Faculty of Arts, Computing, Engineering & Sciences

Department of Computing

Final Year Individual Project (SEGM)

[55-604708]

2017/18

Author:	**Ben Ewen**
Date Submitted:	**11/4/18**
Supervisor:	**Jing Wang**
Degree Course:	**Computer Science**
Title of Project:	**Using natural language processing and deep learning to predict sentiment in online reviews**

Confidentiality Required? NO

Abstract

Business owners rely heavily on the online review industry to advertise and collect highly-valuable feedback on their services. While this gives a wide insight into a business' performance across multiple areas - collating, sorting, and analysing this customer feedback via online review sites is often time-consuming and can leave businesses unsure on where they should improve. Consequently, online review sites have recently turned to algorithms for classifying reviews, yet these often fall short of analysing the text of reviews and tend to rely on metadata for categorisation. This project will expand and improve this categorisation process by introducing novel machine learning algorithms that effectively perform sentiment classification tasks, thus further enabling online review sites to granularly sort and classify their reviews, and in turn, providing business owners with the relevant insights into their operations and potential improvements thereof.

Table of Contents

1 Introduction

Most people today are familiar with online review services; anyone can go online to find ratings and reviews for a particular establishment. Due to the popularity of such services, businesses strive to get the best possible ratings they can, although reviews are notoriously unreliable in giving an overview of a business and often give little to no accurate insight. Fortunately, technologies now exist that aim to automate the art of language-processing in such a way that computer algorithms, to an appropriate degree of accuracy, can predict the context and sentiment of corpora. Algorithms such as these pertain to the study of natural language processing, which recently, has started to combine the power of machine learning techniques, and complex language representations, to perform sentiment analysis and classification upon large corpora.

This paper proposes the implementation of a sentiment analysis tool, driven by machine-learning, that can be integrated into existing online review systems and allows business owners and review websites to harness the power of machine-learning algorithms to automatically classify their reviews into categories of interest. Various existing approaches to sentiment analysis will be explored, extracting relevant ideas and useful algorithms which will then be applied to the development of a machine learning model, this model will then be used to classify reviews based on notable features within them. The resulting application's performance will be evaluated against existing solutions to see how it compares and a proposal of suggested future work will follow.

2 Literature Review

2.1 Neural Networks

Neural networks are computer systems that aim to mimic the rudimentary actions of the neural networks found in animal’s brains. They almost always contain a training step where the network is asked to perform a repetitive action in order to identify relevant characteristics in a particular set of data. During this step, neurons within the network start to affect the resulting output by weighting particular neurons based on the input. Over time, the neural network starts to learn characteristics of the input data that can then be used for a myriad of classification tasks such as image recognition, voice recognition, and text classification. For the scope of this project, neural networks that are viable for sentiment analysis will be discussed.

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 non-linear 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 (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

Now that the neural network is capable of outputting characteristics, this data can be aggregated via various methods to enable the classification of data from the neural network. This section discusses viable classification methods and discusses common problems that classification tasks face.

2.2.1 Linear Regression

Linear regression is used to identify relationships between two continuous variables. An independent variable is used to predict the value of a dependant variable. Linear regression is useful when mapping probabilities of a continuous dependent variable. For example, what is the most likely amount of ice-creams sold on a day with the temperature x?

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.

2.2.2 Over-fitting Classifiers

A common problem in the machine learning space is over fitting. Over-fitting occurs when the model is not easily generalisable, the model will train to the noise and detail of the training set, rather than the general relations behind it. This can often be identified by accuracy rates, if the accuracy of the model is very high for the training data, but very low for unseen data, the model is most likely over-fit.

2.3 Types of Distributed Representations of Words and Documents

In section 2.1 it was explained that neural networks take an input, process it via neural connections, and produce an output. Consider the most effective way of representing input data to a neural network, the representation of the input should not diminish the resolution of the input (i.e. if the task is image classification, the neural network should be able to analyse every pixel, therefore the representation of an image should allow for this). For text classification, this can be less obvious, how can a corpus be represented efficiently, consistently and without the loss of any ‘resolution’? This section aims to provide solutions for the aforementioned problem.

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.

2.4 Frameworks and Libraries

2.4.1 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 implement thanks to many of Tensorflow's high-level libraries.

2.4.2 Gensim

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.

Based on the research of representations of words, for training the model, the Doc2Vec 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 the Doc2Vec model, it will still be important to know how the algorithms are implemented at a naive level.

In order to train the models in a neural network, it is useful to have a large text corpus to train with. Yelp, a popular online review website, have provided a dataset for academic use containing approximately 4.7 million text-based reviews that can be used for training and testing the models. In order to classify such a large dataset into ‘good’ and ‘bad’ categories, a logistic regression classifier should prove efficacious due to their proficiency in binary classification.

This project looks to produce a sentiment analysis application that is tailored towards businesses with an online presence and aims to improve the yield of useful information from their review data, rather than just the review metadata (i.e. star ratings, number of 1-5* reviews etc.). These businesses could benefit from such an application in order to analyse their reviews and output more meaningful data that they can use to improve their services or product. As proof of this concept, the categorisation will be limited to predicting the sentiment of a review as either ‘good’ or ‘bad’, and providing related reviews based on the common language between them.

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 models for representing reviews in vector space and datasets had been identified, experimental development was undergone to find the most appropriate platforms and libraries for sentiment analysis.

3.1.2 Doc2Vec 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 Doc2Vec model was tested. Unfortunately, Tensorflow does not provide an implementation of the Doc2Vec model meaning that the model had to be implemented from scratch.

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

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 for the large size of the Yelp dataset. Gensim's library clearly provides a more appropriate solution for our problem domain, appealing to developers wishing to handle large sets of training data.

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.

3.2 Developing and Training the Model

This section will describe the development process that was undergone to produce the Yelp Doc2Vec model. Figure 3.1 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.

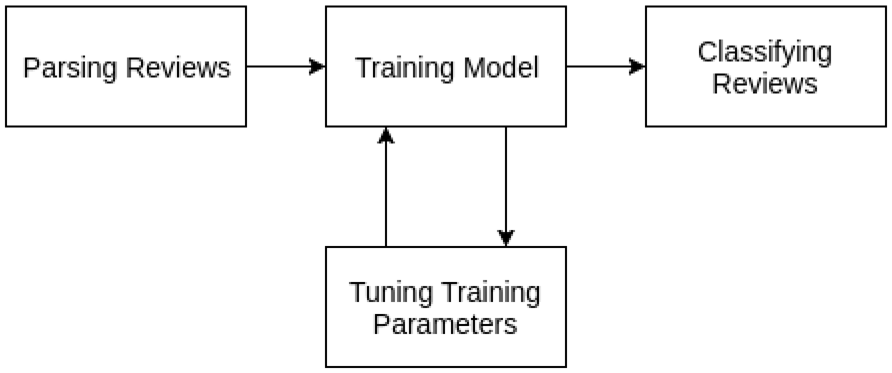


Figure 3.1 shows the development cycle for producing the Yelp Doc2Vec model.

3.2.1 Parsing Yelp Reviews

Doc2Vec builds word embeddings by taking in a large corpora 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 JSON format. JSON is a simple object notation from Javascript that describes objects as key/value pairs. The keys of most interest to train the model with are "text" and "stars". These will be used to train the model's vocabulary and sort the training data.

```
{
 "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"
}
```

Figure 3.2 – the JSON structure of a Yelp review, with the keys highlighted in bold.

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 3.3 – a 'LabeledLineSentence' of the review in Figure 3.2.

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 900,000 reviews.

Furthermore, as a logistic classifier will be used, our training data will be sorted into two features - 'good' and 'bad'. 'Good' describing any reviews with a star rating of 4 or more, and 'bad' describing reviews with a star rating of 2 or less. 3 star reviews are naturally expected to be less opinionated than more polar ratings, and are therefore excluded from training classification. When the model is later fitted into the logistic regression classifier, we can maximise the accuracy of its predictions by providing well fit feature vectors. This is described in more detail in section 3.2.4 of this chapter.

To summarise, there is now an efficient way of parsing and loading large sets of Yelp reviews into the Doc2Vec model. The bespoke 'YelpLabeledLineSentence' class provides efficient data loading via Python iterators, convenience functions for retrieving and labelling sentences for our two features 'good' and 'bad, and the ability to randomise the training set for each training iteration.

