Performance Monitoring Based on Customer Feedback

Author:
Stefan Lauren

Supervised by:
Dr. Hisham Al Assam

Project Report submitted for the degree of
MSc Applied Data Science (Degree Apprenticeship)
to the School of Computing in the University of Buckingham

March 2021

Executive Summary

Analysing customer feedback is one way for a company to understand customers' need and also, the strengths and weaknesses of the products/services. This task becomes arduous when the number of feedback grows into a large amount. Hence, it is important to automate the process with the help of natural language processing (NLP) techniques. Moreover, the growing use of deep learning method in NLP might improve the performance. Sentiment classification helps company to quickly grasp the overall condition of the company. The model trained using word embedding and convolutional neural network architecture has comparable performance to the baseline model. Text summarisation is able to give summaries of the feedback into just several representative sentences. Using extractive summarisation method, the produced summaries qualities are approved from the domain experts' point of view. All metrics are gathered together in a reporting dashboard using Tableau for business use.

Keywords: *Customer feedback, NLP, Deep Learning, Sentiment Classification, Text Summarisation*

Acknowledgements

I have received a load of supports, guidance, and assistance in finishing this project. First and foremost, I would like to thank my supervisors, Dr. Hisham Al-Assam of the School of Computing, University of Buckingham. He was always there for me since the beginning and his guidance motivates me to strive for better goals and results. He helped me better understand the topic and the knowledge needed for this project, and corrected me on my mistakes. Every feedback he gave was valuable on the final completion of this project.

I would also acknowledge many of the University of Buckingham's staffs, who either directly or indirectly supported me on this project. The School of Computing administration staffs who helped me deal with administrative stuff. The work based dissertation and project's team who always answered the questions and supported me since the topic allocation.

I would also like to thank my family who although is far away from here, but still supported me during my study and the completion of this project. The encouragement they gave me always kept me from being distracted and be more focused on the project and tasks ahead. Last but not the least, for my line manager in Avado, who always supported me to finish the project without leaving any regret.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A business measures its performance through various means, such as sales target, delivered milestones, and customer retention rate. In the emergence of Data Science, businesses start to look at another aspect of their customers. Many successful businesses improve themselves by addressing customer feedback as a key performance indicator to make business decisions. Customer feedback are usually collected regularly through different methods, for example, customer surveys and ratings website. They are usually in the form of a feedback rating, summary/head, and body as can be seen in Figure 1.1.

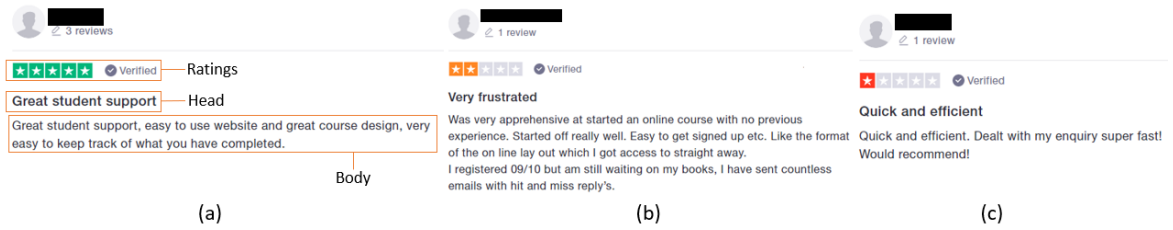


Figure 1.1: Examples of Customer Feedback

While reading through the feedback, many insights can be derived. As an example, the number of feedback indicates the number of customers who are either satisfied or disappointed with the business' products or services. The number of ratings shows the overall sentiment of the feedback. Positive feedback can be used to measure business performance even further, while negative feedback can be used to improve mentioned products/services. Proper feedback analysis can even help the business lays out future data-driven business roadmap.

However, the number of feedback increases daily and when there are a huge number of them, it becomes an arduous task to read and analyse everything. Therefore, data mining should be performed to automate the task. The other concerning issue is the inconsistencies between the feedback rating with its head and body. One basic feature of customer feedback is the sentiment or polarity, which can be classified as positive, negative, and/or neutral. It is quite common to have positive sentiment

feedback with a low rating, or vice versa due to various reasons, such as human input error. Figure 1.1(c) shows an example of an inconsistent feedback.

As mentioned, sentiment analysis is a common task applied to feedback data. There are many techniques to do sentiment classification on text data [22, 16, 28]. Although there are basic sentiment classification methods such as dictionary mapping and weighting; recently, sentiment classification using deep learning methods have gained popularity. Understanding the overall sentiment of customer feedback is vital for other performance indicators to form shape.

Other feature that can be extracted from customer feedback is the summarised feedback [18]. Knowing the average sentiment of all feedback is one thing, understanding the content of those feedback is another important thing. Extracting the summary of those feedback can help businesses point out specific part of their products or services faster. Summaries may also be analysed to see the change of trends within certain time interval.

There are many other features that can be extracted from feedback data. However, even after gathering those features, part of a data mining process is to be able to provide a suitable form of deployment. Simple static visualisations with charts and graphs are not enough for business use. Business reporting tools such as PowerBI or Tableau is suitably used for deployment of this project. This phase concludes the project as performance indicators from customer feedback are compiled and presented back to the business.

1.2 Business Problem Formulation

From a business perspective, there are business oriented objectives and requirements. As mentioned in the previous section, there are ways to measure business performance through customer feedback. Thus, in this project, there are several questions that are needed to be answered.

1. What kind of feedback data should be collected and how to automate the collection process?

   Feedback data can be found from many sources. However, each source offers different kind of data. Hence, it is important to find the right data source and more importantly, to have a seamless and automatic way to collect these data.

2. What are the performance metrics that can be derived from customer feedback?

   Feedback data are a type of unstructured text data and thus, the metrics that can be extracted are often not very obvious. Therefore, it is necessary to formulate the performance metrics that can be extracted from feedback data.

3. What methods should be used to get the metrics?

   As data science techniques are not limited to only one method, there are many ways that the metrics can be extracted. It is vital to use suitable methods to deal with the unique feedback data.

4. How to present the result for business purpose?

   Metrics that are successfully retrieved using data science methods need to be compiled into a business presentable way such that they are easily understood by people from various part of the business.

1.3 Aims and Objectives

While Section 1.2 defines business problems, the aims and objectives for this project concern about the data mining process from the initial process of data collection to deployment.

1. To automate the process of collecting feedback data and compile them into a dataset.

   Automating web scraping process efficiently saves time and resources.

2. To define a set of performance metrics to be used.

   There can be various metrics, such as customer satisfaction, general sentiment, and content overview.

3. To have robust data cleaning and preparation techniques for unstructured data to address various issues, such as imbalanced data and missing values.

   Some basic data problems include imbalanced data, empty data, etc. Feedback data also need to be prepared, such as how to handle the rating numbers and how to relate the header and body of the feedback.

4. To investigate natural language processing techniques, such as sentiment classification and text summarisation in depth to gain meaningful insights from feedback data.

   How each technique processes feedback data to suit the purpose is an important aspect to properly extract the insights.

5. To compare traditional machine learning techniques with deep learning techniques on handling unstructured data.

   There are ways to construct the architecture of a model with deep learning method. Moreover, it is not a guarantee that a deep learning method works better than the traditional one.

6. To create an intuitive and all-inclusive report dashboard to show performance comparison between EdTech companies.

   Feedback data, features, and metrics must be compiled together in a user friendly way.

1.4 Scope Limitation

Scope of the project is limited to make sure the focuses are well-defined in the context of the limited time available to finish it. With that being said, below are few scope limitations for this project.

1. The business scenario for this project is limited to Avado Learning company and its competitors.

   Avado is a company within education sector or EdTech. Naturally, the competitors of Avado are also those working in the same sector. The list of competitors can be found in Appendix A.

2. Feedback data is limited to ratings website Trustpilot[1].

   Although there are many feedback data sources, Trustpilot is picked as this is the main feedback data source, which Avado uses.

3. The focus performance metrics for this project are sentiment classification and feedback summarisation.

   Other metrics that are naturally found in the data may be added in the deployment phase, but not as strict requirements for this project.

4. Final reporting tool is Tableau[2].

   Tableau is chosen because this tool has been used by Avado as the standard reporting tool. Moreover, it is proven to be a strong visualisation tool that is also instinctively easy to understand.

1.5 Methodologies Overview

Data collection is done through web scraping method. The scraping script gathers data of customer feedback from different learning companies. Data analysis, cleaning, and preparation are conducted using Python and utilising various libraries. For the sentiment classification, machine learning and deep learning libraries are used. For the pretrained word embeddings, they are downloaded directly from their website. The feedback summarisation algorithms are both written from scratch and also by using Python library. Figure 1.2 shows the overview framework of the project.

1.6 Outcomes

Many different algorithms and architectures to build a sentiment classifier show promising result. However, sentiment classifier trained using a balanced dataset between positive and negative feedback has better performance. It is able to predict

[1] https://uk.trustpilot.com/
[2] https://www.tableau.com/en-gb

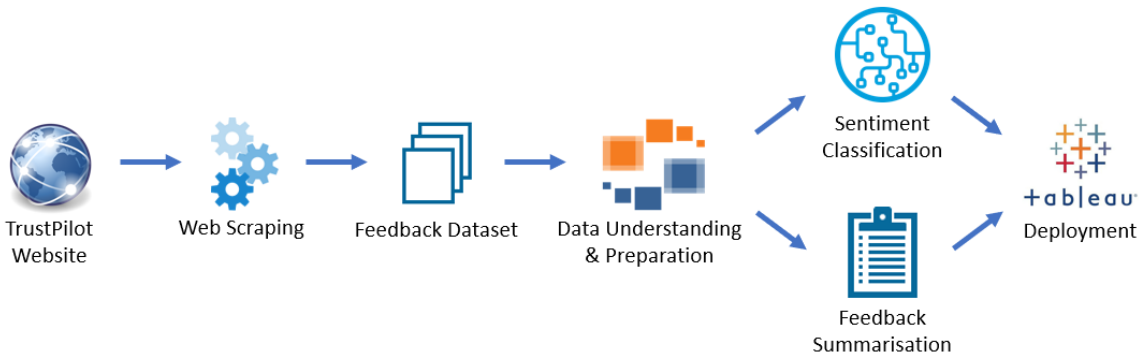


Figure 1.2: Project Overview

more negative feedback from feedback with high ratings. Summaries generated from the extractive summarisation method are representative enough. The difference between the algorithm used makes the summaries fit for different use cases. They can represent the general trend of the feedback, but can also be a unique experienced customer.

1.7 Report Structure

This report is structured as follows. Chapter 1 gives motivation, aims, and overall outcomes of the project. Chapter 2 provides necessary background study and related work to the project. Chapter 3 details the feedback data and shows data understanding and preparation. Chapter 4 lists the core tasks of the work through modelling. Chapter 5 explains the evaluation and deployment of the result achieved from the work. Chapter 6 concludes the work and gives several future directions.

Chapter 2

Background

This chapter is dedicated to set the ground for the project. Sentiment analysis, the main focus of this project, is briefly explained in Section 2.1. In order to understand the state-of-the-art techniques of sentiment analysis, several concepts must be explained first. Section 2.2 explains natural language processing, which is the foundation of sentiment analysis. Word embedding, as explained in Section 2.3, shows how machines process text data. Section 2.4 unfolds the convolutional neural network used in this project. Section 2.5 explains the concept of text summarisation. Lastly, Section 2.6 mentions several related works to this project.

2.1 Sentiment Analysis

Sentiment analysis is one of the most common classification task in natural language processing. It receives an input text and outputs the sentiment/polarity of the text, usually either positive, negative, and/or neutral, but it can also be something else, such as the number of stars in a rating [22, 16, 28]. Many researches and techniques had been conducted in this particular area, from using classic machine learning approach [28, 27, 32] or the current state-of-the-art deep learning method [6, 10, 36, 31].

In businesses perspective, sentiment analysis is very useful to help them understand the sentiment of their customers towards the brands/products/services. It becomes important when it is not trivial anymore to extract the sentiment manually, i.e., by reading every feedback. First, it takes time to understand the sentiment of each feedback, and second, it is difficult to measure the overall sentiment. An easier way to check the sentiment is to count the number of ratings. However, this method is not always reliable because number of ratings might not reflect the true sentiment of the feedback.

2.2 Natural Language Processing (NLP)

Natural language processing is a set of computing techniques to analyse natural text data, both written and spoken, in order to complete the goals of applications as

human-like as possible [21, 5, 4]. Hence, the ultimate goal is have the system to understand natural language, execute the requested task and produce feedback in natural language as well. NLP is also an interdisciplinary field with a combination of proper computer science, artificial intelligence, machine learning, cognitive science, and clearly, linguistics.

