Table of Contents

[1 Introduction 1](#__RefHeading___Toc895_795545725)

[2 Literature Review 1](#__RefHeading___Toc897_795545725)

[2.1 Neural Networks 2](#__RefHeading___Toc899_795545725)

[2.1.1 Recursive Neural Networks 2](#__RefHeading___Toc901_795545725)

[2.1.2 Convolutional Neural Networks 2](#__RefHeading___Toc903_795545725)

[2.1.3 Recurrent Convolutional Neural Networks 2](#__RefHeading___Toc905_795545725)

[2.2 Types of Classification 3](#__RefHeading___Toc958_795545725)

[2.2.1 Logistic Regression 3](#__RefHeading___Toc960_795545725)

[2.3 Types of Distributed Representations of Words and Documents 3](#__RefHeading___Toc907_795545725)

[2.3.1 Word2vec (Shallow Neural Network Implementation) 3](#__RefHeading___Toc909_795545725)

[2.3.2 Doc2Vec (Shallow Neural Network Implementation) 4](#__RefHeading___Toc911_795545725)

[2.4 Frameworks and Libraries 4](#__RefHeading___Toc913_795545725)

[2.5 Datasets 4](#__RefHeading___Toc915_795545725)

[2.6 Summary of Research 5](#__RefHeading___Toc917_795545725)

[3 Development 6](#__RefHeading___Toc919_795545725)

[3.1 Experimental Development 6](#__RefHeading___Toc921_795545725)

[3.1.1 Tensorflow 6](#__RefHeading___Toc923_795545725)

[3.1.2 Word2Vec in Tensorflow 6](#__RefHeading___Toc925_795545725)

[3.1.3 Gensim Doc2Vec 7](#__RefHeading___Toc927_795545725)

[3.1.4 Conclusions of Experimental Development 7](#__RefHeading___Toc929_795545725)

[3.2 Developing and Training the Model 8](#__RefHeading___Toc931_795545725)

[3.2.1 Parsing Yelp Reviews 8](#__RefHeading___Toc935_795545725)

[3.2.2 Training Parameters 10](#__RefHeading___Toc937_795545725)

[3.2.3 Tuning Parameters 10](#__RefHeading___Toc939_795545725)

[3.2.4 Classifying the Reviews 11](#__RefHeading___Toc941_795545725)

[3.2.5 Storing Classified Reviews 11](#__RefHeading___Toc943_795545725)

[3.2.6 Predicting Unseen Documents 11](#__RefHeading___Toc945_795545725)

[3.3 Developing the User Interface 12](#__RefHeading___Toc947_795545725)

[Evaluation 12](#__RefHeading___Toc949_795545725)

[Conclusion 12](#__RefHeading___Toc951_795545725)

[Bibliography 13](#__RefHeading___Toc953_795545725)

# 1 Introduction

# 2 Literature Review

## 2.1 Neural Networks

### 2.1.1 Recursive Neural Networks

Recursive Neural Networks (RvNN) approach the computation of word vectors by applying a particular compositionality function over a binary tree recursively. It is usually done bottom-up, so each child vector is calculated, which in turn calculates the parent vector. Socher et al. (2013) propose an enhanced version of this network, known as a Recursive Neural Tensor Network, where they use the same tensor-based composition function for all nodes in the binary tree. While Recursive Neural Networks are a powerful tool in natural language processing, as Mikolov et al. (2013) points out, they are often subject to high computational complexity due to their nonlinear hidden layers.

### 2.1.2 Convolutional Neural Networks

Although Convolutional Neural Networks (CNN) are often used for tasks such as image classification, they’re also extremely useful for sentiment analysis. Kim, Y. (2014) shows that a CNN trained with one convolution layer achieves excellent results across multiple benchmarks.

### 2.1.3 Recurrent Convolutional Neural Networks

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015) propose a new type of neural network that takes advantage of recurrent neural network’s O(n) time complexity. While RNNs are a biased model, putting emphasis on later words in a corpus rather than earlier words, RCNNs use an unbiased max-pooling layer that gives equal emphasis across the corpus. RCNNs also take advantage of a bidirectional recurrent structure, which produces considerably less noise than a traditional window-based neural network.