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:

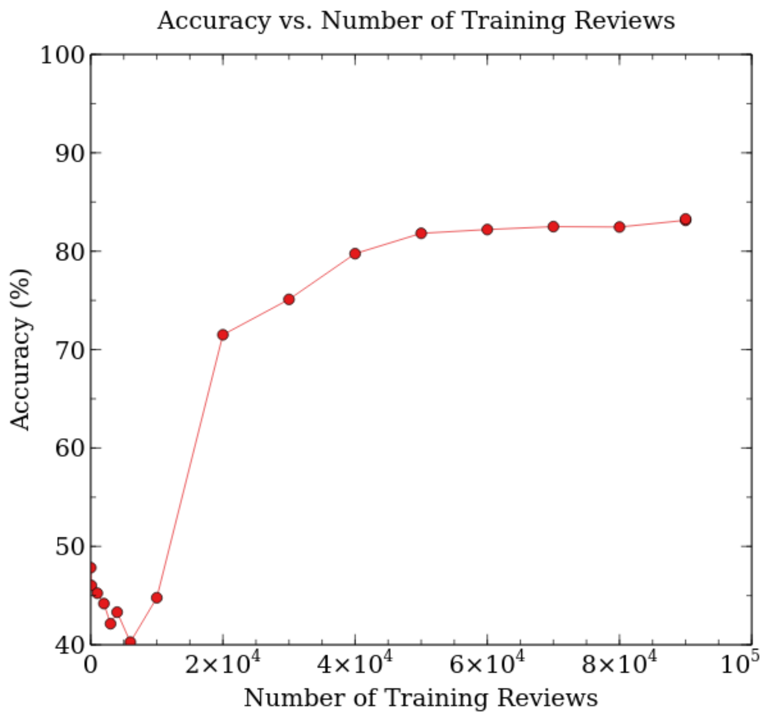
min_count – The minimum occurrences of a sentence label in the corpus to be considered in the model.
window – The maximum distance between the current and predicted word within a sentence.
size – The dimensionality of the feature vectors (how ‘detailed’ do we want our feature vectors to be).
sample – 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. The ideal configuration would provide the best possible accuracy at the lowest possible computational cost. Comparing large n-dimensional vectors is computationally costly, especially with a large data set.

The Accuracy of the model can be calculated by observing the number of correctly categorised reviews vs. the total number of reviews. As in section 3.2.1, our testing data can be similarly constructed by taking **unseen** reviews (i.e. reviews not in the set of training reviews) <=2 and >=4, labelling them good or bad respectively.

Number of Reviews

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. An ideal amount would provide sufficient accuracy at the smallest possible number of reviews. This will be helpful when tuning the iterations parameter, as a smaller number of reviews will reduce the time taken for each iteration.



Graph 3.1 shows accuracy (%) for a number of training reviews. It can be inferred that increasing the number of training reviews past 6×10^4 *provides negligible improvement of accuracy. A table of results can be found at appendix x.*

Min_count

"Min_count" was borrowed from the Word2Vec model that Doc2Vec was built upon. It originally meant the minimum threshold that words in a sentence should appear to be included in the model. When translated to Doc2Vec, this parameter has lost its usefulness, as sentences that appear within documents will most likely only ever appear once (they are unique). This leaves no choice but to set this value to 1.

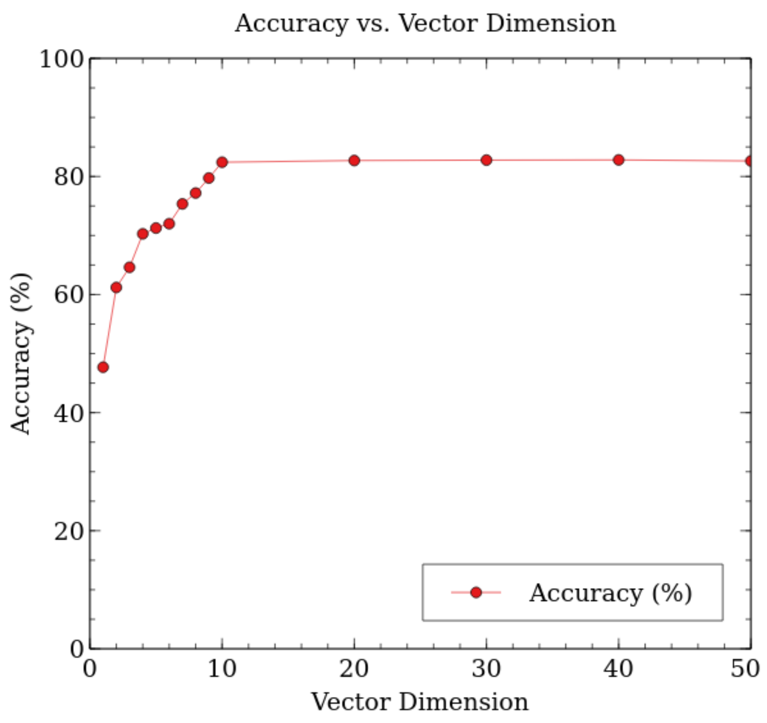
Window

The maximum distance between the current and predicted word within a sentence. To provide rich feature vectors, there should be sufficient overlap between word comparisons within a sentence. The average length of a Yelp review is approximately 30 words, across all stars, therefore a window size of 10 should suffice.

Size

“Size” describes the dimensionality of the feature vectors produced for each document. Doc2Vec is describing the relationships between documents as a function of their cosine similarities, returning 10 dimensional vectors for each review. If the cosine of the angles between two reviews is small, it can be inferred that the two reviews are similar. Scaled to the size of the Yelp dataset, these 10 dimensional vectors can be used to classify reviews into binary categories (good/bad) by observing the relative cosine similarities between all the review’s vectors.

It is important to set a size that sufficiently captures enough detail to produce meaningful classification, but not so high that the model is ‘over-fit’. It was observed that a dimensionality of 10 is ideal.



Graph 3.2 shows accuracy for a particular feature vector dimension. It can be inferred that 10 is an ideal vector dimension.

Sample

When training over a large corpus, it may be desirable to downsample higher-frequency words within the corpus, so that they don't unduly bias the model. In practice, as Doc2Vec traverses through document sentences, it looks to see if the current context word in the sentence has been sampled against other words before.

Very frequent words tend to not provide much contextual information compared to rare words. For example, having the model consider the co-occurrence for the words "good" and "food" is much better than comparing the words "the" and "food" as "the" commonly co-occurs with nouns in sentences. According to Mikolov et al. (2013), An ideal value for sampling is usually 1×10^{-5}.

Negative

The goal of Doc2Vec is to maximise the similarity of the vectors for sentences that appear close together, and minimise the similarity of the vectors for sentences that appear far apart.

$$\frac{V_c \cdot V_w}{\sum V_c' \cdot V_w}$$

Equation 3.1 shows the equation for computing the similarity between a document context V_c and a target sentence V_c. The similarity is summed across all other contexts V_c'.

This formula does not consider the computational complexity of calculating the similarity for all other contexts. Instead, it is better to only consider a few random extra contexts, reducing the time required to train the model.

Mikolov et al. (2013) suggests a value of 5 for negative-sampling. Further elaboration on negative sampling can be found at appendix C.

Iter

"Iter" describes the number of passes over the corpus used to train the model. The most benefit is had when training against a relatively small corpus, where the number of unique documents is low, as it enables the model to produce models comparable in detail to those produced by much larger corpora.

While increasing the iterations provides improved relationships within the model, Mikolov et al (2013) states that "training a model on twice as much data using one epoch [iteration] gives comparable or better results than iterating over the same data for three epochs". For a data set as large as Yelp's, just 1 iteration should suffice.

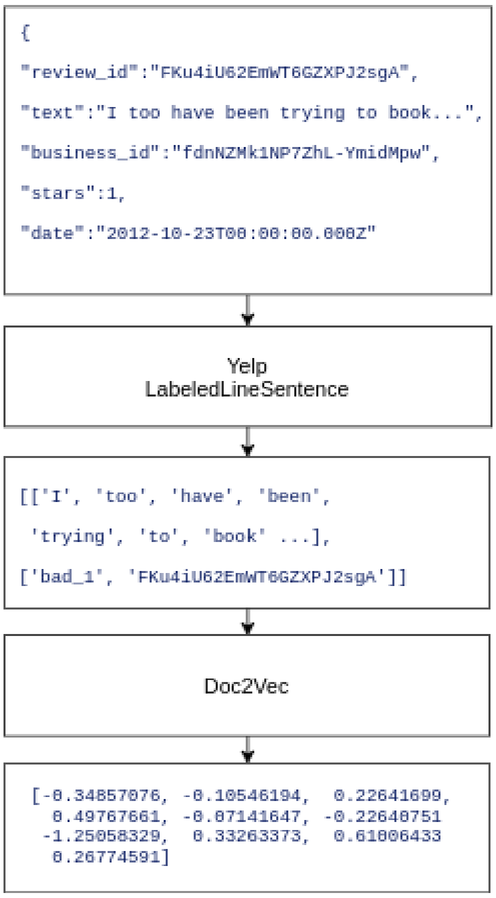
Workers

Gensim’s Doc2Vec library can parallelise the training process by spawning multiple worker threads on multicore machines. The PC training the Yelp model had 4 available cores.

