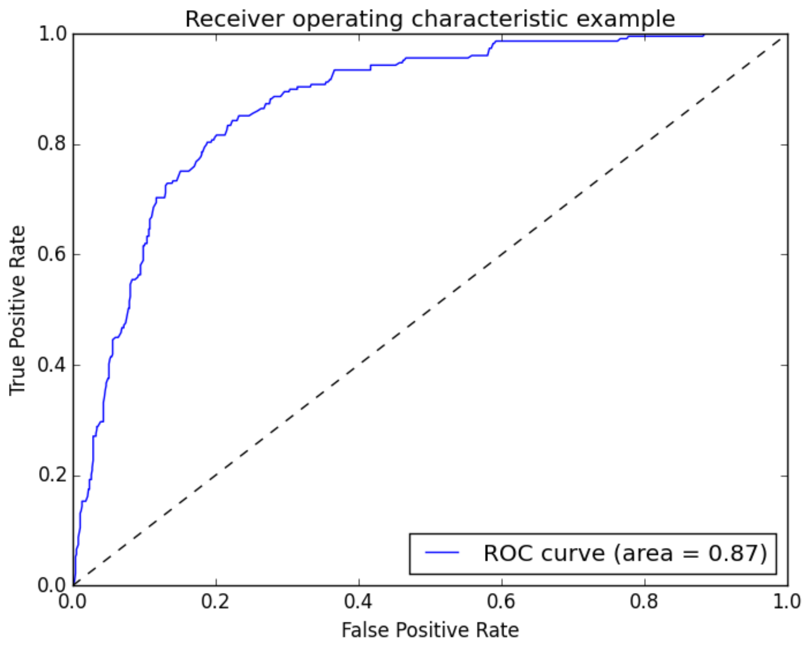
As stated before, sadly we were not able achieve any meaningful results through our Siamese Neural Network. After training, our confusion matrix was:

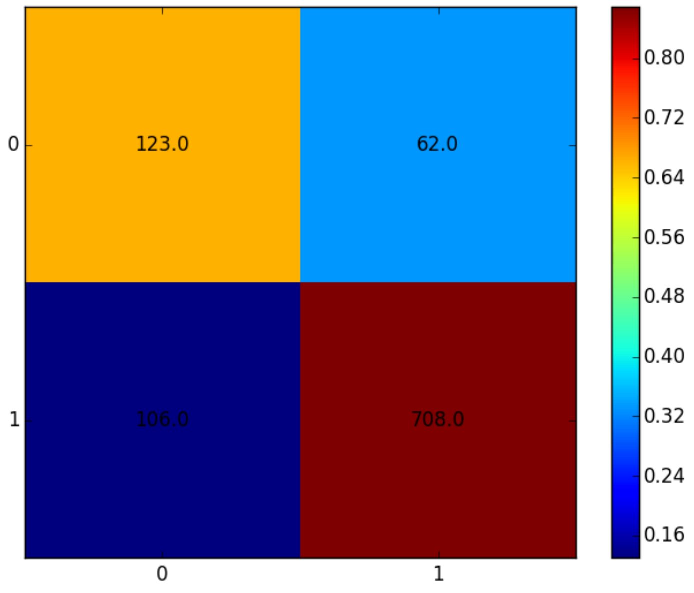


The model does well on true positives, yet also has an alarmingly high false positive rate. While we regret that we weren't able to spend more time refining this approach, we were very positively surprised by how good our results were for the Feed Forward Neural Network.

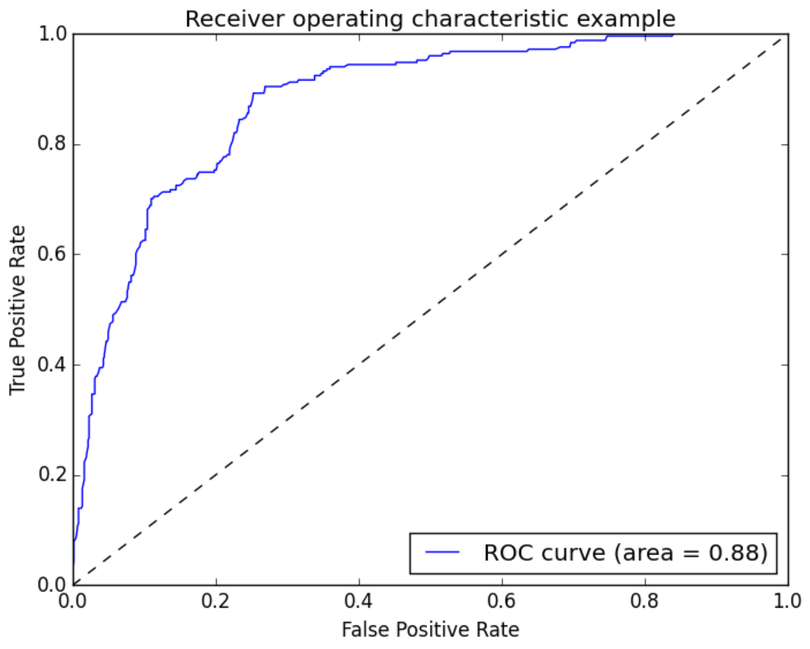
4.1.2 Feed Forward Neural Network

When running our first experiments, we got an auc of between 0.82 and 0.84. However, after running hyperparameters we were able to improve this to 0.876:

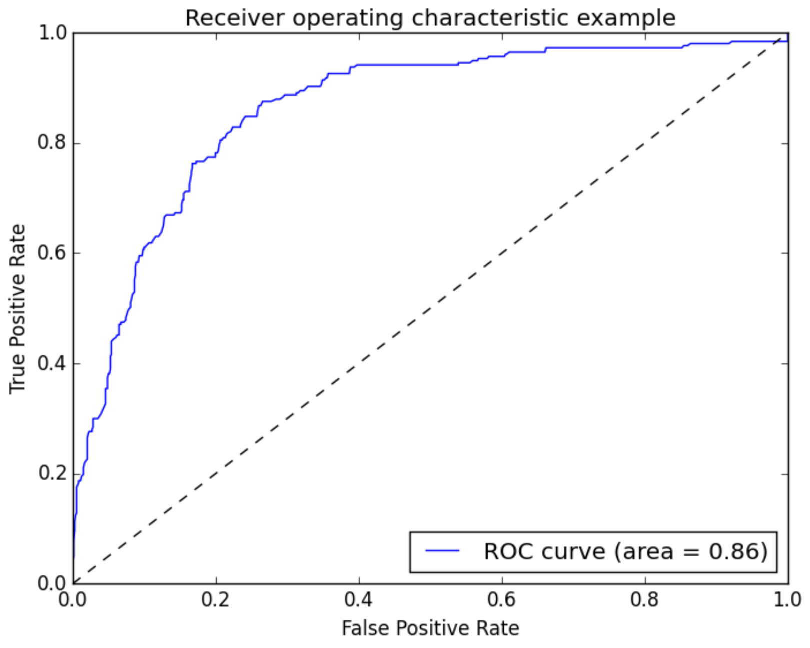


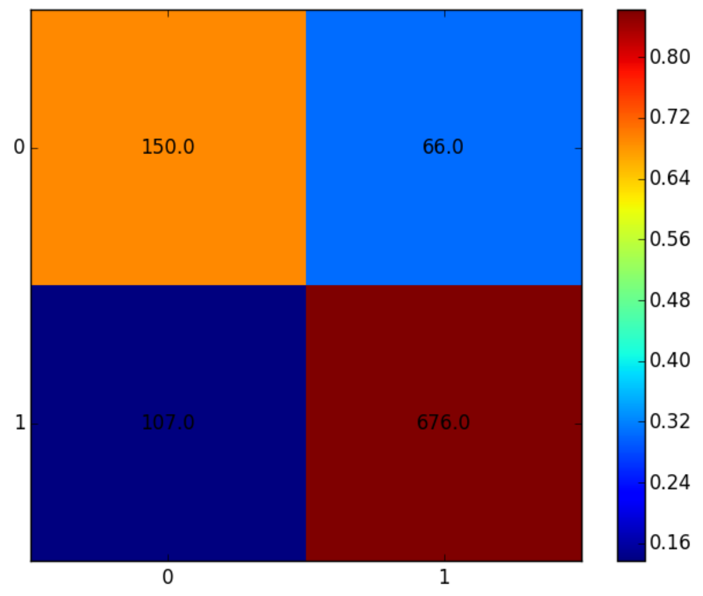


These optimal hyperparameters were a learning rate of 0.01, a training of 100 epochs and an lrdecay of 0.99999.
Clearly, we can see that we nearly achieve the accuracy of the random forest model after tuning our network.
Adding weights decay, we were able to improve our AUC slightly, as seen below:



Here we were just as good as the random forest model.
Adding the second hidden layer did very little to improve our network:





Both perform much better than our siamese variant. One important note is that the networks are generated with randomly initalized paramenters. Throughout training we noticed that this was responsible for some variation in the performance, which may explain the slight differences between our models.

5 Conclusion

We have clearly seen that classifying documents as relevant or non-relevant, as initially labeled by human experts is in fact quite possible to learn. Even with a quite straightforward model and simple mean document vectors, we were able to get our AUC to nearly 90%. We think that through only minor improvements on the system, we should easily be able to break that threshold. However, as stated above, even this very simple model was able to contend against highly optimized non-deep approaches.
In the future, we do think that there is room for a Siamese Neural Network here, and we will continue to work on this over the summer. We think that this could add a significant improvement to the statistics and look forward to further testing.
Finally, we see there may be difficulties in scaling this problem. Currently our vector space model is only the world of political news. If this were to expand (to say news in general) we think that the vectors may be too similar to each other for us to find any meaningful difference between the two. But here too, we hope that further experimentation will prove us wrong.

Acknowledgments

Special thanks to James Ostheimer for invaluable advice and McKinley Smith for help with the dataset.

References

[1] Bromley, J., Guyon I,. LeCun, Y., Sckinger, E. & Shah R. Signature Verifcation using a ”Siamese” Time Delay Neural Network. In *International Journal of Pattern Recognition and Artificial Intelligence*, 1993.

[2] Chopra S., Hadsell R & LeCun Y. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Computer Vision and Pattern Recognition*, 2005.

[3] Le, Q. & Mikolov, T. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[4] Zanzotto, F., Korkontzelos, I., Fallucchi, F. & Manandhar, S. Estimating Linear Models for Compositional Distributional Semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.