## 2.2 Types of Classification

### 2.2.1 Logistic Regression

Logistic regression is a statistical method that aims to categorise the dependent variable into a binary category – that is, given an input Y, the likelihood of Y being 0 or 1. In the context of machine learning, this usually means taking feature vectors (n-dimensional vectors that describe learned features relating to the set of training data), and fitting them into this function.

## 2.3 Types of Distributed Representations of Words and Documents

### 2.3.1 Word2vec (Shallow Neural Network Implementation)

Word2Vec is a powerful model developed by researchers at Google led by Tomas Mikolov that produces rich word embeddings. It improves on existing work in this space by proposing two new models for learning distributed representations of words. The goal of these new models was to minimise the computational complexity of the original models by removing the non-linear hidden layer from existing feedforward and recurrent neural net language models (NNLM).

Mikolov et al. (2013) propose a continuous bag of words model (CBOW) that is similar to the feedforward NNLM model where the non-linear hidden layer is removed. This reduces the computational complexity, allowing a much larger vocabulary to be used during the training step. It is trained by trying to classify a target word from a window of source words.

They also propose a continuous skip-gram model (SG), which is similar to CBOW, but instead performs the inverse training step by trying to classify source words based on a target word in the same sentence.

Mikolov (2013) suggested during a discussion of the various use-cases for these models that the CBOW model works better for a dataset with short sentences but a high number of samples (i.e. tweets, short reviews), while the SG model works better for a dataset with long sentences but a low number of samples (i.e. small quantity of large documents).

### 2.3.2 Doc2Vec (Shallow Neural Network Implementation)

Following the research led by Mikolov et al. (2013) that produced word2vec, researchers tried to extend the word2vec model to produce phrase-level or sentence-level representations. At first a simple approach was used, which simply created a weighted average for all words in a corpus, thus giving a weighted model for a complete sentence. This has weaknesses, as Mikolov (2013) states: “The first approach, weighted averaging of word vectors, loses the word order in the same way as the standard bag-of-words models do”.

The extended work done by Mikolov et al. (2013) borrows the same fundamental concepts that were used to create the word2vec models. The only difference being that instead of applying the SG and CBOW models to words in sentences, it is applied to sentences within a corpus.

Doc2Vec will be used for my own implementation of a sentiment analysis deliverable. I will train and test my model using the Yelp dataset.

## 2.4 Frameworks and Libraries

Existing frameworks and libraries can be used to save time implementing the neural network models previously mentioned. Thanks to their popularity, models such as word2vec and doc2vec have been implemented in various libraries. Gensim (Rehurek, R., 2014) is one such library that has implementations of the work done by Mikolov et al. (2013) on Word2vec and Doc2Vec.

## 2.5 Datasets

Yelp (a popular online review website) offers a comprehensive dataset for academic use. It contains approximately 4.7 million text-based reviews.

## 2.6 Summary of Research

The research surrounding sentiment analysis highlights its often complex nature. Representing language in a model that can be consumed by a natural language processing tool while maintaining its context can prove difficult. There are multiple approaches to this problem, but it is clear from the research undergone that a neural network backed solution will perform the best. Various types of neural networks each offer their own unique advantages, and can be tuned to increase performance.

For training the model, the CBOW architecture will be used, as it should perform better than the SG model due to its high-volume of samples that are short in length (see Yelp dataset).

Although there are existing libraries for these (doc2vec, word2vec) models, it was still important to know how the algorithms are implemented at a naive level. Tensorflow’s tutorial for implementing the word2vec algorithm using Python and Tensorflow was followed to gain a basic understanding of the implementation.

In order to train the models in a neural network, it is useful to have a large text corpus to train with. Thankfully, Yelp (a popular online review website) have provided a dataset for academic use. It contains approximately 4.7 million text-based reviews that can be used for training and testing the models.

The purpose of this research and project is to produce a sentiment analysis application tailored towards businesses with an online presence. Businesses often only get a very uninformative high-level insight into their reviews (i.e. star ratings, number of 1-5\* reviews etc.).

These businesses would benefit from an application that uses sentiment analysis to analyse their reviews and output meaningful data that they can use to improve their services.