These parameters should provide the most optimal performance for training the Yelp model.

Parameter	Value
Number of reviews	6×10^4
min_count	1
window	10
size	300
sample	1×10^{-5}
negative	5
iter	1
workers	4

Figure 3.4 shows each Doc2Vec parameter with its corresponding value.



```
{
"review_id":"FKu4iU62EmWT6GZXPJ2sgA",
"text":"I too have been trying to book...",
"business_id":"fdnNZMk1NP7ZhL-YmidMpw",
"stars":1,
"date":"2012-10-23T00:00:00.000Z"
```

```
[['I', 'too', 'have', 'been',
 'trying', 'to', 'book' ...],
['bad_1', 'FKu4iU62EmWT6GZXPJ2sgA']]
```

```
[-0.34857076, -0.10546194,  0.22641699,
  0.49767661, -0.07141647, -0.22640751
 -1.25058329,  0.33263373,  0.61006433
  0.26774591]
```

Figure 3.5 summarises the steps taken in this section of the development process. Each review was parsed and loaded into a ‘YelpLabeledLineSentence’. Then, Doc2Vec trained its model with the aforementioned parameters, producing 10-dimensional feature vectors for each review.

3.2.4 Classifying Reviews

There are now 10-dimensional vectors describing the contextual relationship between a specific review and the rest of the reviews. In order to take advantage of these vectors, they should be fit to a particular classifier. Classifiers are often derived from regression models found in statistical methods.

Logistic regression is used to classify the Yelp data set. Logistic regression is used when the dependent variable is categorical (good/bad, yes/no). This fits the use-case – for a review, predict the probability of it being a good review or a bad review.

In order to fit our classifier, the feature vectors are recalled from the model and placed in the first index of a two-dimensional array. The second index of the two-dimensional array is for the Y value of the classifier. In logistic regression this is a 1 or a 0. As the sentiment of the training reviews is known, we can accurately specify the Y value for these reviews. 1 represents a good review, whereas 0 represents a bad review.

```
[[[-0.34857076, -0.10546194,...], 1]
…]
```

Figure 3.6 shows a 2D classifier array entry containing a feature vector and corresponding Y value.

The logistic classifier's method is then called on this two-dimensional array that contains our feature vectors and corresponding Y values like so:

```
classifier.fit(train_arrays, train_labels)
```

Now that the logistic regression classifier has been fit with our training data, the classifier is capable of inferring feature vectors of previously unseen reviews. This can be used to check the accuracy of our model. If the sentiment prediction (good/bad) of previously unseen reviews is known, the classifier can score the accuracy of its predictions by calculating the probability of the review's inferred feature vector belonging to the 'good' or 'bad' category, and cross-checking that with the known categorisation.

To do this, unseen reviews from the end of the Yelp dataset are imported into the algorithm. The same two-dimensional array structure is formed, with the first index being the inferred feature vector of the unseen review (calculated like so):

```
model.infer_vector(review[0])
```

Figure 3.7 shows how vectors are inferred for unseen reviews. Where 'review[0]' is the review text from 'YelpLabeledLineSentence' (see figure 3.3).

As the sentiment classification is already known for these unseen test reviews, the Y values (0/1) are placed in the second index of this two-dimensional structure, forming an array structure that is exactly the same as used in the fitting stage of this process (see figure 3.x).

For accuracy testing purposes, as Le et al (2014) suggests using 12,500 unseen good reviews and 12,500 unseen bad reviews against the classifier. The following accuracy was reported:

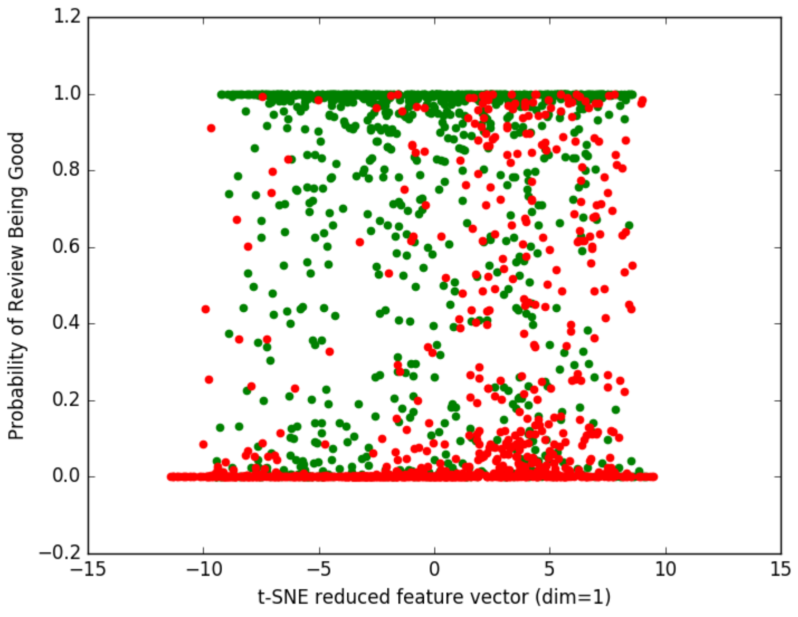
Sentiment	**Review Size**	**Accuracy**
Good	12,500	69.76%
Bad	12,500	84.62%

Figure 3.8 shows the reported accuracy of the model using logistic regression classification.

This data can be visualised by plotting the probability of each review belonging to the good category against each review's feature vector. However, there is an issue with this, as each feature-vector has 10 dimensions. This is a common problem when trying to visualise machine learning relationships as so many dimensions are impossible to visualise graphically.

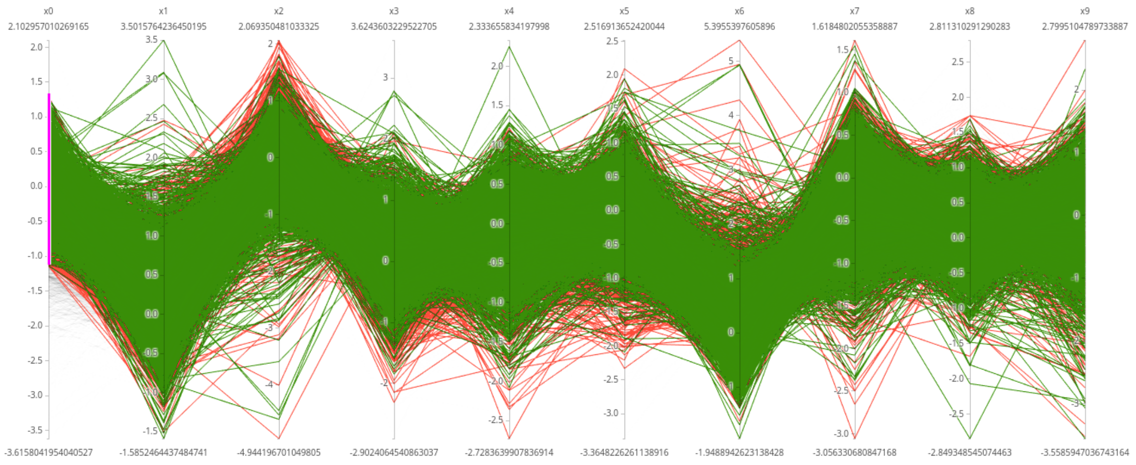
Fortunately, t-distributed stochastic neighbor embedding (t-SNE), developed by Van Der Maaten, L. J. P., & Hinton, G. E. (2008) can be used to reduce a feature vector to any n-dimension.

If the 10-dimensional feature vectors are reduced to just one dimension, they can be plotted against the probability of belonging to the 'good' category.