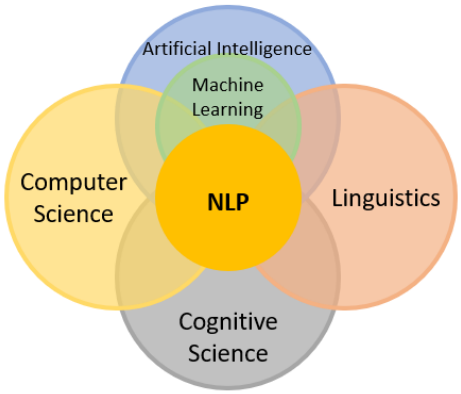


Figure 2.1: Natural Language Processing

Natural language is complex. There are cases of homonyms (one word with multiple meanings), illogical sense (such as 'I haven't slept for 10 days', sense of direction/subject position, etc. There are several building blocks for natural language: words/morphology, structure/syntax, semantics, discourse/relationship, and pragmatics/intents.

There are many applications of natural language processing, such as sentiment analysis, fake news detection, offensive language detection and categorisation, question answering, machine translation, text summarisation, dialog systems, and many more. These applications are made possible because of the long history of NLP. The work has begun since the 1950's, with the foundational work of automata and information theory. The approach used at that time until 1990's was rule-based method to hand-craft rules to model linguistic intuitions.

Starting from the 2000's, the method of modelling linguistic tasks has shifted into a more sophisticated corpus-based method with emphasis on statistical methods. The features are extracted from a corpus to be used on a machine learning/statistical algorithm to build models. With the advancement of machine learning technology, the current state-of-the-art of NLP method is to use deep learning or deep neural network to learn the features and build models from given data.

2.2.1 Deep Learning for NLP

As the current state-of-the-art of NLP, deep learning provides a very flexible framework to represent linguistic information in both supervised and unsupervised fashions [7, 8, 9]. In many machine learning tasks, NLP in particular, feature design phase is very difficult and not always effective due to the nature of data being overspecified, incomplete, etc.

One common way to represent features is to use one-hot vectors. The resulting vectors are very sparse with with dimensionality as each feature is its own dimension and completely independent from each other. Hence, the number of dimensionality of the vector is proportional to the number of distinct feature. In most NLP applications, there is a huge amount of distinct features and it makes learning very computationally expensive. Moreover, deep learning does not perform well with high dimensional, sparse features.

It is also possible to represent features more densely with each feature is a d-dimensional vector. This results on similar features will have similar vectors, hence, better generalisation. A popular method of dense vectors representation is word embeddings. More details of word embeddings can be found in Section 2.3. This method enables model to learn representations from raw input or words. On top of that, it can also learn multiple levels features.

2.3 Word Embedding

In most NLP applications, the basic input starts with words, although it can also be a character in certain languages, such as Chinese; or it can be the smaller units of words, for example, lemmas, root, and sub-word unit. There are several ways to represent these words to be a suitable input type for machine learning:

- Bag of words/TF-IDF

  Uses raw words to create one hot vector or frequency vector [39]. As a result, the vector is very sparse.Moreover, this representation of words cannot model relatedness between words, such as dog and puppy. Despite of the weakness of this method, bag of words model is still popular as it is good for small corpus or a very specific domain context. Figure 2.2 shows an example of the one hot vector from two sentences:

  1. The dog caught the stick.
  2. The puppy sticks to the floor.

	...	dog	catch	floor	puppy	stick	the	to	...
1.	0	1	1	0	0	1	1	0	0
2.	0	0	0	1	1	1	1	1	0

Figure 2.2: Vector representation using bag of words method

  The frequency vector of the two sentences is: {'dog':1}, {'catch':1}, {'floor':1}, {'puppy':1}, {'stick':2}, {'the':4}, {'to:1}. It is assumed that words with higher significance will also appear more in the sentences. in contrast, another approach assumes that high frequency words may not be able to provide much information. This method is called term frequency-inverse document frequency

(TF-IDF) [30]. The significance or weight of a word increases if it appears more in the same document, but decreases if it also appears in many other documents; mathematically formulated as:

$$weight_w = TF_{w \in d_i} \times \log(\frac{N}{N_{w \in d}})$$

with w as the word, d as a collection of documents, $TF_{w \in d_i}$ as the frequency of word w in document d_i, N as the number of documents, and $N_{w \in d}$ as the number of documents containing word w.

- Co-occurrence vectors

  Defines a word by words it co-occurs with in corpus. For example, using binary approach and given a corpus with 5 sentences:

  1. Dogs eat food.
  2. Puppies eat food.
  3. Cats eat mice.
  4. Kittens eat mice.
  5. Trees are green.

  The co-occurrence matrix can be seen in Figure 2.3 below.

	cat	eat	mouse	kitten	dog	food	puppy	tree	are	green
cat	1	1	1	0	0	0	0	0	0	0
eat	1	1	1	1	1	1	1	0	0	0
mouse	1	1	1	1	0	0	0	0	0	0
kitten	0	1	1	1	0	0	0	0	0	0
dog	0	1	0	0	1	1	0	0	0	0
food	0	1	0	0	1	1	1	0	0	0
puppy	0	1	0	0	0	1	1	0	0	0
tree	0	0	0	0	0	0	0	1	1	1
are	0	0	0	0	0	0	0	1	1	1
green	0	0	0	0	0	0	0	1	1	1

Figure 2.3: Example of Co-occurrence Matrix

Representing words by their context has some advantages. It can remove stop-words, use the frequency as feature, calculate weight of words, and calculate the similarity between words. However, the co-occurrence matrix is still very large and sparse. The problem can be alleviated by reducing the dimensionality of the matrix, although the process is computationally expensive.

- Word embedding

  Uses vector space models to represent words in continuous vector space where semantically similar words are mapped together nearby each other. Hence, instead of counting co-occurrences, it predicts context words in context. This is computationally efficient and it is possible to add new words to the model. In the current state-of-the-art, there are few word embedding models:

  1. Word2Vec [25, 26]

     Word2Vec provides two types of word embedding:

     (a) Skip-gram model

     It predicts context words from the target word. Statistically, the goal is to find the best representation to predict context words w_{t+j} given a word w_t by maximising the log probability of function:

     $$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t;\theta)$$

     where c is the context window or window size. Maximising the function means to find the parameter θ so that the probability is maximised through the entire corpus. The probability p is defined using softmax function:

     $$p(w_{t+j}|w_t)=\frac{\exp\left(v{\prime}_{w_{t+j}}^{T}v_{w_t}\right)}{\sum_{w\prime=1}^{W}\exp\left(v{\prime}_{w\prime}^{T}v_{w_t}\right)}$$

     where v_w and v'_w are the target and context vector representation of word w.

     In the tutorial given in [23, 24], the skip-gram model consists of 3 layers neural network: input layer, hidden layer, and output layer. By removing the output layer after training the model, the word embedding of a certain word is the weights vector output of the hidden layer. It can be inferred as the probabilities of context words to appear if given a target word. For example, if the target word is 'United', then the probabilities of context words 'Kingdom' and 'States' are much higher than other words such as 'England' or 'New York'.

     (b) Continuous bag of words (CBOW) model

     It predicts target word from context words. Architecturally, it is similar to the skip-gram model with 3 layers network and same dimension of hidden and output layer with softmax function in the end. The training is also similar with the model is trained with the pair of context-target word. The difference is in the input layer and how to calculate the weights in the hidden layer.

  2. Global Vectors for Word Representation (GloVe) [29]

     Different from Word2Vec models, which use local contexts to train, GloVe is trained on global co-occurrence matrix from a corpus, and hence, produces more reliable output. However, using global corpus also makes training computationally expensive for the first time. Moreover, it does not perform well in determining analogy, which Word2Vec model performs better.

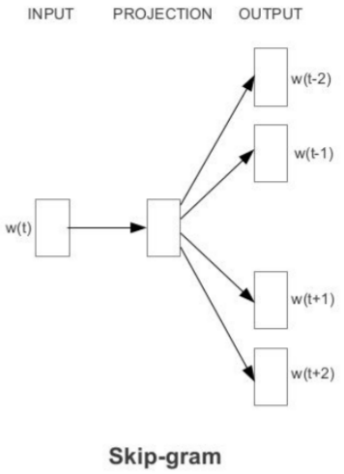


Figure 2.4: Skip-gram model with $w(t)$ as target word, $w(t-n), w(t+n)(n>1)$ as context words [25]

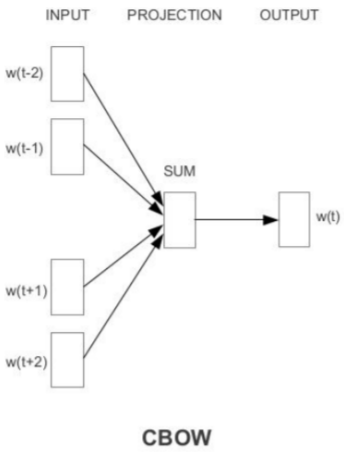


Figure 2.5: Continuous bag of words model with $w(t)$ as target word, $w(t-n), w(t+n)(n>1)$ as context words [25]

The output of the model is the correlation probabilities between words, however, some words correlate with unrelated words due to the training corpus. To solve this problem, GloVe uses both probability and ratio of two target words. Assume target words t_1, t_2 and some context words $c_1, c_2, \ldots, c_n$. Calculating the probability of each target word with each context word yields, respectively, $P(c_1|t_1), P(c_1|t_2), P(c_2|t_1), P(c_2|t_2), \ldots,$ $P(c_n|t_1), P(c_n|t_2)$. Finally, by calculating the ratio between two probabilities of the same context word $P(c_i|t_1)/P(c_i|t_2)$, if the ratio is very high, then c_i is related to t_1; if the ratio is very small, then c_i is related to t_2; if the ratio is close to 1, then c_i is not related to any of the target words.

2.4 Convolutional Neural Network in NLP

Convolutional neural network (CNN) is a type of deep learning network, which is very successful in image recognition, for example, as an object detector [20]. It is possible as the networks combine three architectural ideas to compute the shift and distortion invariance: local receptive fields, shared weights/weight replication, and spatial/temporal subsampling [19]. For a typical image processing task, such as digits recognition, Figure 2.6 shows a common convolutional neural networks architecture.

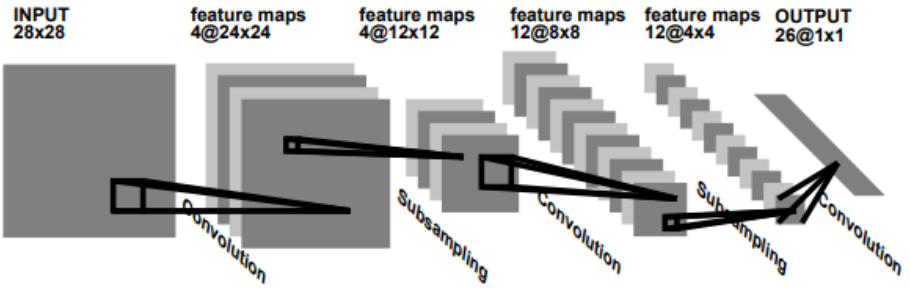


Figure 2.6: Example of CNN for image processing [20]

The input in this case is images of digits with a specific pixel size. To transform the input into the first feature maps layer, a convolution operation is applied to every set of pixels from the input layer. The convolution operation is analogous to filtering operation, with a filter kernel size $h \times w \times c$, where h and w are the height and width of the filter and c is the colour channels. The operation takes a set of input pixels the size of filter size and transforms it into a new pixel by multiplying each pixel from the input with the weight and sum them, i.e., $Convolution(x, y) = x * y = \Sigma_{k=0}^{n-1} x_k y_k$, where x is the input, y is the filter, and n is the number of pixels of kernel size $h \times w$.

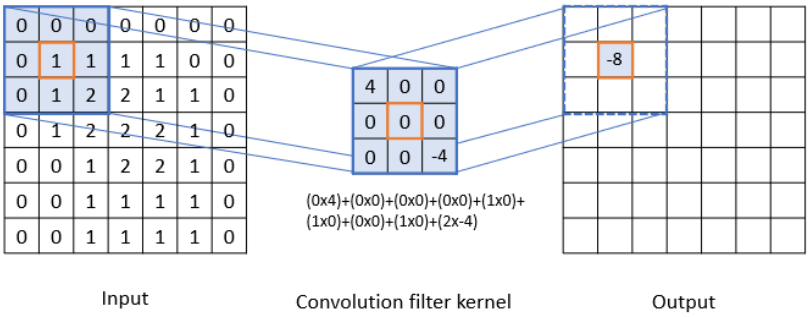


Figure 2.7: Convolution operation

Filter can also be seen as a local receptive field, in which neurons can extract features, such as edges and corners in images, and hence, the result of the operation is a feature map. A convolutional layer is usually composed of several feature maps so that multiple features can be captured from the same set of pixels of the filter. The subsampling/downsampling operation, also called pooling operation, is applied to reduce the size of feature maps in order to reduce the sensitivity of the output

to various modifications/invariance, such as image distortion or shifts. The pooling operation takes a set of pixels of size $p \times p$ and performs average or maximum calculation to yield a new output pixel.