At a high-level, this application will provide:

* Categorisation of reviews into good and bad categories.
* Filtering of reviews not only by their sentiment but also by their context. For example, a restaurant could see all of the good reviews that mentions the quality of their food.

# 3 Development

In this chapter, we explore the development process that lead to the production of the sentiment analysis application. It will focus on relevant ideas learned while studying existing literature and applying them to the domain of sentiment analysis.

## 3.1 Experimental Development

Now that appropriate neural network models and datasets had been identified, experimental development was undergone to find the most appropriate platforms and libraries for sentiment analysis.

### 3.1.1 Tensorflow

Tensorflow is an option for developing neural networks. The main benefit of using Tensorflow is that it abstracts a lot of the complexity around developing a platform to create a neural network. Things such as types of neural networks, regression models, training models and loss models are made easy to implements thanks to many of Tensorflow’s high-level libraries.

In order to test the viability of Tensorflow, a basic MNIST program was created (MNIST is essentially the “Hello World” program of the machine-learning space). It was very easy to implement the soft-max regression and the loss function.

### 3.1.2 Word2Vec in Tensorflow

Once a basic understanding of the Tensorflow library was attained, the viability of Tensorflow for implementing the work of Mikolov et al. (2013) on the Word2vec model was tested. Unfortunately, Tensorflow does not provide an implementation of the Word2vec model meaning that the model had to be implemented from scratch.

Although the algorithm implementation was fairly trivial to implement and Tensorflow provides some out-of-the-box convenience libraries for dealing with large amounts of data, the implementation was inefficient.

### 3.1.3 Gensim Doc2Vec

Alternatively, Gensim (an open-source vector space modelling library) provides a highly optimised implementation of the Doc2Vec model. Additionally, Gensim’s implementation of the Doc2Vec model provides convenience functions for inferring new vectors, finding similar documents, and storing models for future use.

### 3.1.4 Conclusions of Experimental Development

While Tensorflow is a viable framework to build our Doc2Vec model on, Gensim’s library clearly provides a more complete solution for our problem domain.

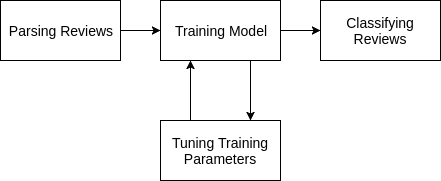
Gensim’s training cycle is much faster than Tensorflow’s. This is most likely due to Gensim optimising their implementation of Doc2Vec by providing efficient file streaming via their ‘utils’ library and also by using NumPy vectors for storing feature vectors within the model.

|  |  |  |
| --- | --- | --- |
| **Library** | **Corpus Size** | **Time Taken** |
| Tensorflow |  |  |
| Gensim |  |  |

Figure 3.x - comparisons of the training time for each library.

## 3.2 Developing and Training the Model

This section will describe the development process that was undergone to produce the Yelp Doc2Vec model. Figure 3.2.x shows – at a high level – the development process from start to finish. Firstly, it is established how Yelp’s JSON reviews will be parsed into a format digestible by Gensim’s Doc2Vec implementation. Secondly, the configuration of the training process is described, including how parameters were tuned to improve the accuracy of the model. Finally, now the model has achieved adequate accuracy, the process of fitting the model’s feature vectors into an appropriate classifier is described.

Figure 3.x shows the development cycle for producing the Yelp Doc2Vec model.

### 3.2.1 Parsing Yelp Reviews

Doc2Vec builds word embeddings by taking in a large corpus of text(s) and produces large dimensional feature vectors describing the contextual relationship between words in documents and documents themselves.

As with any machine learning process, data should be appropriately partitioned into training data and testing data. Yelp’s dataset contains ~ 5.2 million reviews, giving us a more than sufficient amount of data. Mikolov et al. (2013) used approximately 912,000 sentences to train their model. To ensure similar results, a similar amount will be used. It is worth noting that the separation of testing and training data is important. Allowing the accuracy to be tested with previously seen and trained on data skews the accuracy of the model as it will be weighted/biased towards that data. Instead, it is required to use unseen data of which we already know the sentiment of, and use that to report our accuracy.