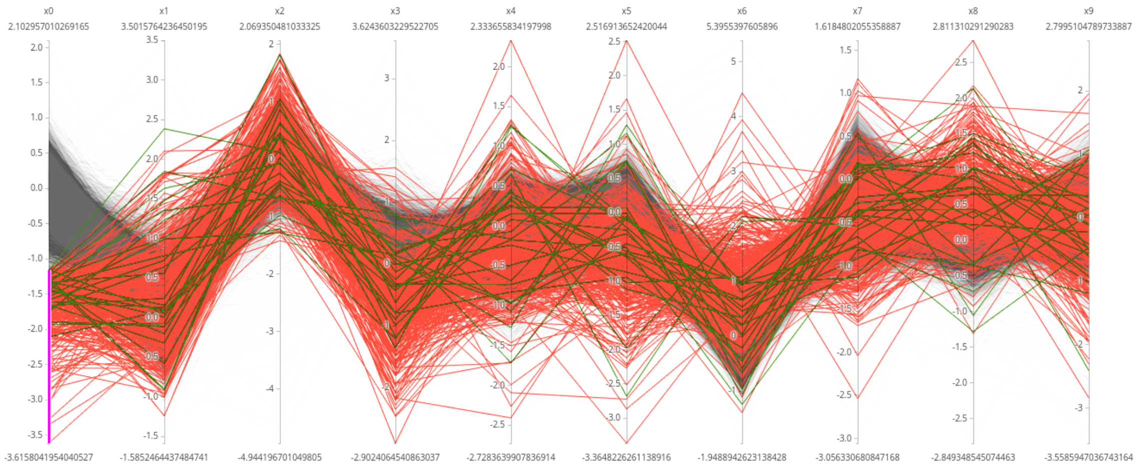


Graph 3.3 shows the probability of a particular review belonging to the good category when its feature vector is reduced to just one-dimension. Green points indicate that the review is good, whereas red points indicate the review is bad.

Alternatively, a parallel coordinate graph[1] can be used to plot all dimensions of the feature vectors. The following graphs were produced using Plotly, a graphing library for Python that enables high-fidelity graphing. Limiting the first dimension of the feature vectors to the ranges $1.4 \leq y \geq -1.1$ and $-1.2 \leq y \geq -3.5$ shows the classification of good and bad reviews respectively.



Graph 3.4 shows a parallel coordinate plot of all reviews that satisfy $1.4 \leq y \geq -1.1$ *in the dimension x0 of the feature vector. This appears to classify good reviews well.*



Graph 3.5 shows a parallel coordinate plot of all reviews that satisfy $-1.2 \leq y \geq -3.5$ *in the dimension x- of the feature vector. This appears to classify bad reviews well.*

Classifying a large portion of the Yelp data set now becomes a trivial task. The model infers feature vectors for each review in the data set which are then classified by the logistic regression classifier. One problem remains – with the Yelp data set having over 4 million reviews, there are foreseeable issues in regards to classifying and storing such a large data set.

1An interactive version of this graph can be found at: https://plot.ly/~benewen/6.embed

3.2.5 Storing Classified Reviews

Opening large files such as the Yelp data set is computationally costly. Fortunately, Gensim has a useful ‘utils’ library, containing ‘smart_open’ - a library that creates an iterator for large files.
The Yelp data set is iterated over, for each review the relevant metadata is extracted such as the star rating, corresponding business ID, submission date, and the review text. The model infers a feature vector for the current review. The feature vector is then classified via logistic regression. The resulting sentiment classification is then stored temporarily in a variable. Finally, all the review’s metadata including the newly calculated sentiment classification is stored in a MongoDB database.

MongoDB is a document oriented database. It is capable of storing collections of documents in a format called BSON. BSON is syntactically similar to JSON. MongoDB provides drivers for common programming languages, thus the MongoDB Python driver is used for this task. Additionally, Yelp’s data set provides a JSON file containing business metadata. In preparation for developing the user interface, the business data file is also parsed and stored in a separate MongoDB collection.

3.3 Developing the User Interface

This section describes how the user interface was developed. It will discuss architecture choices, how the user interface application interfaces with the Python machine learning algorithm, and features of the application available to the end user.

The vast majority of review applications are hosted as web applications, allowing customers to access their services anywhere from any device. Rather than creating a stand-alone application, the sentiment analysis tool we have created would be best fit into existing online review applications such as TripAdvisor, Yelp, Facebook, and Google. For the purposes of illustrating the functionality of the sentiment analysis tool, a hypothetical online review application will be created.

3.3.1 Application Features

The application should provide the following features:

- Simulate being able to search for reviews by a particular business. This would be the natural flow an end user would expect on existing online review platforms. This is also achievable as Yelp’s data set contains business metadata.
- Show at a glance, reviews sorted by the classifier into ‘good’ and ‘bad’ categories.
- The ability to open a review to see its full text, rating, date, and sentiment.
- Provide a list of related reviews in respect to the open review.
- Report the accuracy of the sentiment predictions.

3.3.2 Architecture

The model-view-controller (MVC) achitecture pattern will be implemented for this application. Creating an application using this pattern provides a robust modularised code-base with clearly separated concerns between each application layer. Additionally, the application will be API centric, allowing it to be easily extensible thanks to the separation of concerns in the MVC model. This API will follow the RESTful architectural style.

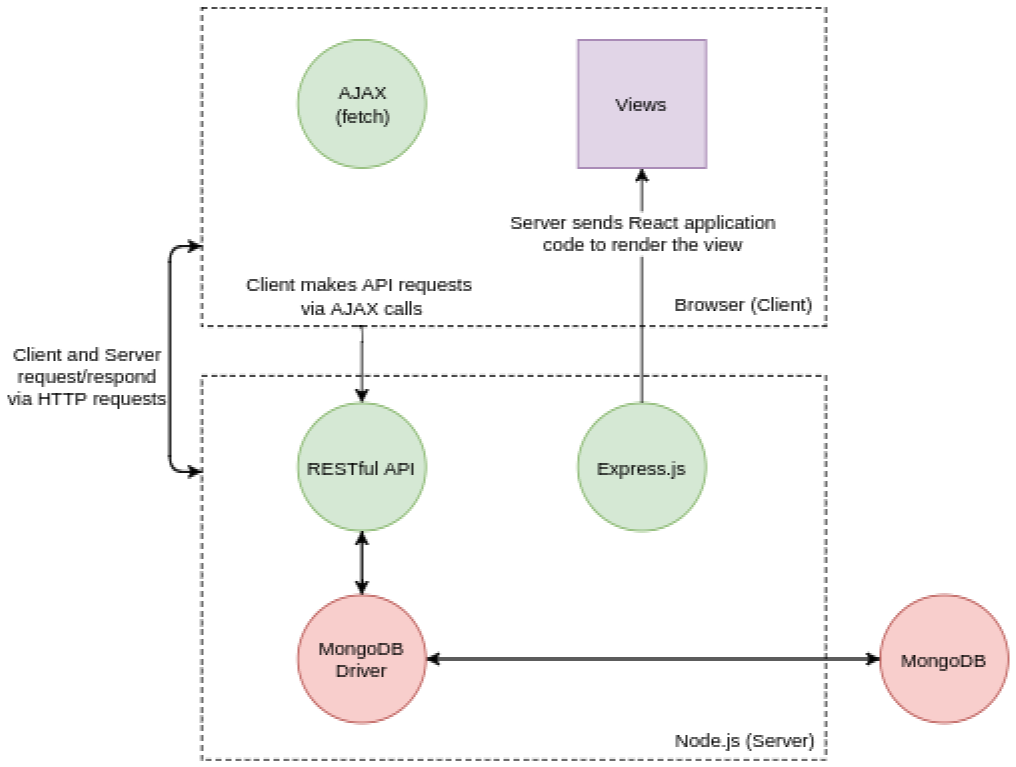

AJAX (fetch)
Views
Server sends React application code to render the view
Client makes API requests via AJAX calls
Browser (Client)
Client and Server request/respond via HTTP requests
RESTful API
Express.js
MongoDB Driver
MongoDB
Node.js (Server)

Figure 3.9 shows the architecture of the web application. Items coloured in red refer to the model, purple the view, and green the controller.

Drawing from existing experience, the following languages, libraries and frameworks will be used:

- Node.js – An asynchronous, event-driven Javascript runtime for building server-side applications.
- Express.js – A web framework for Node.js that creates lightweight web servers.
- React – A Javascript library for building user interfaces as reuasble stateful components.
- Material UI – A React library containing components that adhere to Google’s material design specification.
- MongoDB Driver for Node.js – A Javascript library that enables interaction with a MongoDB database.

3.3.3 Model

According to Burbeck, S. (1992) “the model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller)”.