Despite of its success on image processing, CNN has also begun to be gradually used to tackle NLP tasks [2, 14, 33, 38, 6]. The main motivation is the complexity of processing sentences with different sizes, making the traditional neural network does not work well [19, 11]. Analogous to CNN for image processing, the goal is to learn smaller and simpler features of the text. In a sense, it is similar to segmenting sentences into words, and further into characters or phonemes, and other features, among many other techniques [17, 38, 13]. An example architecture can be seen in Figure 2.8.

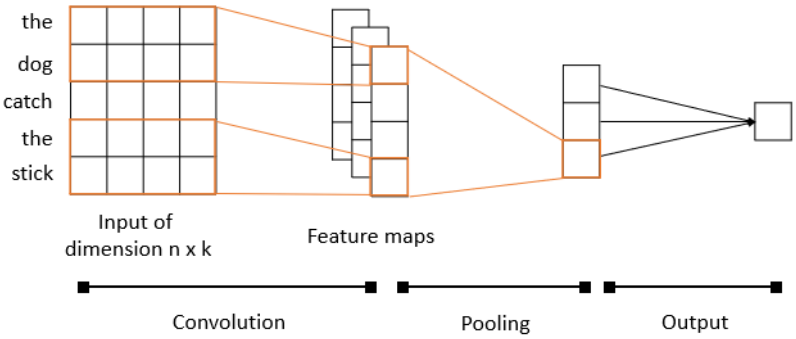


Figure 2.8: CNN for NLP

As the input, words from a sentence are transformed into word vector, e.g., by word embedding. The result is $x_i \in \mathbb{R}^k$ where k is the dimension of the word vector of the i-th word in the sentence. Hence, a sentence with length n can be written as $x_{1:n} = x_1 \oplus x_2 \oplus \cdots \oplus x_n$ with concatenation operator $\oplus$. The convolution operation also uses a filter $w \in \mathbb{R}^{h \times k}$, which is applied to an h number of words with k dimension window to get a feature. In Figure 2.8, $n = 5, k = 4, h = 2$. This filter is applied repeatedly to every window of words in the sentence. For example, a feature p_i generated from words window $x_{i:(i+h-1)}$ can be written as $p_i = g(w.x_{i:(i+h-1)} + b)$ where g is the non-linear activation function and b is the bias. The result from the convolution operation is a feature map from this particular filter $p = [p_1, p_2, \ldots, p_{n-h+1}]$, $p \in \mathbb{R}^{n-h+1}$.

The pooling phase comes next with max pooling operation $\hat{p} = max(p)$. In NLP tasks, pooling is necessary to capture only the most important feature with highest value for each feature map so that the model can deal with variable sentence lengths. The processes from input, convolution layer, and pooling operation are usually done with multiple filters of different window sizes in order to extract more features. As an example, in Figure 2.8 there are three feature maps that come from three different filters, resulting in pooling layer with three dimension. Finally, the output can be calculated using softmax function. Similar to the other deep learning techniques, interpretability can be an issue. Although theoretically the filters function to capture different semantic patterns, it is not clear on which specific features are captured.

2.5 Text Summarisation

Automatic text summarisation is an NLP task with a goal to shorten a long piece of text into a coherent summary containing only the main points. It is very important that the summaries have to be accurate, concise, and fluent. Some advantages of having accurate summaries of a huge size of text are time saving, fast information retrieval and processing, and compact information, among many others. There are mainly two types of text summarisation techniques: extractive and abstractive summarisations [18].

Extractive text summarisation technique filters and selects words/sentences from the text and combines them together as a summary [1, 15]. In other words, there are no changes on the text; and the filters are usually based on some metrics, for example frequency, sentiment, and length. On the contrary, abstractive text summarisation involves paraphrasing and shortening the text to create new phrases/sentences it deems important from the source text [34, 3]. Conceptually, abstractive summarisation is better than the extractive summarisation. However, there are still many challenges to have a good abstractive summarisation model.

There are three main problems in text summarisation, especially the extractive summarisation method. First, how to find the most important words from the document. Next, how to calculate scores for each sentence in the document based on the information about the important words. Finally, how to rank and choose the most important sentences to form a summary.

One simple method to address these issue is to use weighted word occurrence. The method assumes that the more frequent a word is used throughout the document, the more important it is. Then, each sentence is scored by summing the weighted word occurrence value for each word in the sentence. The summary can be formed by selecting few top sentences with highest weighted word occurrence value. There are may other methods to handle summarisation issue, such as TF-IDF and word embedding method. Each of them has different approaches with their own strengths and weaknesses, but the core steps are similar.

2.6 Related Works

Yang et al. [35] compared sentiment analysis on user reviews (review body) and the review summaries (review heads) using various learning architectures. Sometimes, the reviews contain the sentiment, while the summaries do not, and vice versa. Nevertheless, they found out that the sentiment of a review and its corresponding summary are complementary to each other as their correlation is quiet subtle. The proposed attention architecture to model deeper relation between reviews and summaries showed that it performed well to analyse sentiment in user written summaries rather than generated summaries from summarisation algorithm.

Jain and Pamula [12] conducted a systematic literature review on machine learning applications for sentiment analysis of online customer reviews. Focusing in the

domain of hospitality and tourism industries, they studied the applicability, scope, and usefulness of machine learning for customer sentiment analysis. The goal for the study was to find the gaps in the sentiment analysis of customer online reviews domain and formulate future directions.

Shrestha and Nasoz [31] evaluated the compatibility of user reviews and the corresponding polarity ratings. Taking Amazon online reviews as the studied dataset, it is found that inconsistencies exist where the user reviews do not reflect the number of rating given. Recurrent neural network was used to train sentiment analysis task with input reviews text and an additional input of product information, which was proven to be a strong feature in this case. The application of the model was to detect reviews-ratings mismatches and inform the users for clarification.

Zhang et al. [36] conducted a thorough survey study on sentiment analysis using deep learning methods. As deep learning becomes the state-of-the-art for prediction results, along with the growing number of user reviews in various format, it is critical to cater the needs of both individual and businesses to understand the sentiment in a simple and quick manner. Sentiment analysis tasks are mainly done in three level of granularity: document level, sentence level, and aspect level. The document level analysis usually classify the sentiment as positive or negative in the assumption that documents always contain opinion toward certain entity. On the other hand, a sentence can be opinionated or not, and hence can be categorised as positive, negative, or neutral. Lastly, the aspect level sentiment analysis extracts the sentiment towards a specific entity. Sentiment analysis task can also be expanded into emotion analysis, sarcasm detection, or multilingual sentiment analysis, among many others.

Gupta et al. [10] performed multi-class classification on customer feedback into either comment, request, bug, complaint, meaningless, and/or undetermined. Moreover, the model was trained using feedback from different languages as well, namely English, French, Japanese, and Spanish. To help with the complex feature engineering and to build general model across languages, the deep CNN and recurrent neural network were used. Overall, the resulting networks performed quite well, but can still be improved further, for example by using pre-trained word embedding. Few issues need to be addresses as well, such as ambiguous feedback, missing entity, and short text.

Wang et al. [33] tried to model short texts for text categorisation using semantic clustering and CNN to deal with data sparsity and ambiguity often found in such texts. The architecture first clusters the word embeddings, resulting in multiple scales of semantic units and are believed to contain useful information for short texts. The clusters are rearranged using n-gram approach to select only strong semantic units above a certain threshold. These semantic units are then combined as an input to the convolutional layer, which is followed by max pooling operation to get the classification output. The architecture is comparable to the other benchmark models.

Zhang et al. [38, 37] used CNN for text classification with character level processing. Few large datasets were prepared to ensure the convolution architecture could perform at maximum. They claimed that information from words is not needed for

convolutional networks, i.e., CNN does not need knowledge of syntactic or semantic structure of a language. This character level convolutional networks perform well in comparison with other traditional methods, such as bag of words, n-grams, and deep learning methods, such as word level CNN and recurrent neural network. Another advantage is that the networks can recognise misspelling and emoticons quite naturally.

Johnson and Zhang [13] explored the effective use of word order in text data for text categorisation using convolution neural networks. While usually the words are converted into word vectors using pre-trained embedding method, this particular work used raw words as the input and let the algorithm to learn the embedding itself. This was done by having the first convolution layer with filter window size equals to one. They argued that in some NLP tasks, especially sentiment classification, word order is important. The result shows that the models outperformed conventional method, such as bag-of-words.

Santos and Gatti [6] proposed new architecture of convolutional neural network to tackle issues of classifying sentiment for short texts, such as Twitter messages. A one sentence message has only limited contextual information and hence, information on both character and word level are equally important. One way to achieve this is to use pre-trained word embedding model from a large corpus. The proposed architecture has two convolution layers to cope with any number/length of words and sentences. The first layer converts words into both word and character vectors. The subsequent layers extract features of different complexity. They achieved sentiment prediction accuracy of 85.7% for movie reviews dataset and 86.4% for Twitter dataset.

Kim and Hovy [16] presented a system to analyse sentiment of a word and the combined sentiment of words in a sentence from opinion text. The goal was to match up opinions to the topic and calculate the sentiments while pointing the people who hold the opinions. They experimented with various conventional methods, such as lookup table using WordNet and entity tagger. Although the processes involved some manual works, the experiments gave a promising result despite the simple model.

Pang et al. [28] used movie reviews and classified them as either positive or negative. Traditional machine learning methods are used, i.e., Naive Bayer, maximum entropy classification, and support vector machine. However, the machine learning model still cannot outperform the classic topic-based sentiment analysis technique.

Chapter 3

Data Collection, Understanding and Preparation

This chapter is dedicated to explore the beginning processes of data mining, which are all about data. Data collection explains the method used to get the feedback data. Data understanding shows the nature of the data, challenges, and relationships between features. Lastly, data preparation details the processes to make the data ready for the next step. The code for data collection can be found in Appendix C and the code for data understanding and preparation can be found in Appendix D.

3.1 Data Collection

The data is collected using web scraping technique. It is a common method to extract data by visiting webpages programmatically and tracing the HTML structure to get the intended information. In this project, target webpage URLs are the Trustpilot webpage for every company listed in Appendix A. Figure 3.1 shows an example of Trustpilot webpage. It can be seen that there are many information on the webpage not related to the customer feedback. Python library BeautifulSoup[1] is used to help the scraping process, which iteratively visits each page of every company's webpage and extract the information. The result of this process is a collection of CSV files containing customer feedback data of each corresponding company.

3.2 Data Understanding

3.2.1 Data Description

Feedback data are known to have some challenges. First of all, there are inconsistencies between the number of ratings, the head, and the body of the feedback. Ratings number might not reflect the head and the body, and vice versa. Moreover, the head and the body can even be in different context. The next problem with feedback data is the extreme length. A feedback can be very short, i.e., contains only 1 word, or

[1] https://www.crummy.com/software/BeautifulSoup/

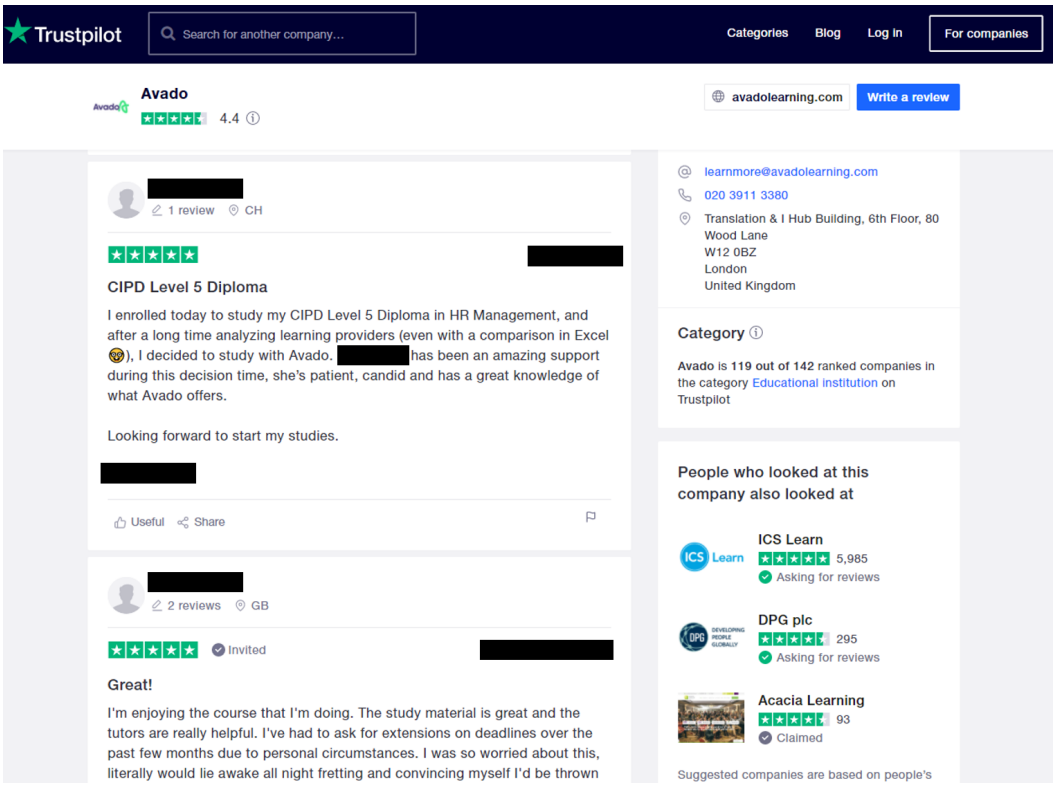


Figure 3.1: Trustpilot Webpage

very lengthy, i.e., several paragraph/pages. This might suggest that the longer feedback are more credible. However, short feedback prone to be more consistent with the ratings number.