Yelp reviews are stored in the following JSON format:

{

"funny":0,

"user\_id":"u0LXt3Uea\_GidxRW1xcsfg",

"review\_id":"FKu4iU62EmWT6GZXPJ2sgA",

"text":"I too have been trying to book an appt to use my voucher - it's been months and countless of phone calls - no response yet.\n\nAgree with the buyers beware warning. I only wish reviews on this place was posted previous to my purchase of this voucher.",

"business\_id":"fdnNZMk1NP7ZhL-YmidMpw",

"stars":1,

"date":"2012-10-23T00:00:00.000Z",

"useful":0,

"cool":0

}

Figure 3.x - the JSON structure of a Yelp review.

Gensim’s Doc2Vec model requires each sentence to be stored in a particular fashion. Each sentence should be an array of words that form the sentence, with a corresponding label identifying the sentence. For training purposes, bad reviews are labelled bad\_*i* where ‘i’ is the index of the review in the list of reviews. The same process is also done for good reviews. The review’s ID is also transferred to a second label.

[['I', 'too', 'have', 'been', 'trying', 'to', 'book', 'an', 'appt', 'to', 'use', 'my', 'voucher', '-', "it's", 'been', 'months', 'and', 'countless', 'of', 'phone', 'calls', '-', 'no', 'response', 'yet.', 'Agree', 'with', 'the', 'buyers', 'beware', 'warning.', 'I', 'only', 'wish', 'reviews', 'on', 'this', 'place', 'was', 'posted', 'previous', 'to', 'my', 'purchase', 'of', 'this', 'voucher.'],

['bad\_1', 'FKu4iU62EmWT6GZXPJ2sgA']]

Figure 3.x – a ‘LabeledLineSentence’ of the review in Figure 3.x.

Gensim refers to this as a ‘LabeledLineSentence’. Unfortunately, their implementation of this class only supports loading from a text file with sentences separated by new lines. To avoid having to transform the ~ 4.2GB JSON file, our own ‘YelpLabeledLineSentence’ class is used that makes loading the Yelp dataset into this format easier by optimising the loading of a large JSON file using Python’s JSON library and iterators, meaning the whole corpus is not loaded into RAM. This should ensure we can train on corpus sizes similar to the aforementioned

It has been reported that repeatedly training the same data in the same order decreases the learning rate over each iteration. This is because the model’s features are being ‘reinforced’ by observing the same contextual relationships between sentences in the same order repeatedly. An analogy for this is a heat map where the same data is layered over and over. To remedy this, the training data will be randomised on each iteration, providing near-unique sentence comparisons over each iteration.

For training the Yelp Doc2Vec model, a training set of 100,000 reviews was used, equally shared between 1 star and 5 star reviews. As a logistic regression classifier will be used, our training data should be binary (contains two sets of data).

### 3.2.2 Training Parameters

Gensim’s Doc2Vec model provides parameters that are analogous to the definitions in the original paper, they are as follows:

**min\_count** – The minimum occurrences of a sentence label in the corpus to be considered in the model.

**window –** The maximum distance between the current and predicted word within a sentence.

**size –** The dimensionality of the feature vectors (how ‘detailed’ do we want our feature vectors to be).

**sample** – The threshold for configuring which words are randomly downsampled.

**negative** – Specifies how many ‘noise words’ will be drawn during negative sampling.

**iter** – The number of training iterations (epochs).

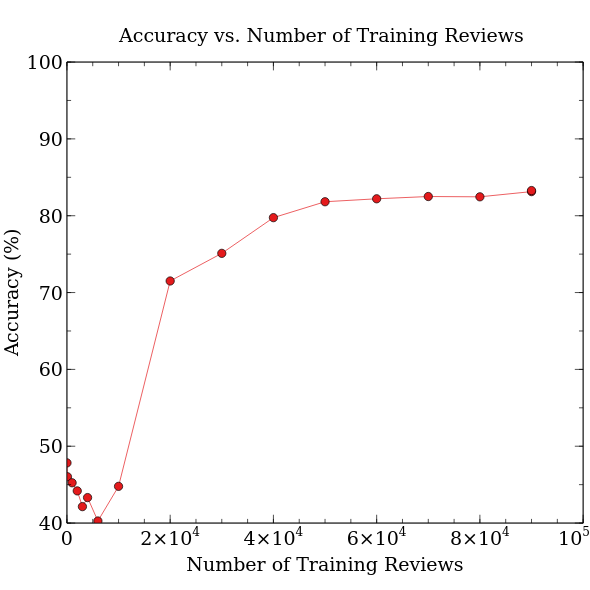
**workers** – The number of threads to train the model with. Provides faster training on multi-core machines.