Database Configuration

In section 3.2.5 it was described how Yelp reviews were stored in the MongoDB database. This now requires further elaboration. MongoDB is a document-oriented database, rather than the primary focus be on the relationships between database entries (such as in a SQL/relational database), the primary focus is on the document schema. MongoDB has the following hierarchical structure:

$$\text{Database} \rightarrow \text{Collection} \rightarrow \text{Document(s)}$$

The database contains collections of documents. Each document has a unique ID that is automatically generated upon insertion. This can be thought of as the primary key/unique identifier for each document. Additionally, documents do not have to conform to the same schema, meaning that two documents within the same collection can have different fields.

For the purposes of the sentiment analysis application, one database titled “sentiment” was created, containing two collections: “reviews” and “businesses”. These collections store the reviews and business metadata.

Interfacing with MongoDB

Similarly to interfacing with MongoDB in Python via its respective ‘driver’, MongoDB also maintains a Node.js driver. The driver provides a layer of abstraction around interacting with the database such as providing helper functions for performing CRUD operations on the database.

In summary, the model will be responsible for maintaining the data and state of the application. Any modifications, retrieval, updates, deletions will have to pass through the driver. Furthermore, any modification to the model via the controller will trigger the view to update. This ensures parity between what the user sees on the view, and the model’s state.

3.3.4 View

According to Burbeck, S. (1992), “the view manages the graphical and/or textual output to the portion of the bitmapped display that is allocated to its application.” For this application, React will provide the tools to create the application’s views as components. These components contain the visual layout of portions of the application. While Burbeck suggests that the view and controller are separate entities, React provides the ability to introduce state to components. Stateful components allow for complex, interactive user-interfaces. For example, a component can store the data it should render in its own state.

App Component

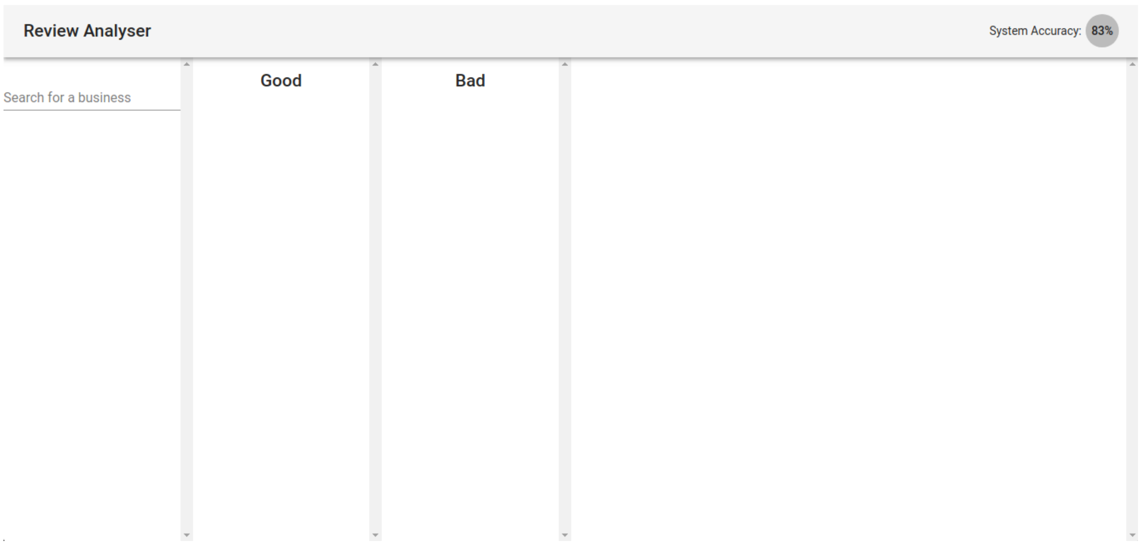


Figure 3.10 shows the App component. The app’s title bar can be seen, along with the system’s reported accuracy. To the left, is the ability to search for a business. In the centre are columns for rendering the businesses’ reviews. On the right is the space where a review will be rendered.

The app component can be thought of as the ‘entry’ of the view. Every subsequent component is rendered within the main app component. This component’s state stores:

- *accuracy* - The system’s accuracy.
- *foundBusinesses* – An array containing the results of a search for businesses.
- *currentBusiness* – A reference to the business entry from the database for the current business.
- *goodReviews* – An array of good reviews for the current business.
- *badReviews* – An array of bad reviews for the current business.

Review Component

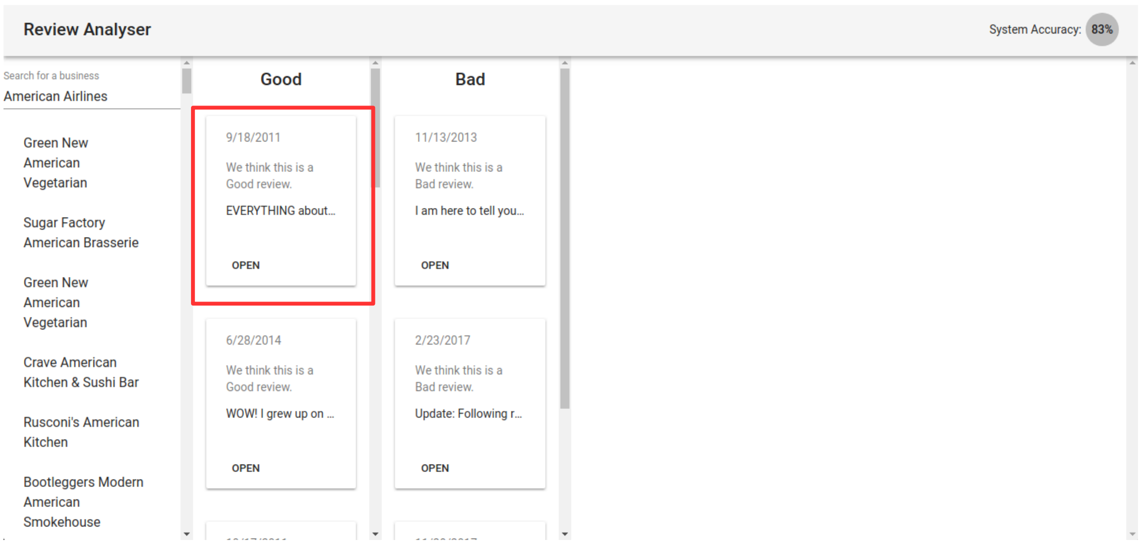
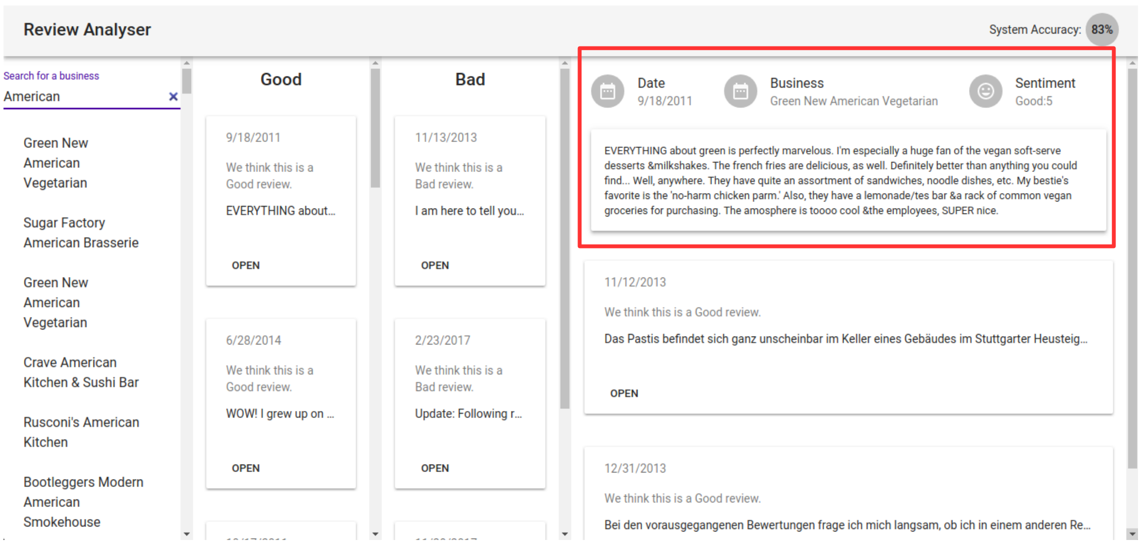

Review Analyser
System Accuracy: 83%
Search for a business
American Airlines
Green New American Vegetarian
Sugar Factory American Brasserie
Green New American Vegetarian
Crave American Kitchen & Sushi Bar
Rusconi's American Kitchen
Bootleggers Modern American Smokehouse
Good
Bad
9/18/2011
We think this is a Good review.
EVERYTHING about...
OPEN
11/13/2013
We think this is a Bad review.
I am here to tell you...
OPEN
6/28/2014
We think this is a Good review.
WOW! I grew up on ...
OPEN
2/23/2017
We think this is a Bad review.
Update: Following r...
OPEN

Figure 3.11 shows the review component highlighted in red.