Another problem is that customer feedback are subjective and context dependent. Each customer has their own standard of rating number and also, different customers portray the same subject differently. The terms and phrases often used in the feedback depends on the bigger context and hence, makes it difficult to judge. Most of these problems are not addressed in this project; they become some of the future work directions. However, the inconsistencies can be addressed using sentiment classification so that there is another metric to compare and inconsistencies can be detected.

The collected feedback data consist of 17575 rows (observations) and 17 columns (features) of data from February 1, 2013 to January 26, 2021. It is formatted as a CSV file and Figure 3.2 shows the first and second row of the dataset. Table 3.1 describes each features on the dataset.

Feature	Data Type	Description	Example
Unnamed: 0	Integer	Auto-generated index from scraping process	0
@type	String	The type of data/information in the review HTML block	'Review'

Feature	Data Type	Description	Example
datePublished	String	The date and time the review was published/submitted	'2021-01-15 T13:50:34+00:00'
headline	String	The head/summary of the review	'Great!'
inLanguage	String	The detected language used in the review	'en'
reviewBody	String	The body of the review	'I'm enjoying the course that I'm doing...'
author_@type	String	The type of data/information in the author HTML block	'Person'
author_image	String	The profile picture image of the author of the review, given as a URL to the image	'https://user-images.trustpilot.com/5ff541c7b33...'
author_name	String	The name of the author	'Stefan Lauren'
author_url	String	The URL to the author Trustpilot profile	'https://uk.trustpilot.com/users/600170bc9cd3e5...'
itemReviewed_@type	String	The type of data/information in the target company HTML block	'Thing'
itemReviewed_name	String	The name of the target company	'Avado'
publisher_@type	String	he type of data/information in the publisher HTML block	'Organization'
publisher_name	String	The name of the publisher	'Trustpilot'
publisher_sameAs	String	The URL to the publisher	'https://uk.trustpilot.com'
reviewRating_@type	String	The type of data/information in the ratings HTML block,	'Rating'
reviewRating_ratingValue	Integer	The number of ratings given to the reviews with range between 1 and 5 inclusive	5

Table 3.1: Features Description

For columns with '@type' in the name, the value of each of the column is the same. For example, column 'author_@type' has only one value 'Person' for the whole dataset. Other than this, the columns 'publisher_name' and 'publisher_sameAs' have the same value for the whole dataset as well because in this case, there is only one publisher, which is 'Trustpilot'. The dataset fulfills the minimum requirements needed for this project with focus on columns 'headline', 'reviewBody', and 'reviewRating_ratingValue', which make up a complete customer feedback; and also, 'itemReviewed_name' to indicate the target company. Other features may also be useful for future developments, but they are outside the focus of this project.

Unnamed: 0	@type	datePublished	headline	inLanguage	reviewBody	author_@type	author_image	author_name
0	Review	2021-01-15T13:50:34+00:00	When speaking to one advisors on...	en	When speaking to one of the advisors on the ph...	Person	NaN	Phoebe
1	Review	2021-01-15T10:45:08+00:00	CIPD Level 5 Diploma	en	I enrolled today to study my CIPD Level 5 Dipl...	Person	NaN	Melissa Guillen

author_url	itemReviewed_@type	itemReviewed_name	publisher_@type	publisher_name	publisher_sameAs	reviewRating_@type	reviewRating_ratingValue
https://uk.trustpilot.com/users/5fa08296e7f841...	Thing	Avado	Organization	Trustpilot	https://uk.trustpilot.com	Rating	3
https://uk.trustpilot.com/users/600170bc9cd3e5...	Thing	Avado	Organization	Trustpilot	https://uk.trustpilot.com	Rating	5

Figure 3.2: Examples of the Data in Dataset

3.2.2 Data Exploration

To understand more about the data, some explorations are conducted. First of all, it is interesting to see the distribution of the number of ratings within the dataset. Figure 3.3 shows a count plot for each rating. At least 80 percents of the feedback have five stars rating; and it seems like that the number of ratings 2 and 3 are not popular choices. It may indicate that people tend to go to the extreme choice on both positive and negative feedback.

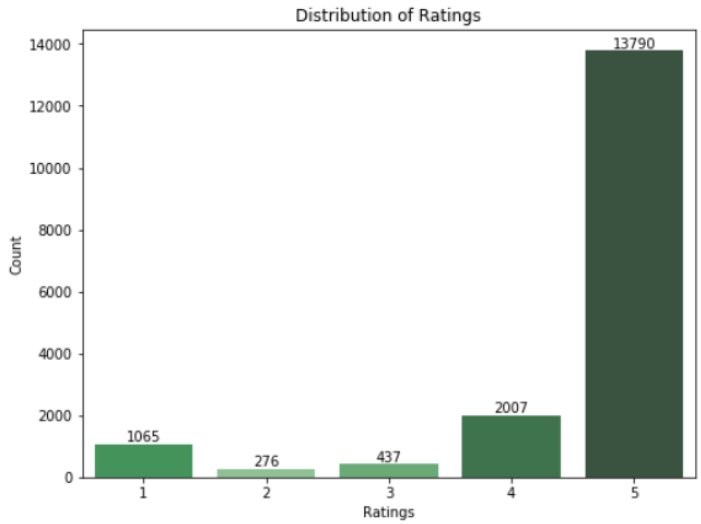


Figure 3.3: Distribution of Ratings

There are 17 companies in the dataset and therefore, a breakdown of number of ratings for each company can also give insights. Figure 3.4 shows the distribution of ratings for each company. The trends are quite similar with the total distribution. Most of the companies have 5 stars feedback as the most common. It can also be inferred that only a part of the companies utilise Trustpilot as a feedback platform, for example Avado, ICS Learn, and learndirect Limited. For companies such as BPP, Deloitte, and FutureLearn, the number of feedback they have are very small such that they do not show in the plots.

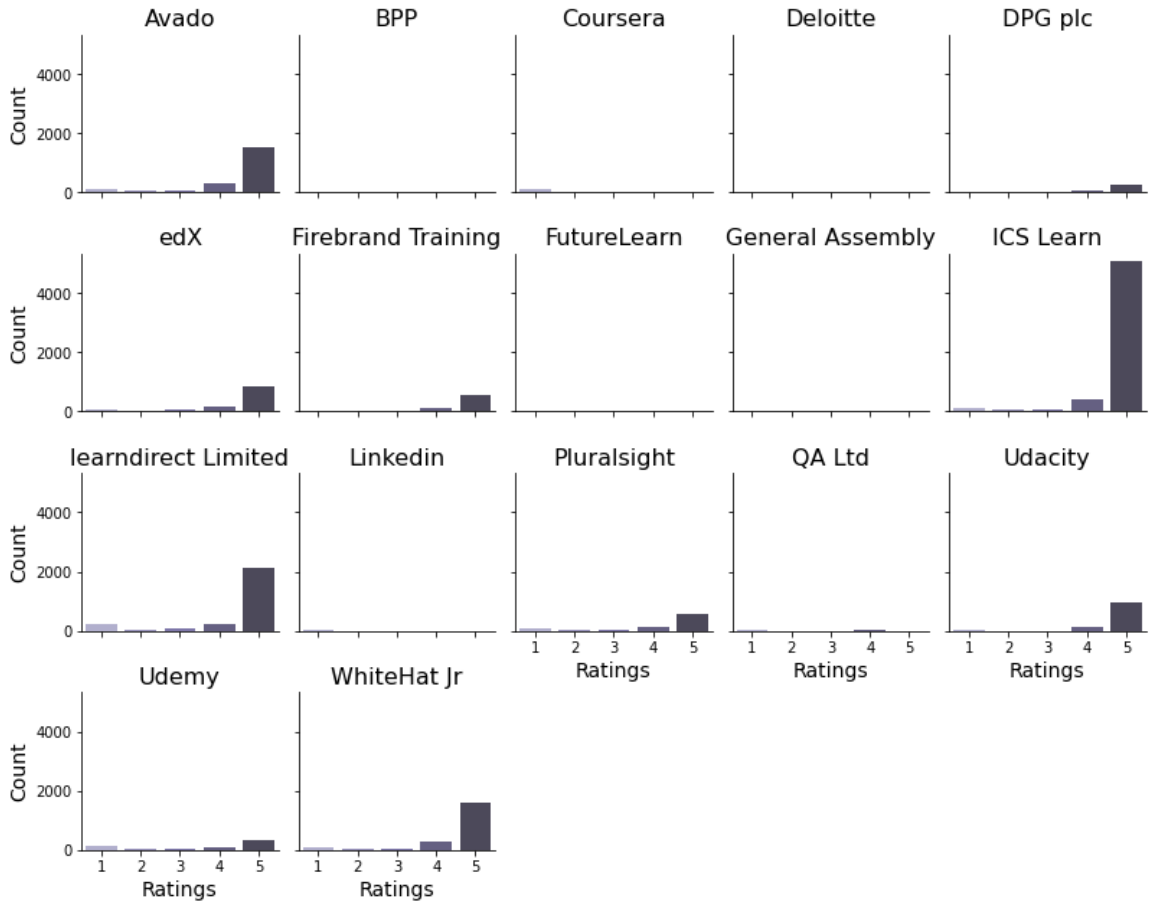


Figure 3.4: Distribution of Ratings per Company

The relationship between the number of ratings and the length of the feedback is also interesting. The length of a feedback is defined as the number of words in the head and the body of the feedback, combined together. The reasoning behind this is that the head of the feedback typically consists of only a few words and thus, quite trivial in this comparison. Figure 3.5(a) shows the length of each feedback on every number of rating. It is quite obvious that there are several outliers, in which cases the lengths are very large. However, the trend is pretty clear as well; the lower rating numbers are more likely to have longer feedback. This claim is backed up by Figure 3.5(b), which shows the average feedback length per number of ratings.

The explorations above are further explained by the correlation between the number of ratings, head length, body length, and total length. As can be seen in Figure 3.6, the correlation between body length and total length is 1, while the correlation between head length and total length is close to 0; strengthening the assumption that feedback head does not really contribute to the length. The correlation value between the ratings and the length of feedback is negative, which means longer feedback tends to have lower rating; this also fits with the information on Figure 3.5.