Finding the appropriate configuration was a mixture of drawing from well-established default values, and experimental trial-and-error tests.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Value** | **Justification** |
| min\_count | 1 | Each sentence’s unique label appears only once. |
| window | 10 | A reasonable number for the average sized Yelp review. |
| size | 300 | Sufficient dimensionality for required accuracy. |
| sample | 1e-4 | The paper sets an ideal value of 1e-4. |
| negative | 5 | (find source) speculates this as an ideal value. |
| iter | 5 | Sufficient iterations for required accuracy. |
| workers | 4 | Maximum number of threads to utilise for training. |

### 3.2.3 Tuning Parameters

As the time taken to train the model correlates with the amount of training reviews used, accuracy was logged for each increase of training reviews to find an ideal amount.

Fig 3.x shows accuracy (%) for a number of training reviews. It can be inferred that increasing the number of training reviews past 6×104 provides negligible improvement of accuracy. A table of results can be found at appendix x.

### 3.2.4 Classifying the Reviews

* Logistic Regression
* Accuracy
* ‘Sanity checks’

### 3.2.5 Storing Classified Reviews

* Large dataset (~100,000 reviews + ~150,000 businesses)
* MongoDB (easy in both Python and Node.js)
* Calculating sentiment before or after insertion to database? (before - much quicker lookup times)

### 3.2.6 Predicting Unseen Documents

* Inferring vectors of user input
* What is an unseen doc?
  + Keywords in docs (i.e. food, drink, service)
  + Full text corpus (i.e. a new review)
* Finding related documents

## 3.3 Developing the User Interface

# Evaluation

Confusion matrix

# Conclusion

# Bibliography

Socher, R., Perelygin, A., & Wu, J. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. Proceedings of the …, 1631–1642. Retrieved from <http://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf>

Santos, C. N. dos, & Gatti, M. (2014). Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. Coling-2014, 69–78. Retrieved from <http://www.aclweb.org/anthology/C14-1008>

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. <https://doi.org/10.3115/v1/D14-1181>

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent Convolutional Neural Networks for Text Classification. Retrieved from <https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9745/9552>

Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 1–12.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 26 (pp. 3111–3119). Curran Associates, Inc. Retrieved from <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>

Tomas Mikolov. (2013). de-obfuscated Python + question - Google Groups. Retrieved November 25, 2017, from [https://groups.google.com/forum/#!msg/word2vec-toolkit/NLvYXU99cAM/E5ld8LcDxlAJ](https://groups.google.com/forum/" \l "!msg/word2vec-toolkit/NLvYXU99cAM/E5ld8LcDxlAJ)

RaRe Technologies. (2016). Doc2Vec Tutorial on the Lee Dataset. Retrieved from <https://github.com/RaRe-Technologies/gensim/blob/develop/docs/notebooks/doc2vec-lee.ipynb>

RaRe Technologies. (2015). Gensim Doc2vec Tutorial on the IMDB Sentiment Dataset. Retrieved from <https://github.com/RaRe-Technologies/gensim/blob/82c394a9085d583e8a75c2bb32ecd37cf61236f0/docs/notebooks/doc2vec-IMDB.ipynb>

Le, Q. V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents, 32. Retrieved from <http://arxiv.org/abs/1405.4053>

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Retrieved from <https://arxiv.org/abs/1603.04467>

Rehurek, R. (2014). gensim: Topic modelling for humans. Retrieved from <http://radimrehurek.com/gensim/>

Tensorflow. (n.d.). Vector Representations of Words | TensorFlow. Retrieved November 26, 2017, from <https://www.tensorflow.org/tutorials/word2vec>

Yelp Inc. (n.d.). Yelp Dataset. Retrieved November 25, 2017, from <https://www.yelp.com/dataset>