The review component is a small, retangular component designed to show a brief overview of a review. It contains the date the review was made, it's predicted sentiment, and an excerpt of the review. Its open button will open a review detail to the right of it in the empty white space.

Review Detail Component


Review Analyser
System Accuracy: 83%
Search for a business
American
Green New American Vegetarian
Sugar Factory American Brasserie
Green New American Vegetarian
Crave American Kitchen & Sushi Bar
Rusconi's American Kitchen
Bootleggers Modern American Smokehouse
Good
Bad
9/18/2011
We think this is a Good review.
EVERYTHING about...
OPEN
11/13/2013
We think this is a Bad review.
I am here to tell you...
OPEN
6/28/2014
We think this is a Good review.
WOW! I grew up on ...
OPEN
2/23/2017
We think this is a Bad review.
Update: Following r...
OPEN
Date
9/18/2011
Business
Green New American Vegetarian
Sentiment
Good:5

EVERYTHING about green is perfectly marvelous. I'm especially a huge fan of the vegan soft-serve desserts &milkshakes. The french fries are delicious, as well. Definitely better than anything you could find... Well, anywhere. They have quite an assortment of sandwiches, noodle dishes, etc. My bestie's favorite is the 'no-harm chicken parm.' Also, they have a lemonade/tes bar &a rack of common vegan groceries for purchasing. The amosphere is toooo cool &the employees, SUPER nice.

11/12/2013

We think this is a Good review.

Das Pastis befindet sich ganz unscheinbar im Keller eines Gebäudes im Stuttgarter Heusteig...

OPEN

12/31/2013

We think this is a Good review.

Bei den vorausgegangenen Bewertungen frage ich mich langsam, ob ich in einem anderen Re...

Figure 3.12 shows the review detail component highlighted in red.

The review detail component shows a review in more detail. The full review text can now be read, along with the predicted sentiment good/bad next to the rating given on Yelp. Below the review detail are related reviews, inferred by our sentiment analysis application based on the review's text.

3.3.5 Controller

According to Burbeck, S. (1992), “the controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate.” For this application, the controller encompasses:

1. the possible interactions with the component,
2. how these interactions are translated to commands (AJAX requests to the server),
3. how these commands are handled by the server requests,
4. how the business logic forms the appropriate response for the server to send back to the client.

React Components

Each React component controls the user interaction with itself either imlicitly via React’s component life-cycle[1] due to changes in props or state, or explicitly via on-click events. For example, when a review’s ‘open’ button is pressed the following chain of events occur:

1. An AJAX POST request is sent to retrieve similar reviews via ‘/similar’ which is then stored in the review component’s state.
2. The review component then notifies its parent component ‘App’ that it should update the ‘Review Detail’ component via its ‘updateDetail’ method.
3. The ‘App’ component then re-renders itself implicitly due to its state being changed, redrawing only the necessary updates to the view.

RESTful API

As previously mentioned, the application’s view is updated by user interactions that trigger the controller to make requests to the application’s API. This API follows a RESTful architecture, meaning that all URLs only specify one resource, and HTTP verbs are used to perform appropriate requests. The following available API endpoints are as follows:

- */review* – used to query the database for reviews. This endpoint can return all reviews, it also accepts query parameters in the URL i.e. ‘GET /review?stars=5&limit=10’ will return 10 five-star reviews.
- */business* – used to query the database for business data. This is often used alongside the retrieval of a review, getting the business data for that particular review.
- */similar* – used to retrieve similar reviews based on the supplied text via the Yelp model.

1 React’s documentation explains the component life-cycle in greater detail https://reactjs.org/docs/react-component.html#the-component-lifecycle

Database Controller

The database controller is responsible for processing API requests. More specifically, the database controller provides methods that act as a layer between the API and the MongoDB database driver. The available controller methods are as follows:

- getSimilarReviews(review) => Returns an array of similar reviews.
- findReviews(filter) => Returns an array of reviews that match the supplied filter.
- getReviewIds(filter) => Returns an array of review IDs that match the supplied filter. As findReviews() can return a large amount of data, this method returns only the IDs.
- getAccuracy() => Returns the reported system accuracy.
- getBusiness(filter) => Returns an array of businesses that match the supplied filter.

These methods are used in combination with the above RESTful API endpoints to provide an easy to use, extensible API.

Interfacing with Python

In order to process requests for similar reviews, the application must ask the Yelp model to infer a vector of the review text. As this process is done within the Python environment, it is required to somehow interact with a Python shell from within the Node.js application.

Fortunately, Node provides a library called ‘child_process’ that enables spawning processes on child threads. This allows Node to keep its non-blocking architecture while allowing interaction between different programming environments. The Node and Python processes communicate via the standard streams (stdin/stdout). An event listener is attached to the stdout stream of the Python process.

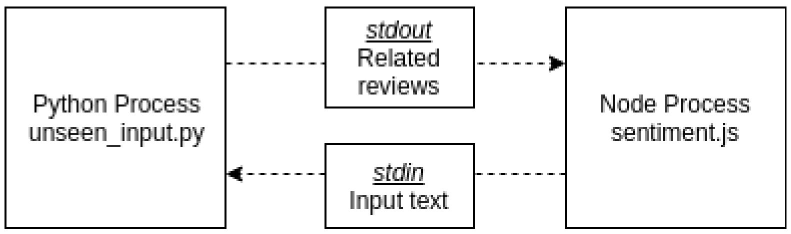


Figure 3.13 shows the communication protocol between the Python and Node processes.

4 Evaluation

This chapter of the report will discuss and evaluate the successes and failures of this project, including issues that arose during planning, research, and development. It will also discuss professional, ethical, and personal development planning concerns during the project.

4.1 Critical Evaluation

4.1.1 Project Scope

The original aims of this project were as follows:

a) Learn deep learning concepts and how these can be used for sentiment analysis.
b) Use TensorFlow and Python to create a basic sentiment analysis tool.
c) Research and evaluate the performance of existing solutions for sentiment analysis.
d) Evaluate my own sentiment analysis tool by comparing its performance with existing solutions.

Additionally, the following project deliverable was proposed:

> "Using TensorFlow I will develop a deep learning algorithm that will analyse large data sets of human language and interpret the sentiment behind them. I will devise an evaluation method that will compare the performance of my own algorithm compared to existing ones. Once I have the algorithm working and if time allows I will look at creating a 'suite' that others can use to leverage the power of sentiment analysis."

The following sections will critically evaluate the above project aims and deliverable by measuring the successes and failures of relevant areas of the project.

4.1.2 Computational Power

A large oversight during the development of this project was the availability of computational power to perform the training. As shown in section 3.1 of the development chapter, a lot of effort was placed in optimising the loading of large data sets. Fortunately, Gensim's "smart_open" utilitiy and Python's iterators managed to heavily optimise the training phase, which made it possible to mean-test to find the ideal parameters for training the Yelp Doc2Vec model.

4.1.3 Accuracy

The accuracy of the Yelp model is a good measure of the application's ability to correctly infer the sentiment behind a particular review (performance). Le, Q., & Mikolov, T. (2014) propose an evaluation method that compares the accuracy across various sentiment classifiers to discern the relative effectiveness of their model. To do this, they use the Stanford Sentiment Treebank Dataset and the IMDB Dataset[1]. As Yelp's dataset contains reasonably long review sentences, the comparison to the Treebank Dataset will be discarded, as the datasets should be comparable.

IMDB Accuracy Benchmark

In order to evaluate the accuracy of the Yelp Doc2Vec model, its error rate will be compared to that of a Doc2Vec model for the IMDB dataset. As both datasets are similarly constructed (both from review websites, similar in length, similar in context), it is presumed that their accuracies should be somewhat comparable.

The following experimental protocols are set by Mikolov et al:

- A window size of 10.
- Vector Dimensions of 400.
- 75,000 training reviews, equally split between good and bad.
- 25,000 unseen testing reviews, equally split between good and bad.

Model	**Error Rate**
BoW (Maas et al., 2011)	12.20 %
WRRBM (Dahl et al., 2012)	12.58%
MNB-uni (Wang & Manning, 2012)	16.45%
Paragraph Vector IMDB (Le & Mikolov, 2014)	**7.42%**
Paragraph Vector Yelp	**22.81%**

Figure 4. 1 shows a comparison of error rates for different sentiment classification methods.