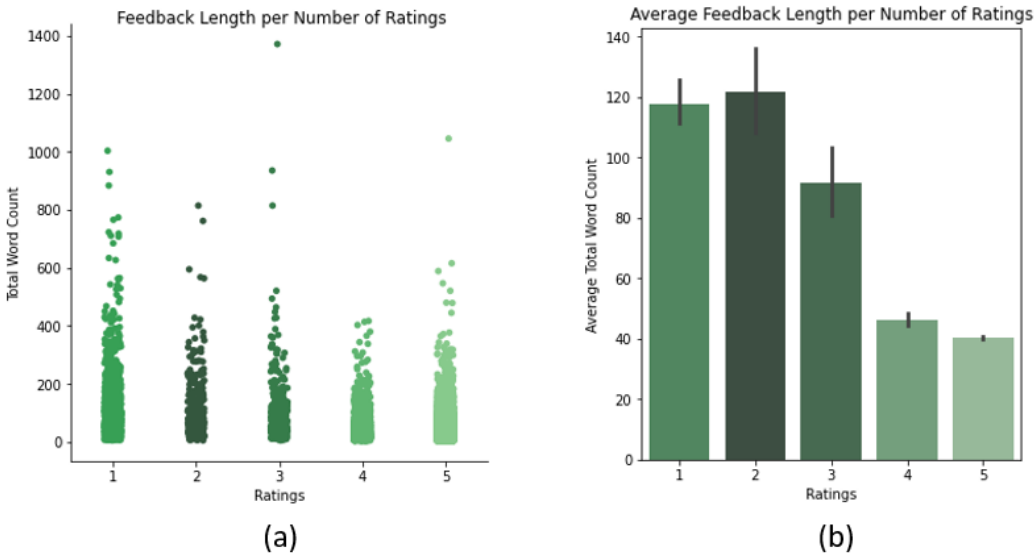


Figure 3.5: (Average) Feedback Length per Number of Ratings

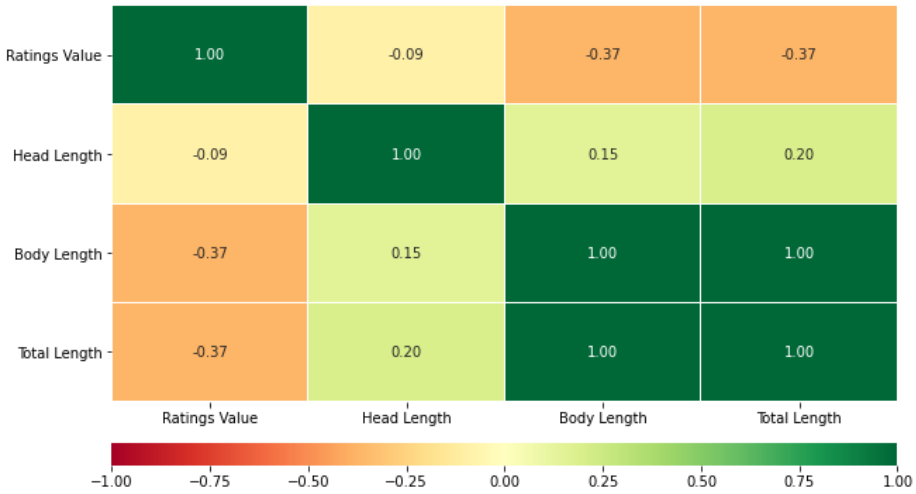


Figure 3.6: Correlations: Ratings Value, Head Length, Body Length, Total Length

3.2.3 Data Quality Verification

As part of understanding the data, the quality of the data need to be verified as well as this will affect the actions applied in the data preparation step. First of all, Figure 3.7 shows the total amount of missing values for each feature in the dataset. It can be seen that there are only two features with missing values: author_image and author_name, both of which can be ignored as the scope for this project does not need to use any of these two features. There is also no issue in data format as each feature has been assigned with appropriate data type. Last but not the least, the data input quality must be checked. For the ratings value, it is confirmed that the values range between 1 to 5 only, inclusive, as can be seen in Figure 3.3. For the head and body of the feedback, after manually checking the dataset for any discrepancies, there are several issues. Firstly, it seems that the headline is automatically filled

with several words from the body if the reviewer did not fill in manually; marked by three dots (...) at the end of the headline. This should not be a problem as the context of the feedback can usually be inferred from the first sentence. Next, there are some seemingly random feedback such as 'Xggv', emoticons such as ':)', shorten words such as 'Fab' from fabulous, and typos such as 'consise' (should be 'concise'). Although some of them will be removed in the data cleaning process, these issues will be a part of the model training such that the model can recognise the problems as features as well.

	Total	Percent
author_image	15110	85.974395
author_name	1	0.005690
total_word_count	0	0.000000
body_word_count	0	0.000000
@type	0	0.000000
datePublished	0	0.000000
headline	0	0.000000
inLanguage	0	0.000000
reviewBody	0	0.000000
author_@type	0	0.000000
author_url	0	0.000000
itemReviewed_@type	0	0.000000
itemReviewed_name	0	0.000000
publisher_@type	0	0.000000
publisher_name	0	0.000000
publisher_sameAs	0	0.000000
reviewRating_@type	0	0.000000
reviewRating_ratingValue	0	0.000000
head_word_count	0	0.000000
Unnamed: 0	0	0.000000

Figure 3.7: Missing Values for Each Feature

3.3 Data Preparation

3.3.1 Data Selection

To begin the preparation, it is a good practice to exclude unwanted columns first as it will make it easier to prepare the important data. The selected columns are chosen simply because they are needed for the project. They are 'headline', 'reviewBody', and 'reviewRating_ratingValue', which make the whole feedback structure. Then, 'itemReviewed_name' is needed to give comparison for business insights purpose. Lastly, 'datePublished' is also included for indexing/sorting purpose. The excluded columns might be useful, but they can be explored further as a future work direction.

3.3.2 Data Cleaning

Data cleaning is a process that takes the headline and body as the input and output the cleaned headline and body texts. There are several steps involved in cleaning the data:

1. Removing unwanted characters, numbers, and symbols

   The removal technique used is a simple regular expression matching after each sentence is split into words. This step will help alleviate problems that may occur during the training and evaluation processes.

2. Making the entire text lowercase

   Converting the head and body feedback texts into lowercase helps in the process of the later pre-processing step and also, the model training phase.

3. Removing stopwords from the text

   Stopwords are words that do not have special meanings. The list of stopwords used in this project is taken from the NLTK[2] corpusThe removal process uses a simple map matching technique for each word in the head and body of the feedback.

Below are a couple of input and output examples from the above cleaning process:

Input 1: *I enrolled today to study my CIPD Level 5 Diploma in HR Management, and after a long time analyzing learning providers (even with a comparison in Excel:)), I decided to study with Avado. Cheryl Ball has been an amazing support during this decision time, she's patient, candid and has a great knowledge of what Avado offers. \n\nLooking forward to start my studies. \n\nMelissa G.*

Output 1: *enrolled today study cipd level diploma hr management long time analyzing learning providers even comparison excel decided study avado cheryl ball amazing support decision time patient candid great knowledge avado offers looking forward start studies melissa g*

Input 2: *A brilliant blend of different learning styles including group work, online materials and live webinars. I really enjoyed this, and it was great to collaborate and learn with my fellow 'squares'*

Output 2: *brilliant blend different learning styles including group work online materials live webinars really enjoyed great collaborate learn fellow squares*

Although the outputs show promising improvements from the original feedback text, it is still not enough to be effectively used in machine learning training. There are many words which are not in their root form. For example, some are affected by the grammatical rule, for example, 'enrolled' (past tense) and 'analyzing' (present continuous tense). Other examples include the plural form, such as 'providers' and 'styles'. Therefore, another cleaning step is added, which is lemmatizing the words.

[2]https://www.nltk.org/

A lemmatizer receives a word as the input, searches the corpus, and outputs the root word of the input. Below are output examples from the lemmatizing process:

Lemmatized Output 1: *enrol today study cipd level diploma hr management long time analyze learn provider even comparison excel decide study avado cheryl ball amaze support decision time patient candid great knowledge avado offer look forward start study melissa g*

Lemmatized Output 2: *brilliant blend different learn style include group work online material live webinars really enjoy great collaborate learn fellow square*

3.3.3 Data Construction

Data construction can also be called feature engineering, where new features are produced from the existing features. Not only that, constructing the data can also be adding new rows on the data, merging multiple data sources, etc. The head length, body length, and total length (word count) from Figure 3.5 and Figure 3.6 are examples of feature engineering as these three new features do not exist in the original dataset.

Some other new features are produced as products of the data cleaning process, i.e., the cleaned and lemmatized head and body of the feedback. As inferred from the data understanding, the head length is comparatively shorter than the body length. Hence, it might be useful to combine the head and body into one longer feedback text. Lastly, a new rating scale is also introduced with only zero and one as the value. If the original rating of the feedback is one, two, or three; then it will be given zero as the new rating. On the other hand, a rating of one is given to the feedback with original rating four or five. This feature is produced for experiments purpose in Chapter 4.

The structure of the final prepared dataset can be seen in Figure 3.8.

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head_word_count	body_word_count	total_word_count
0	2021-01-15T13:50:34+00:00	When speaking to one advisors on...	When speaking to one of the advisors on the ph...	Avado	3	6	153	159
1	2021-01-15T10:45:08+00:00	CIPD Level 5 Diploma	I enrolled today to study my CIPD Level 5 Dipl...	Avado	5	4	64	68

headline_clean	reviewBody_clean	lemmatized_headline	lemmatized_reviewBody	lemmatized_feedback	overall_rating
speaking one advisors	speaking one advisors phone felt answers given...	speak one advisor	speak one advisor phone felt answer give quest...	speak one advisor speak one advisor phone felt...	0
cipd level diploma	enrolled today study cipd level diploma hr man...	cipd level diploma	enrol today study cipd level diploma hr manage...	cipd level diploma enrol today study cipd leve...	1

Figure 3.8: First Two Rows of the Final Dataset

Chapter 4

Implementation

At this point, the basic dataset has been prepared (Figure 3.8) and created for the purpose of two main tasks: sentiment classification and feedback summarisation. The experiment plan for both tasks can be seen in Section 4.1.

4.1 Experiment Set Up

The experiment set up is meant to give clearer goals for the sentiment classification and feedback summarisation by setting down each subtask.

4.1.1 Set Up for Sentiment Classication

For the sentiment classification, it is a binary classes classification task, which uses the column *overall_rating* as the class labels with values 0 and 1 only. A multi-classes classification can also be done using column *reviewRating_ratingValue*, but this is for future work. With the binary classes of feedback data, it is expected to have imbalanced number of data for each class. Hence, for each of the task the performance comparisons between the imbalanced and balanced (through resampling) datasets are given.

As for the learning algorithms, they include:

1. Logistic Regression

   This is a simple, yet powerful machine learning classification algorithm built for binary classes, however, it can also be extended for multi classes. The logistic regression model serves as a baseline model for conventional machine learning method.

2. Neural Network

   A basic feed forward neural network with an intended simple architecture of one layer perceptron to serve as a baseline model for deep learning method.

3. Convolutional Neural Network (CNN)

The main algorithm to focus on. The architecture for CNN will follow the proposed architecture by many research papers and it has been the standard architecture so far.

The performance of a sentiment classifier is also affected by how the training input is represented. There are several representation:

1. Bag-of-words and TF-IDF

   In this project, the bag-of-words and TF-IDF representations are limited to only 1-gram/unigram with the assumption that feedback texts are short in general.

2. Hardcoded word embedding

   Takes the vocabulary of the training dataset, assigns index for each word in the vocabulary, and converts the dataset into indexed representation.

3. Pre-trained word embedding

   The pre-trained word embeddings used in this project are GloVe and Word2Vec.

4.1.2 Set Up for Feedback Summarisation

First of all, the feedback summarisation uses the extractive summarisation method. Even with this method, it is already interesting enough to discover how combined short feedback texts are summarised. There are three extractive techniques to be used in this project:

1. Weighted word occurrence

   This is a very simple and straightforward technique that assumes the more frequent words, the more important they are. It first counts the occurrence for each word in the document, i.e., the combined feedback texts; takes the highest occurrence and divides each word's occurrence with the highest occurrence to get the weighted word occurrence values. The sentence score is calculated by summing the occurrence value for each word in the sentence.

2. TF-IDF

   The difference with the weighted word occurrence method should be that in TF-IDF, highly frequent words (excluding stopwords) that appear in many sentences are penalised. This technique assumes that a single occurring word is probably more important than a common word that appears everywhere in the document. Hence, the scoring algorithm is different (using TD-IDF method) which can be calculated as:

$$TFIDF = TF(w) \times IDF(w)$$

$$TF(w) = \frac{\text{Number of times word } w \text{ appears in a sentence}}{\text{Total number of words in the sentence}}$$

$$IDF(w) = \log\left(\frac{\text{Total number of sentences}}{\text{Number of sentences with word } w \text{ in it}}\right)$$

3. BERT Extractive Summarisation Library

   This is a Python library that uses word embedding to score the text[1]. Due to the nature of deep learning algorithm, the library requires a big amount of resources to run effectively.

As at least two out of three methods use word frequencies in the algorithm, the form of the words in the feedback texts affect the result. Hence, another experiment is to compare the input feedback texts in their original form and their lemmatized form, which is a method to get the root of a word.

The summarisation experiments use multiple datasets to cover several scenarios, i.e., by sentiments or by companies. By doing this, it is easier to gain insights, for example, to find the strengths and weaknesses of each company. Therefore, for each method, multiple datasets are used:

- Good feedback from whole dataset,
- Bad feedback from whole dataset,
- All feedback from each company,
- Good feedback from each company,
- Bad feedback from each company.

4.2 Sentiment Classification

The full script of sentiment classification can be seen in Appendix E.