Using a similarly sized corpus as Le & Mikolov (2014) resulted in an error size three times as large as theirs (22.81% vs 7.42%).

1 http://ai.Stanford.edu/ amaas/data/sentiment/index.html

Confusion Matrix

A confusion matrix helps to see the relative accuracy of different classes. Merely stating that the Yelp model has a 22.81% error rate doesn't provide insight as to where that accuracy lies in the model. Additional measures of model quality can be derived from the results of the confusion matrix: sensitivity (true-positive rate), specificity (true-negative rate), positive/negative predictive value (how reliable is the given good/bad result)[1]. For the Yelp model, predictions of a negative review were more reliable than predictions of a positive review (85% vs. 70%).

		Actual Sentiment	
		Good	*Bad*
Predicted Sentiment	*Good*	**8720**	3780
	Bad	1922	**10578**

Figure 4.2 shows a confusion matrix of the Yelp model.

Measure	**Value**
Sensitivity	0.8194
Specificity	0.7367
Positive Predictive Value	0.6976
Negative Predictive Value	0.8462
False Positive Rate	0.2633
False Discovery Rate	0.3024
False Negative Rate	0.1806
Accuracy	0.7719

Figure 4.3 shows derived measures of performance for the Yelp Model.

4.1.4 User Interface

In the project specification, it was proposed that the project deliverable would be a 'suite' of tools that websites such as Yelp could use to leverage the power of sentiment analysis for their business. The deliverable aimed to demonstrate the business value of such software by recreating the basic functionality of an online review website with the addition of sentiment analysis. All of the application features were fully implemented, suggesting that there was opportunity to expand the amount of features.

1 A full elaboration on classifier performance metrics is Powers, D. M. W. (2007). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation.

4.2 Personal Reflection

4.2.1 Time Management

For a project of this size, time management would be paramount to the success of the project. There were many challenging topics to understand and translate into business requirements. In hindsight, while doing background research was essential; too much time was spent at the beginning researching broad areas rather than attempting to implement relevant ideas early on. This was felt when starting the development of the deliverable, progress was initially slow; full development was initially attempted right away but a lack of full understanding meant that experimental development was revisited in an attempt to solidify concepts.

4.2.2 Software Limitations

There were several limitations of the software used for this project:

- Manually implementing Doc2Vec in Tensorflow & Python during experimental development presented red-herrings with limitations on the amount of data it could be tested with. Tensorflow was inefficient at training, taking large amounts of time to complete. This lead to confusion when previously working code appeared to stop working when introduced to the full Yelp data set. This motivated the search for a better library, which resulted in the discovery of Gensim.
- Even though Gensim solved the issue of large data sets, Python was still a concern with its memory usage while training, often crashing during the process. Preparing the training computer was temperamental and in hindsight would have benefited from more dedicated hardware.
- One of the shortfalls of Gensim however, was its incompatibility with Yelp’s data set format. Expecting a large text file of reviews separated by new-lines was not possible with Yelp’s dataset, as it also contained valuable metadata that needed to be saved to prove the viability of the deliverable. This led to the development of custom ‘LabeledLineSentence’ classes.
- Gensim’s documentation was overall substandard. It lacked depth, especially in important places such as defining its parameters. It was cumbersome to relate these parameters back to those in the Doc2Vec paper by Mikolov et al. (2014). Additionally, online example were often outdated, further complicating development.

4.2.3 Mathematical Complexity

The perceived complexity of the problem domain was originally greatly underestimated. In hindsight, a recap of relevant statistical methods would have proved useful when researching the Doc2Vec model and developing an appropriate classifier. Fortunately, plenty of resources were readily available that helped assist understanding the underlying mathematical concepts.

4.3 Future Work

4.3.1 Improving Accuracy

While an accuracy of 77% was achieved with a training set of 75,000 Yelp reviews, the accuracy certainly has room for improvement. One suggestion for improving accuracy would be to analyse and pre-process the data-set before training. Taking out noise in the reviews such as punctuation may offer additional gains in accuracy. Additionally, a supervised approach to the training data could be used. Similarly to Stanford's Treebank data-set, have humans correctly label the training data into good and bad categories rather than assuming that all reviews with a star rating of <= 2 are bad and >=4 are good. Realistically, having a 75,000 strong data set being labelled by hand would take a lot of time unless there was a coordinated approach, but not impossible, as Stanford demonstrates.

4.3.2 Granular Sentiment Analysis

Rather than just classifying reviews into two categories – good and bad – a good addition would be the ability to predict more granularly. For example, a 1-5 star prediction, or classifying reviews based on what the review is describing – such as food quality, service quality, etc. Le & Mikolov (2014) demonstrated that this is already possible. In their evaluation method of their classifier, they propose a 5-way fine-grained classification task, sorting reviews into {Very Negative, Negative, Neutral, Positive, Very Positive} classes.

Conclusion

This project aimed to create a sentiment analysis tool that provided valuable insight into businesses by analysing online reviews with machine learning algorithms. While the analysis of reviews wasn't as fine-grained as originally desired, this project proved the viability of such a tool by showing that reviews can be automatically categorised by their overall sentiment. This was applied to a real-world data set and implemented into a prototype review application, again showing that it can be used in the real world.

Bibliography

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

Burbeck, S. (1992). Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). Retrieved from https://web.archive.org/web/20120729161926/http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method, (2), 1–5. https://doi.org/10.1162/jmlr.2003.3.4-5.951

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

Le, Q., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. Retrieved from https://arxiv.org/pdf/1405.4053.pdf

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

Powers, D. M. W. (2007). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Retrieved from http://www.flinders.edu.au/science_engineering/fms/School-CSEM/publications/tech_reps-research_artfcts/TRRA_2007.pdf

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

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

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

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

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

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

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

Van Der Maaten, L. J. P., & Hinton, G. E. (2008). Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research, 9, 2579–2605. https://doi.org/10.1007/s10479-011-0841-3

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

Appendix

A – Project Specification

Student:	**Ben Ewen**
Date:	**20/10/17**
Supervisor:	**Jing Wang**
Degree Course:	**Computer Science**
Title of Project:	**Using natural language processing and deep learning to predict sentiment in online reviews**

Elaboration

Natural language processing (NLP) aims to allow computers to process human language in a meaningful way such as responding to questions, converting data into sentences that can be understood by humans, or sentiment analysis. Additionally, as deep learning algorithms have matured, the ability to have a computer analyse human language has become more autonomous and relies less on supervision by a human.

Sentiment analysis, in essence, is trying to understand the sentiment behind a sentence. For example the sentence: "We went to the cinema, it was affordable but the movie was rubbish!" can be easily interpreted by a human. We know that the price was good, but the movie wasn't great. This is valuable information to the cinema, and sentiment analysis would be able to automate the analysis of these reviews.

This project will look at the various ways sentiment analysis can be performed via deep learning algorithms.

Project Aims

- Learn natural language processing concepts in relation to sentiment analysis
- Learn deep learning concepts and how these can be used for sentiment analysis
- Create a sentiment analysis tool to categorise online reviews
- Research and evaluate the performance of existing solutions for sentiment analysis
- Evaluate my own sentiment analysis tool by comparing its performance with existing solutions

Project deliverable(s)

I will develop a deep learning algorithm that will analyse large data sets of human language and interpret the sentiment behind them. I will devise an evaluation method that will compare the performance of my own algorithm compared to existing ones. This algorithm will then be used to create a prototype sentiment analysis tool that can be used by online review businesses to categorise their reviews by sentiment.

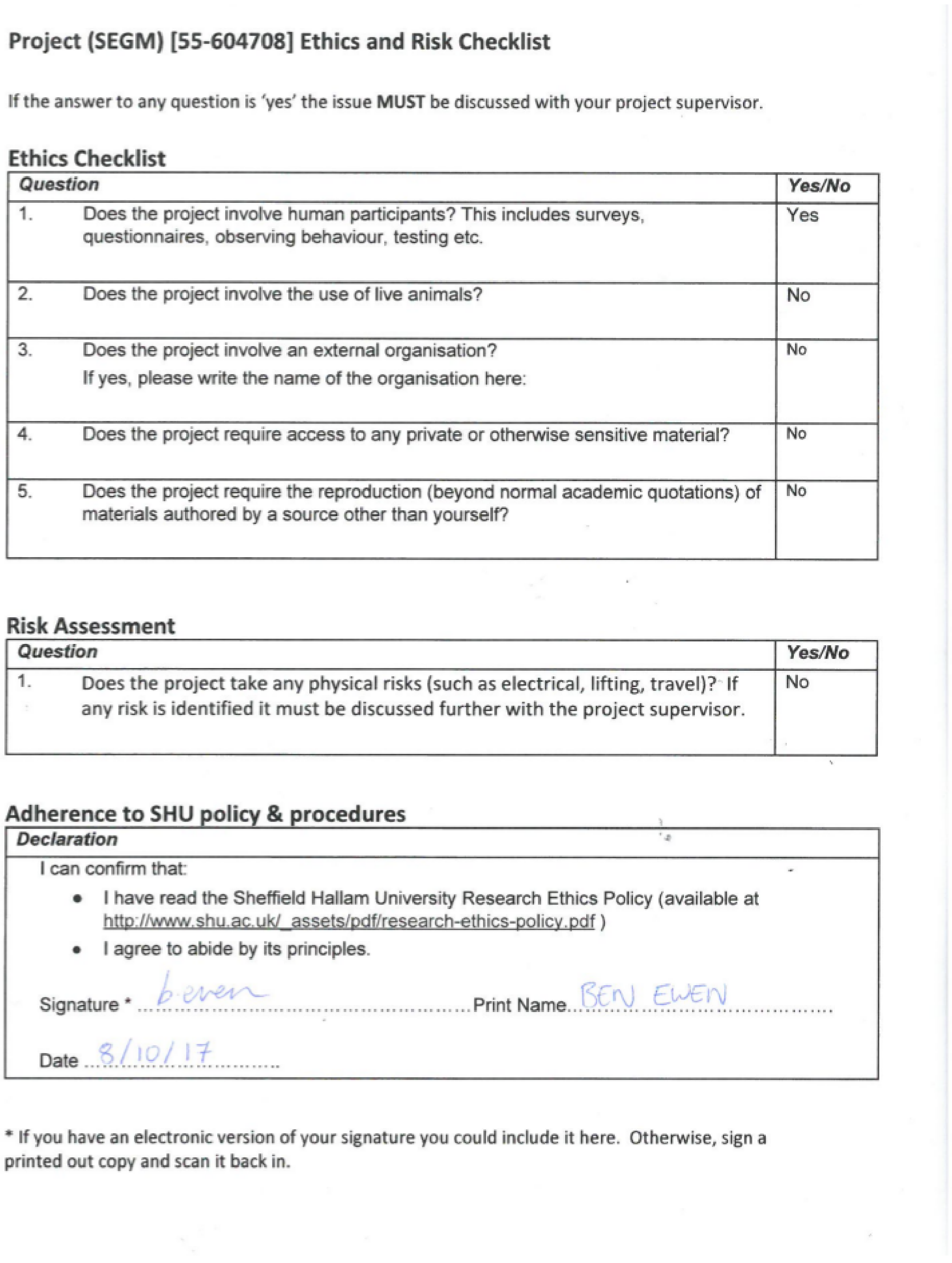
Action plan

Task	Deadline
Find a Project Supervisor	06/10/17
Complete Project Specification and Ethics Form	20/10/17
Background Research • Natural Language Processing ○ Word Embeddings ▪ Word2Vec • Deep Learning ○ Logistic Regression • Types of Neural Networks ○ Convolutional Neural Networks ○ Recursive Neural Networks ○ Recurrent Neural Networks & LSTM • Existing products for sentiment analysis • Traditional NLP algorithms	10/11/17 (Overall)
Information/Literature Review	01/12/17
Produce Deliverable • Experimental Development • Development of Learning algorithm • Development of front-end system	 14/12/17 01/02/18 01/03/18
Provisional Contents Page	09/02/18
Draft Critical Evaluation	09/03/18
Sections of a Draft Report	09/03/18
Evaluation • Accuracy • Personal Reflection • Future work	 14/03/18 01/04/18 01/04/18
Submit Body of Project to TurnItIn	11/04/18
Submit Project	12/04/18

Ethics

Based on the discussion with my project supervisor, this project has a low risk in terms of unethical practices. I will be analysing openly available data – none of the data used will be personally identifiable.

B – Ethics Form



Project (SEGM) [55-604708] Ethics and Risk Checklist

If the answer to any question is 'yes' the issue **MUST** be discussed with your project supervisor.

Ethics Checklist

	Question	***Yes/No***
1.	Does the project involve human participants? This includes surveys, questionnaires, observing behaviour, testing etc.	Yes
2.	Does the project involve the use of live animals?	No
3.	Does the project involve an external organisation? If yes, please write the name of the organisation here:	No
4.	Does the project require access to any private or otherwise sensitive material?	No
5.	Does the project require the reproduction (beyond normal academic quotations) of materials authored by a source other than yourself?	No

Risk Assessment

	Question	***Yes/No***
1.	Does the project take any physical risks (such as electrical, lifting, travel)? If any risk is identified it must be discussed further with the project supervisor.	No

Adherence to SHU policy & procedures

Declaration

I can confirm that:

- I have read the Sheffield Hallam University Research Ethics Policy (available at http://www.shu.ac.uk/_assets/pdf/research-ethics-policy.pdf)
- I agree to abide by its principles.

Signature * b.ewen Print Name BEN EWEN

Date 8/10/17

* If you have an electronic version of your signature you could include it here. Otherwise, sign a printed out copy and scan it back in.

C – Negative Sampling Mathematical Proof

Goldberg, Y., & Levy, O. (2014) explain in detail the mathematics behind negative-sampling:

Consider the probability $p(c \vee w;\theta)$ where *w* is a word in the corpus and where *c* is the context in which it resides: $c(w_i) = w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k}$ (i.e. the surrounding words exclusive of *w* where *k* is the specified window size). Additionally, consider input parameters θ where the goal is to maximise the probability.

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(c|w;\theta)$$

Equation 3.1 shows the objective of maximising the probability of c given w under parameters θ for all w, c in the set D where D is the set of all word and context pairs.

To parameterise equation 3.1, the conditional probability of $p(c \vee w;\theta)$ can be modelled using soft-max, a generalisation of the logistic function. Where v_c and v_w are vectors representing *c* and *w* respectively, and where *C* is the set of all available contexts. Parameters v_c, v_w should be set so that the product in equation 3.1 is maximised.

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}$$

Equation 3.2 shows the probability $p(c \vee w;\theta)$ *modelled as soft-max.*

This approach does not consider the computational complexity of calculating the similarity for all other contexts. Instead, it is better to only consider a few random extra contexts, reducing the time required to train the model. Mikolov et al. (2013) consider a negative sampling approach to do this.

Based on the soft-max approach, the negative sampling method instead tries to maximise a different objective. Consider a pair *(w, c)* of word and context, the probability that it came from the training data $p(D=1 \vee w, c;\theta)$ and the probability that it didn't come from the training data $1 - p(D=1 \vee w, c;\theta)$. As before, assume that there are parameters θ that control the distribution of these probabilities.

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(D=1|w,c;\theta)$$

Equation 3.3 shows the objective of maximising the probability that (w, c) did in fact come from the training data *D.*

The probability of *(w, c)* originating from the training data *D* can again be modelled using soft-max:

$$p(D=1|w,c;\theta) = \frac{1}{1+e^{-v_c \cdot v_w}}$$

Equation 3.4 shows the probability $p(D=1 \vee w, c; \theta)$ *modelled as soft-max.*

Taking the log (enables faster computation, the sum of log probabilities can be performed rather than the product of the probabilities) of the derived equation from equations 3.3 and 3.4 produces the following:

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log \frac{1}{1+e^{-v_c \cdot v_w}}$$

Equation 3.5 shows equation 3.3 as a sum of the log probabilities, including the soft-max equivalence as defined in equation 3.4.

Parameters θ are again v_c, v_w. To achieve the objective in equation 3.5, parameters are set so that $p(D=1 \vee w, c; \theta) = 1$ where $v_c = v_w$ for every pair *(c, w)*. To prevent all the vectors having the same value of 1, certain *(c, w)* combinations can be disallowed. This can be done by presenting the model with *(c, w)* pairs where its probability must be low. This is achieved by generating the set *D'* of random *(c, w)* pairs where the pairs are not in the set *D*. The objective now becomes:

$$\arg\max_{\theta} \sum_{(w,c)\in D} \log \frac{1}{1+e^{-v_c \cdot v_w}} + \sum_{(w,c)\in D'} \log(\frac{1}{1+e^{v_c \cdot v_w}})$$

Equation 3.6 shows the final objective for negative-sampling.