4.2.1 Data Preprocessing

For the feedback summarisation, the cleaned and lemmatized texts are used. For the data preprocessing, the first step is to split the dataset into training and testing sets, which is very important to validate the trained model and make sure it is not overfitting. Overfitting happens when the trained model describes the training data too perfectly, resulting in poor performance for other kind of data.

```
x_train,x_test,y_train,y_test = train_test_split(
    df['lemmatized_feedback'], df['overall_rating'],
    test_size = 0.2, random_state = 1000)
```

[1]https://pypi.org/project/bert-extractive-summarizer/

This piece of code randomly takes 20 percents of the dataset into test set (3515 rows) and the remaining 80 percents into training set (14060 rows). The random state parameter is needed to ensure reproducability. In total, the training set has 14060 rows of data with 12615 rows of good feedback (overall rating = 1) and 1445 rows of bad feedback (overall rating = 0). As it can be seen that the dataset is imbalanced, resampling is applied to make it balanced, although the model will be trained using both imbalanced and balanced datasets for comparison and evaluation purposes. Oversampling is applied to the minority class 0 and undersampling is applied to the majority class 1.

```
ros = RandomOverSampler(sampling_strategy={0:2890},random_state=1000)
x_ros, y_ros = ros.fit_resample(x_train.values.reshape(-1,1), y_train)
```

The minority class is resampled to have 2890 number of data. Afterwards, the majority is downsampled to have the same number as the oversamples minority class. At the end, the balanced dataset has a total of 5780 number of data.

```
rus = RandomUnderSampler(random_state=1000)
x_train, y_train = rus.fit_resample(x_ros.reshape(-1, 1), y_ros)
```

The next step is to transform the texts into machine readable formats. The first two are bag-of-words and TF-DF:

- Bag of words representation

  The output for the bag-of-words method is a sparse matrix with all zeros except for some that denote the words.

  ```
  count_vectorizer = CountVectorizer(min_df=0, ngram_range=(1,1))
  cv_train = count_vectorizer.fit_transform(x_train)
  cv_test = count_vectorizer.transform(x_test)
  ```

  As unigram is used, the size of the vocabulary equals to the number of unique words, which in this case is 10148 words. Thus, the output for this transformation is a training set with shape (14060, 10148) and testing set with shape (3515, 10148). As an example, an input text 'friendly helpful professional quick easy sign friendly service knowledgeable' would be transformed into a matrix $[[0, 0, \cdots, 0, 0]]$ of length 10148 and each entry is zero except for entry index 3717 with value 2; and indexes 4197, 7072, 7265, 2849, 8213. 8095, 5120 with value 1. The entry index 3717 represents the word 'friendly' in the text and as there are two same words, the value becomes 2.

- TF-IDF representation

  The output is also a sparse matrix with all zeros except for some that denote the words. The difference from the bag-of-words method is that the values are weighted by the corpus frequency.

  ```
  tfidf_vectorizer = TfidfVectorizer(min_df=0, ngram_range=(1,1))
  tfidf_train = tfidf_vectorizer.fit_transform(x_train)
  tfidf_test = tfidf_vectorizer.transform(x_test)
  ```

The output also has the same dimension of (14060, 10148) adn (3515, 10148) for training and testing data, respectively. Using the same example text 'friendly helpful professional quick easy sign friendly service knowledgeable', the transformation yields a matrix $[[0, 0, \cdots, 0, 0]]$ of length 10148 and each entry is zero except for entry index 3717 with value 0.593, 4197 with value 0.213, 7072 with value 0.3339, and so on.

Another representation is the hardcoded indexed word embedding. It is similar to the bag-of-words representation, but it is applied to streamline the pipeline for the deep learning part. This method also takes the corpus vocabulary, assigns indexes for each word in it, and transform the text into a matrix form of index. The difference is that the output is not a sparse matrix, but instead a matrix with the shape of the text length,i.e., not the vocabulary length like the output from bag-of-words. It also has different indexing algorithm. Hence, an input text 'friendly helpful professional quick easy sign friendly service knowledgeable' is transformed into $[33, 4, 65, 44, 7, 68, 33, 5, 205]$. The code snippet for hardcoded indexed word embedding can be seen below:

```
tokenizer = Tokenizer(num_words=5000) # take most common n words
tokenizer.fit_on_texts(x_train)

indexed_word_embedding_train = tokenizer.texts_to_sequences(x_train)
indexed_word_embedding_test = tokenizer.texts_to_sequences(x_test)
```

As the output length of each sentence is different, it cannot be used in the training phase. Thus, a zero padding is applied at the end of each output to have the same length.

4.2.2 Baseline Models

There are two baseline models: logistic regression and simple neural network. For the logistic regression algorithm, it is trained using both bag-of-words and TF-IDF dataset.

```
# bag-of-words
log_reg_classifier = LogisticRegression()
log_reg_bow = log_reg_classifier.fit(cv_train, y_train)

# TD-IDF
log_reg_classifier = LogisticRegression()
log_reg_tfidf = log_reg_classifier.fit(tfidf_train, y_train)
```

The architecture for neural network is very simple with only 1 hidden layer and a classification layer using sigmoid function, shown in Figure 4.1. The model is trained using early stopping mechanism to avoid overfitting. With this mechanism, the training stops when the validation loss does not decrease for 3 consecutive epochs/iterations.

```
callback = keras.callbacks.EarlyStopping(monitor='val_loss',
                                         patience=3,
                                         restore_best_weights=True)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                101490
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 101,501
Trainable params: 101,501
Non-trainable params: 0
```

Figure 4.1: Baseline Model Neural Network Architecture

```
training_log = model.fit(tfidf_train, y_train,
                         epochs=15,
                         verbose=False,
                         validation_data=(tfidf_test, y_test),
                         batch_size=5,
                         callbacks=[callback]
                        )
```

With sigmoid classification function, the output prediction will be a real value between 0 and 1. Thus, the output is further processed with `prediction = [1 if x >= 0.5 else 0 for x in y_nn_predict]`, which makes the output to be either 0 or 1.

4.2.3 Word Embeddings

For the word embedding models, the training uses the hardcoded indexed word embedding representation. The architecture is similar to the neural network, but with an additional embedding layer in the beginning. The embedding layer takes input with vocabulary size dimension and gives output a matrix the size of input length and the intended embedding dimension. A flatten layer is added to reduce the dimension of the matrix into a single array to be processed by the dense layer, i.e., the hidden layer.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 100, 50)           508450
_________________________________________________________________
flatten_2 (Flatten)          (None, 5000)              0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                50010
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 558,471
Trainable params: 558,471
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.2: Word Embedding Architecture

The training is done similarly, but with the harcoded indexed word embedding dataset as the input.

```
callback = keras.callbacks.EarlyStopping(monitor='val_loss',
```

```
                                        patience=3,
                                        restore_best_weights=True)
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=15,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test,
                                          y_test),
                         batch_size=10,
                         callbacks=[callback]
                         )
```

As feedback text data are sequential, it is better to have a model that is able to learn the sequence information rather than to only learn each word positional information. To do this, a pooling layer is used, which will downsize the number of features. A global max pooling layer is added, which will take the highest valued feature across all features as can be seen in Figure 4.3. There's also another pooling layer, max pooling layer, which takes the highest valued feature from a set of features. However, in this case the performance between the two pooling layers is very similar.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           508450
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                510
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 508,971
Trainable params: 508,971
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.3: Word Embedding with Global Max Pooling Architecture

On all the architectures above, the embedding layer's weights are trained from the beginning. Another way to optimise the model is to use a pretrained word embedding weight. The benefit of using a pretrained word embedding is that it provides better initial weights that have been trained using a very large corpus and thus, can map word similarity. Refer to Section 2.3 for more details. In this project, the pretrained word embedding models used are GloVe and Word2Vec, both of which can be downloaded from their websites. For example, the GloVe model 'glove.6B.50d.txt' consists of 6 billion words of 50 dimension each. The next step is to map the weights from the pretrained model into the words in the vocabulary.

```
def create_embedding_matrix(filepath,word_index,embedding_dim):
    vocab_size = len(word_index) + 1  # plus 1 as index 0 is reserved
    embedding_matrix = np.zeros((vocab_size, embedding_dim))

    with open(filepath, encoding="utf8") as f:
        for line in f:
            word, *vector = line.split()
            if word in word_index:
                idx = word_index[word]
```

```
                embedding_matrix[idx] =
                    np.array(vector, dtype=np.float32)[:embedding_dim]

    return embedding_matrix
```

After the mapping, the GloVe model covers 85% - 90% of feedback dataset vocabulary, while the Word2Vec model covers 83% - 89%. Hence, there are dropouts vocabulary among the words. At this point, each word in the vocabulary has its own embedding weight and this is fed into the architecture at the embedding layers. The embedding layer also has an option on whether the dataset should be trained together with the pretrained embedding weights or the model should just use the pretrained embedding weights as they are. By experiments, having the dataset trained together yields better model.

```
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           weights=[embedding_matrix],
                           input_length=maxlen,
                           trainable=True))
```

The training is done similarly as before.

4.2.4 Convolution Neural Network

Experiments on CNN involve training on different kind of word embedding and to tune the hyperparameters at the end to get the best combination. In the network architecture, a convolutional layer is added after the embedding layer. It is called a 1 dimension convolutional layer as it convolves sequential data, such as text.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           508450
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 96, 128)           32128
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 541,879
Trainable params: 541,879
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.4: Convolutional Neural Network Architecture

Basic parameters on the convolutional layer are the number of output filters used in the convolution operation and the kernel size of the convolution window.

```
model.add(layers.Conv1D(128, # number of filters
                        5, # kernel size
                        activation='relu'))
```

Kernel size can also be seen as how many words the convolution window takes at one time to produce a feature vector the size of filters number. The last step to find the best model is to do the hyperparameters tuning. In this project, the tuned hyperparameters are the number of output filters (32, 64, 128) and kernel size (3, 5, 7). The process uses a randomised search, which randomly takes the combination of hyperparameters. Cross validation is also applied to make sure the generated models are general and cover many features.

```
RandomizedSearchCV(estimator=model,
                   param_distributions=param_grid,
                   cv=5, # cross validation
                   verbose=1,
                   n_iter=5, # parameter settings that are sampled
                   random_state=1
                  )
```

After finding the best combination of parameters, which yields the highest scored model, a final model is trained and saved to be used for prediction.

4.3 Feedback Summarisation

The full script of feedback summarisation can be seen in Appendix F and examples of summarisation results can be seen in Appendix B.

4.3.1 Data Preprocessing

For the feedback summarisation task, the only data preprocessing is to prepare few datasets for the experiment. First of all, prepare two datasets containing only the good (overall rating is 1) and bad (overall rating is 0) feedback:

```
df_good = df[df['overall_rating']==1]
df_bad = df[df['overall_rating']==0]
```

Next, prepare 17 datasets for each company in the itemReviewed_name column:

```
companies = list(set(df['itemReviewed_name']))
df_companies = {'df_' + str(company):df[df[
    'itemReviewed_name']==company] for company in companies}
```

The company dataset can be retrieved by, for example, df_companies['df_Avado'], which will give a feedback data for Avado. Lastly, the companies datasets needs to be split by the overall rating to yield 34 more datasets:

```
for company in companies:
    name = 'df_{0}'.format(company)
    df_companies['{0}_good'.format(name)] = df_companies[name][
        df_companies[name]['overall_rating']==1]
    df_companies['{0}_bad'.format(name)] = df_companies[name][
        df_companies[name]['overall_rating']==0]
```

These datasets can be accessed by, for example, df_companies['df_Avado_good'] and df_companies['df_Avado_bad'].

4.3.2 Weighted Word Occurrence Method

A dictionary is used to collect the word occurrence information; and iteratively read each word while increasing the count of the word as a value in the dictionary:

```
word_occurences = {}
for word in word_tokenize(combined_cleaned_feedback):
    if word not in word_occurences.keys():
        word_occurences[word] = 1
    else:
        word_occurences[word] += 1
```

To calculate the weighted word occurrence, the dictionary is updated by dividing each count value with the highest occurrence value:

```
highest_occurence = max(word_occurences.values())

for word in word_occurences.keys():
    word_occurences[word] = (word_occurences[word]/highest_occurence)
```

An example using the Avado dataset, the first three words in word occurrences dictionary before weighting are: (speaking, 11), (one, 148), (advisors, 18). With the highest occurrence value 1249, those three words value become (speaking, 0.0088), (one, 0.1184), (advisors, 0.0144).

Next, to score each sentence by utilising the weighted word occurrence dictionary to get each word value. There is an effort to filter out short sentences in assumption that short sentences do not have much value.

```
sentence_scores = {}
for sent in original_feedback_sentences:
    for word in word_tokenize(sent.lower()):
        if word in word_occurences.keys():
            if len(sent.split(' ')) < 30: # to filter out sentences
                if sent not in sentence_scores.keys():
                    sentence_scores[sent] = word_occurences[word]
                else:
                    sentence_scores[sent] += word_occurences[word]
```

Finally, after sorting the sentences by its score, the top n number of sentences are picked and combined to form a summary. In this method, common feedback words such as thank you, wonderful, etc. get more value as they appear more. However, those sentences do not give extra insights to the summary. Therefore, a sentence score threshold is applied.

```
summary = []
n = 0
for key, value in sorted_sentences.items():
    if n == n_sentence: break
    if value <= 5: # score threshold
        summary.append(key)
        n += 1
```

In fact, using the good feedback dataset, the top score sentences without looking at the threshold are: 'Thank you!', 'Thank you.', 'Excellent service.', 'So far so good.', 'Would recommend.', etc.; which are indeed not something insightful to be added in

the summary. Below is an output summary example for Avado using weighted word occurrence without lemmatization:

Thoroughly enjoyed doing my course via Avado The course material was excellent, the portal looked very professional and I had great tutor support throughout my course. honest and easy to understand process - would recommend Alison was really helpful with booking the course Great learning and experience and online set up! The course advisors at Avado have been extremely helpful, and all of the course information sent through so far has been very informative, clear and easy to understand. So easy to start a course, well explained, course is easy to work your way through Very helpful and informative. Excellent communication and efficient service Great customer service David was a great explaining the course and making the enrolment process really clear and easy. INCORRECT COURSE MATERIAL - The course would often provide guidelines that were needed to complete assignments for the course then assessors would fail the assignment for using their guidelines.5. Really easy process, with very helpful account managers I didn't start the course yet, so I cannot say anything about the course.

For the weighted word occurrence with lemmatization, besides the difference in the input text, a slight change is applied to the scoring script:

```
sentence_scores = {}
for sent in original_feedback_sentences:
    for word in word_tokenize(sent.lower()):
        if is_lemmatized:
            word = lemmatize_word(word)
        if word in word_occurences.keys():
            if len(sent.split(' ')) < 30: # to filter out sentences
                if sent not in sentence_scores.keys():
                    sentence_scores[sent] = word_occurences[word]
                else:
                    sentence_scores[sent] += word_occurences[word]
```

As the input text is also lemmatized, the word occurrence dictionary changes as well. The first three weighted word occurrences are: (speak, 0.0251), (one, 0.1156), (advisor, 0.0593). Then, the output summary for Avado using weighted word occurrence with lemmatization becomes:

honest and easy to understand process - would recommend Alison was really helpful with booking the course Great learning and experience and online set up! So easy to start a course, well explained, course is easy to work your way through Very helpful and informative. INCORRECT COURSE MATERIAL - The course would often provide guidelines that were needed to complete assignments for the course then assessors would fail the assignment for using their guidelines.5. Excellent communication and efficient service Great customer service David was a great explaining the course and making the enrolment process really clear and easy. The course advisors at Avado have been extremely helpful, and all of the course information sent through so far has been very informative, clear and easy to understand. Really easy process, with very helpful account managers I didn't start the course yet, so I cannot say anything about the course. This course was really simple to set up, great communication, the course looks great and the site looks easy to navigate round.

4.3.3 TF-IDF Method

First for the TF-IDF without lemmatization, to reduce computational time, calculate all words inverse document frequency (IDF):

```
word_idf = {}
for word in word_tokenize(combined_cleaned_feedback):
    if word in word_idf.keys():
        continue
    contained_sentence = 0
    for sent in original_feedback_sentences:
        if word in sent.lower():
            contained_sentence += 1
    if contained_sentence == 0: contained_sentence += 1 # smoothing
    word_idf[word] = np.log(len(original_feedback_sentences)
                            /contained_sentence)
```

An example using the Avado dataset, the first three words in word IDF dictionary are: (speaking, 6.0788), (one, 2.0294), and (advisors, 5.5864). See that the word 'one', which has more occurrences, has lower IDF value. Then, the term frequency is calculated alongside with the sentence score, which is simply a multiplication between the TF score and IDF score.

```
def calculate_sentence_score(sentence, word_idf):
    sentence_score = 0
    tokenized_sentence = word_tokenize(sentence)
    for word in tokenized_sentence:
        # calculate term frequency tf
        tf = sentence.count(word)/len(tokenized_sentence)

        # get the IDF score from dictionary
        if word in word_idf.keys():
            idf = word_idf[word]
        else:
            idf = 0

        sentence_score += tf*idf
    return sentence_score
```

The rest of the processes is similar to the weighted word occurrence method, which include sorting the sentences and choosing the highest score sentences below a certain threshold to form a summary. Below is an output summary example for Avado using TF-IDF method:

"If you're going to honour the twelve months free membership that you have promised, tell me how you plan to reimburse me for the four months from July to October 2019 that you offered but hasn't been included!Now lets move on to the HR Inform three month free subscription - they didn't bother to add the essential caveat that this is in fact not free at all - if you want the three free months, you have to pay for a 12 month subscription and you will then get 3 of those months for free.Now lets move on to the Avado learning platform. Please, please be very careful. There were spelling mistakes and other mistakes frequently. I turned up to another exam and the exam had not been booked with the exam center, luckily they had a space and let me sit my exam. I particularly had challenges with my local bank because of our local restrictions. You

sign in with your username and there is a tickbox option to remember your username. or it's just a hoax.

For TF-IDF with lemmatization, the calculation for both TF and IDF are modified into:

```
word_idf = {}
for word in word_tokenize(combined_cleaned_feedback):
    if word in word_idf.keys():
        continue
    contained_sentence = 0
    for sent in original_lemmatized_feedback_sentences:
        if word in sent.lower():
            contained_sentence += 1
    if contained_sentence == 0: contained_sentence += 1 # smoothing
    word_idf[word] = np.log(len(original_feedback_sentences)
                            /contained_sentence)
```

The difference is in the original feedback text, which has been lemmatized for this case.

```
def calculate_sentence_score(sentence, word_idf):
    sentence_score = 0
    tokenized_sentence = word_tokenize(sentence)
    for word in tokenized_sentence:
        # calculate term frequency tf
        tf = sentence.count(word)/len(tokenized_sentence)

        # get the IDF score from dictionary
        if is_lemmatized:
            word = lemmatize_word(word)
        if word in word_idf.keys():
            idf = word_idf[word]
        else:
            idf = 0

        sentence_score += tf*idf
    return sentence_score
```

Using lemmatization, the first three words in the IDF dictionaries are: (speak,4.9214), (one,2.1120), and (advisor,4.120). The decrease in value means that the word has more frequency due to the lemmatization. Then, the output summary for Avado using TF-IDF with lemmatization is:

"If you're going to honour the twelve months free membership that you have promised, tell me how you plan to reimburse me for the four months from July to October 2019 that you offered but hasn't been included!Now lets move on to the HR Inform three month free subscription - they didn't bother to add the essential caveat that this is in fact not free at all - if you want the three free months, you have to pay for a 12 month subscription and you will then get 3 of those months for free.Now lets move on to the Avado learning platform. Please, please be very careful. You sign in with your username and there is a tickbox option to remember your username. There were spelling mistakes and other mistakes frequently. I particularly had challenges with my local bank because of our local restrictions. I turned up to another exam and the exam had not been booked

with the exam center, luckily they had a space and let me sit my exam. or it's just a hoax.

4.3.4 BERT Extractive Summarisation Library

The implementation for BERT method is quite simple and straightforward because it is a Python library. The process starts with initiating the summarizer model and then feed the model the text and other parameters. However, the library by standard can only process text with a maximum number of characters of one million. Hence, for some datasets, the library simply gives an error.

```
def bert_summarisation(sentences):
    if len(sentences) > 1000000:
        return '''Feedback length exceeds 1000000 number of
        characters. It is going to cause memory allocation error.'''
    model = Summarizer()
    result = model(sentences, min_length=30, num_sentences=7)
    summary = ''.join(result)
    return summary
```

Below is an output summary example for Avado using BERT summarisation library:

When speaking to one of the advisors on the phone I felt that the answers that were given to my questions were being read off a screen. The site itself is very comprehensive and the induction section is very helpful to show you what to expect from your learning with AVADO.Would definitely recommend AVADO AvadoThe introduction to the training was excellent I had a great phone conversation where the outline of the course was discussed to see if it suited my requirements. My class will start on 7th October, will give more feedback regarding the session... Intranet portal is very easy to use, and probably the best staff I've seen in terms of politeness and helpfulness I found it a very easy process and all my questions were answered I really enjoyed my last AAT course with AVADO. the support team have a very poor response Found the enrolment process very easy, my course adviser gave me all the info needed and was very informative. First session with tutor was really good. Alice Beaven has been very helpful to me from day one of thinking about this course. I am very happy that I have chosen Avado for my course. Andrew brine of home learning college was proberly the most helpful person I've ever spoke to on the phone.. Enrolment process was very quick and easy, just a quick phone call and any questions were answered and full information given so you knew straight away if the course was right for you and if it was affordable Recently joined the Home Learning College and I have found their services to be brilliant and their aftercare is very helpful.

Chapter 5

Evaluation and Deployment

This chapter evaluates the results from the implementations and to analyse the pros and cons of the results. Recommendations are also given for both sentiment classification and feedback summarisation. The deployment section shows the prototype of Tableau dashboard to be proposed for the business.

5.1 Evaluation

5.1.1 Sentiment Classification

For each model built, a classification report is produced. It has score metrics for each class label and the total score as well. In a report, the information provided are:

- Precision: how many are correctly classified among that class $(\frac{TP}{(TP+FP)})$
- Recall: the ability to find all positive samples $(\frac{TP}{(TP+FN)})$
- F1 Score: mean between precision & recall
- Support: the number of occurrence of the given class in dataset, in this case it is the size of the validation set

The validation set consists of 3515 rows of feedback data with 3182 rows good feedback (class label = 1) and 333 rows bad feedback (class label = 0). As it is an imbalanced dataset, accuracy score is not a good metric. It is better to focus on the precision and recall scores for the minority class (class 0). Even with this, the scores might seems to be misleading and therefore, they are sometimes drilled down to the number of true positives, false positives, and false negatives for the minority class. For each evaluation, the scores are compared between models trained using imbalanced and balanced datasets.

The scores for each model can be seen from Table 5.1 to Table 5.11. The scores for class 1 prediction are constantly high; and this applies to all models, regardless of the technique involved. For the case of imbalanced dataset, it makes sense as the model is biased towards predicting class 1. However, for the balanced dataset, high

precision and recall for class 1 might suggest that the features within the training data with class 1 are representative enough.

It is more interesting to see the scores for class 0. At one glance, it seems that the models trained using imbalanced dataset perform better than the models trained using the balanced dataset. As a biased model, it makes sense if the model has a tendency to classify the feedback as the majority class (class 1) and hence, the lower recall for class 0. However, with unbiased model, it's intriguing for the model to have a very low precision on class 0, albeit the high recall.

To understand the reason behind such scores, inspection on the prediction results is conducted. Upon closer inspection, it turns out that the model trained using the balanced dataset is able to capture more feedback with negative words in it, which have high ratings (class 1). Please note that the whole feedback might not have a negative sentiment, but may just contain negative words. This explains the high number of false positives on class 0 as most of these false positives contain negative words/phrases, and thus, low precision score. In other words, the model is more sensitive towards class 0 and indicates overpowering features of class 0. In contrast, the biased model has lower false positives, which explains the high precision. Moreover, looking at the average prediction scores, the false positives from unbiased model has lower average of 0.19, compared to the biased model of 0.3. This may also suggest that the biased model gives more weight on features of class 1.

Examples of false positives feedback, which are captured by the unbiased model, but are not captured by the biased model include:

- (Rating 5) *The network be horrible so it be hard for me to fully understand or hear but other than that, I be able to get clear instructions.*
- (Rating 5) *Everything be amaze except the hour of their call and the reschedule should be do online and not email or call back and forth multiple time a day. People in America don't have that kind of time. That's why I stopped. Other than that, it be very organized.*
- (Rating 5) *I've be use this platform for many time. Materials be so useful, I don't understand why there be so many negative reviews. The website be well know for many people and it generate innumerable benefits.*

Another aspect, which makes the model trained using balanced dataset is better, is the lower number of false negatives for class 0. The first impact of this is that the unbiased model has a higher number of true positives compared to the biased model; hence the high recall. The problem with the biased model is that there are more false negatives feedback with negative words, marked by the lower average prediction scores. In other words, the model simply misclassifies the feedback. Examples of such feedback are:

- (Rating 1) *Initially the service be great to get me onboard, however I decide to cancel my enrollment before the 14 day period and have not yet be successful.*

- (Rating 2) *Overall the experience with ICS Learn be OK. The student service team be always helpful and efficient with responses. Unfortunately the level of support from the Tutor, Keith Watson wasn't great. Feedback be always provide with minimal information and somewhat unhelpful when try to seek guidance well far assignments. The notification of my pas be not sent to student service therefore meaning my confirmation of completion have be delayed. I would probably look to use other learn provider in future.*
- (Rating 3) *Navigating the website/learning platform could be better.*

There may be many factors that contribute on such behaviour of the models. First, improving the training example selection might improve the performance. More importantly, there are some limitations in feature extraction for this project. The input feedback text for the model are lemmatized and although by theory this step does not affect the overall meaning of the feedback, but in some cases the lemmatizer outputs wrong root words. The other thing is the removal of stopwords. Stopwords by nature should not have any weights on the sentences. However, some stopwords indicate the sentiment of the sentences and removing these words completely change the sentiment of the overall feedback. Next is the discussion of each model performance.

Baseline Models

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.85	0.74	0.79
Imbalanced Dataset - Class 1	0.97	0.99	0.98
Balanced Dataset - Class 0	0.61	0.86	0.71
Balanced Dataset - Class 1	0.98	0.94	0.96

Table 5.1: Logistic Regression (Bag-of-words) Scores

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.91	0.62	0.73
Imbalanced Dataset - Class 1	0.96	0.99	0.98
Balanced Dataset - Class 0	0.59	0.90	0.71
Balanced Dataset - Class 1	0.99	0.93	0.96

Table 5.2: Logistic Regression (TF-IDF) Scores

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.86	0.74	0.79

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 1	0.97	0.99	0.98
Balanced Dataset - Class 0	0.67	0.91	0.77
Balanced Dataset - Class 1	0.99	0.95	0.97

Table 5.3: Simple Neural Network Scores

Table 5.1 and Table 5.2 show the models performance using logistic regression with two different data representations, while Table 5.3 shows the score for neural network model. For the imbalanced dataset, the baseline model using neural network is just slightly better than the logistic regression as it has a slightly higher precision, while the logistic regression model using TF-IDF dataset is arguably worse than the others because as though the precision is the highest, the recall score is very low. This suggests that there are many false negatives and less true positives for class 0.

For the balanced dataset, the model with neural network is clearly better than the logistic regression models as the precision and recall scores for class 0 are both higher. This might indicate that information/features within feedback data can be discovered more using neural network, which makes sense as more information, such as sequential information from the text can be learned as well.

Word Embeddings

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.84	0.75	0.79
Imbalanced Dataset - Class 1	0.97	0.99	0.98
Balanced Dataset - Class 0	0.71	0.81	0.76
Balanced Dataset - Class 1	0.98	0.97	0.97

Table 5.4: Hardcoded Indexed Word Embedding Without Pooling Scores

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.86	0.79	0.83
Imbalanced Dataset - Class 1	0.98	0.99	0.98
Balanced Dataset - Class 0	0.67	0.90	0.77
Balanced Dataset - Class 1	0.99	0.95	0.97

Table 5.5: Hardcoded Indexed Word Embedding With Pooling Scores

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.88	0.74	0.80
Imbalanced Dataset - Class 1	0.97	0.99	0.98
Balanced Dataset - Class 0	0.68	0.81	0.74
Balanced Dataset - Class 1	0.98	0.96	0.97

Table 5.6: GloVe Word Embedding Scores

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.88	0.79	0.83
Imbalanced Dataset - Class 1	0.98	0.99	0.98
Balanced Dataset - Class 0	0.63	0.90	0.74
Balanced Dataset - Class 1	0.99	0.95	0.97

Table 5.7: Word2Vec Word Embedding Scores

For the word embeddings, using pooling layer obviously yields a better model because the features are more generalised and indeed, the model has similar, if not better, performance to the baseline model. When using pretrained word embedding models, such as GloVe and Word2Vec, it is necessary to have the pretrained model to be trained together with the input data vocabulary. Without it, the performance is very poor as the weights of each word embedding are not updated based on the dataset. From the result, it seems that using pretrained word embedding does not improve the performance by much.

Convolution Neural Network

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.83	0.78	0.80
Imbalanced Dataset - Class 1	0.98	0.98	0.98
Balanced Dataset - Class 0	0.62	0.88	0.72
Balanced Dataset - Class 1	0.99	0.94	0.96

Table 5.8: CNN Without Pretrained Word Embedding Scores

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.89	0.74	0.80
Imbalanced Dataset - Class 1	0.97	0.99	0.98
Balanced Dataset - Class 0	0.65	0.85	0.74

Input Training Dataset	Precision	Recall	F1 Score
Balanced Dataset - Class 1	0.98	0.95	0.97

Table 5.9: CNN With GloVe Word Embedding Scores

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.87	0.78	0.83
Imbalanced Dataset - Class 1	0.98	0.99	0.98
Balanced Dataset - Class 0	0.62	0.89	0.73
Balanced Dataset - Class 1	0.99	0.94	0.96

Table 5.10: CNN With Word2Vec Word Embedding Scores

Input Training Dataset	Precision	Recall	F1 Score
Imbalanced Dataset - Class 0	0.82	0.79	0.80
Imbalanced Dataset - Class 1	0.98	0.98	0.98
Balanced Dataset - Class 0	0.77	0.82	0.79
Balanced Dataset - Class 1	0.98	0.97	0.98

Table 5.11: CNN With Best Parameters Word Embedding Scores

For the models trained using CNN architecture, using imbalanced dataset yields model with similar performance with other models mentioned previously. However, with the exception of the model trained using best parameters, the unbiased models trained using balanced dataset do not have better performance than the baseline model. One thing that may affect this is the number of training data (data selection). Deep learning, especially CNN, is known to require a large amount of data to be able to perform well. In this case, the models trained using balanced dataset only has 5780 rows of input data, compared to the initial dataset of 14060 rows. Also, in this project the model learns the features for each word (unigram), not combinations of words (bigram, trigram, etc). As deep learning architecture should be able to capture more information, these kind of features may improve the performance. However, after hyperparameter tuning the models have more balanced scores, indicating further tuning might still improve the scores.

For each model trained, the training history is plotted into graphs of training and validation set accuracy/loss. Figure 5.1 shows the training history for CNN using Word2Vec word embedding and best parameters combination trained using imbalanced dataset, while Figure 5.2 shows the training history for CNN using GloVe word embedding and best parameters combination trained using balanced dataset. For other models' training history, refer to the sentiment classification script in Appendix E. The training plots stop before the model becomes too overfitting by using early stopping mechanism; and this applies to all models.

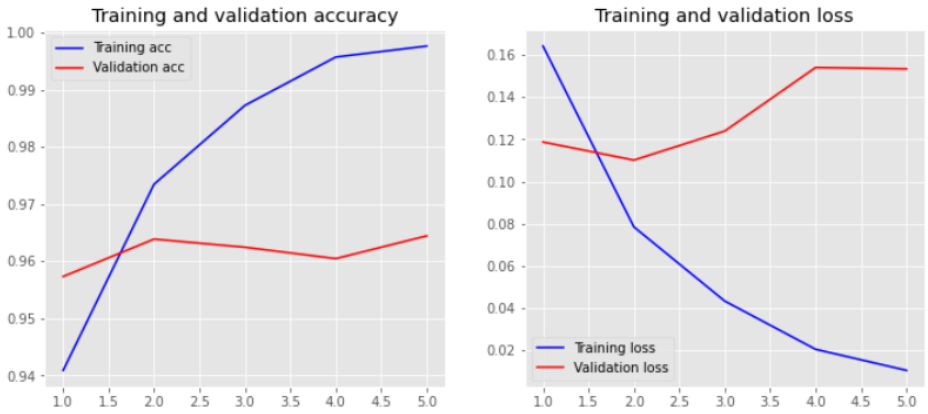


Figure 5.1: Training History Plot: CNN With Word2Vec Embedding, Best Parameters & Imbalanced Dataset

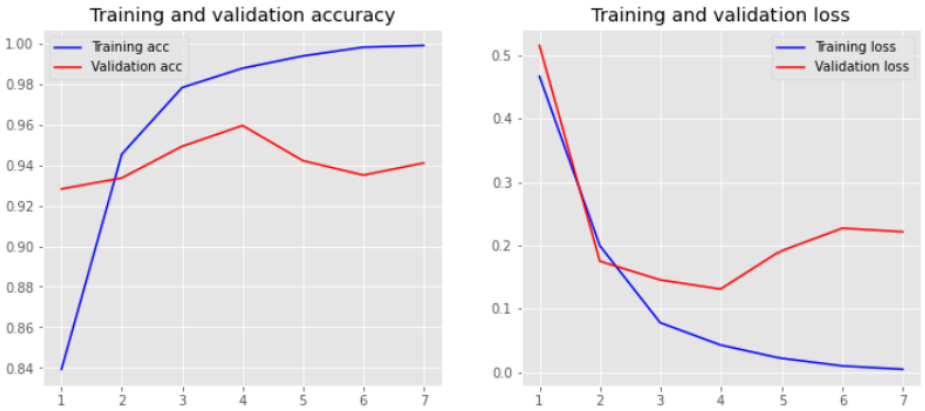


Figure 5.2: Training History Plot: CNN With GloVe Embedding, Best Parameters & Balanced Dataset

To sum up, for the imbalanced dataset, almost all produced models have similar performance. With the exception of base model using TF-IDF representation of having low recall, other models have comparable precision and recall scores. However, the CNN model with Word2Vec embedding is arguably slightly better than the rest with a more balanced precision and recall for class 0. After tuning the hyperparameters, the model has lower precision and a slightly higher recall than the original model with Word2Vec. This might be caused by the sampling of hyperparameters when tuning them. With the best combination of parameters, the CNN model with Word2Vec word embedding gives 262 true positives, 56 false positives, and 71 false negatives for class 0.

And for the balanced dataset, while it looks like that the models are not performing better due to the lower score, they are actually able to capture more distinctive features. This results in the models ability to predict the sentiment of more feedback correctly. With the best combination of parameters, the CNN model with GloVe word embedding gives 272 true positives, 81 false positives, and 61 false negatives for class 0. They are little differences between the two models with best parameters.

This may suggest that the features between good and bad feedback are distinctive enough that the amount of training data, whether they are resampled or not, are representative.

5.1.2 Feedback Summarisation

The evaluation for feedback summarisation uses three different methods: running time, scoring, and domain experts evaluation. Running time is an important aspect as server resources are limited within the company and there are many processes queuing throughout the day. For this reason, the BERT method is not suitable as it takes long time to produce a summary. This might be solved if there is GPU access in the server later in the future, however, in the current situation, this method is out of consideration. Thus, for the other evaluation methods, only the weighted word occurrence and TF-IDF methods are compared.

For the scoring method, a tool called ROUGE, Recall-Oriented Understudy for Gisting Evaluation, is used. It is simply a metric to compute the quality of the produced summary against the reference. There several types of ROUGE and for this evaluation, ROUGE-1 is chosen, which calculates the quality based on the unigram. The result is given as precision, recall, and F1 scores. The precision can be defined as how many the words in the summary appeared in the reference, and on the contrary, the recall is defined as how many the words in the reference appeared in the summaries. The F1 score is calculated as $2 \times (Precision \times Recall)/(Precision + Recall)$. Table 5.12 and Table 5.13 show examples of scores comparison between the weighted word occurrence and TF-IDF methods using the good feedback and Avado dataset.

Method	Precision	Recall	F1 Score
Weighted Word Occurrence (no lemmatization)	1	0.000240	0.000480
Weighted Word Occurrence (with lemmatization)	1	0.000221	0.000442
TF-IDF (no lemmatization)	0.997101	0.000599	0.001198
TF-IDF (with lemmatization)	0.995726	0.000406	0.000811

Table 5.12: Summarisation Scores for Good Feedback Dataset

Method	Precision	Recall	F1 Score
Weighted Word Occurrence (no lemmatization)	1	0.002002	0.003996
Weighted Word Occurrence (with lemmatization)	1	0.001956	0.003905
TF-IDF (no lemmatization)	1	0.002048	0.004087
TF-IDF (with lemmatization)	1	0.002048	0.004087

Table 5.13: Summarisation Scores for Avado Dataset

Observing the scores, the precision scores are closed or equal to 1. This is because the nature of extractive summarisation technique, which forms the summary using the sentences extracted from the reference without changing the words. Thus, the precision score can be ignored and it is more important to evaluate the recall and f1 scores. The scores are generally higher for the TF-IDF methods and it is to be expected as in TF-IDF method, more unique words in the sentence get higher score and hence, the summaries from TF-IDF method have more unique words, which contribute to higher recall and F1 scores. Similarly, the reason that non-lemmatization method scores are generally higher than the lemmatization method is because of the number of unique words in the summary. Using lemmatization technique, all words are computed using their root form, so different word forms are deemed as one word, which reflect on the produced summaries.

Looking at the scores, there's only slight difference between method with and without lemmatization, but as the TF-IDF scores are higher, it might produce better quality summaries. However, asking the domain experts for evaluating the summaries show different opinion. After showing them the produced summaries and the high-level explanation of the techniques used, they argued that the weighted word occurrence method is more beneficial/important for the business. The reason for this is that for company portfolio/marketing purpose, it is better to have feedback that represent more the general sentiment of the customers. Moreover, if more customers give similar bad feedback, then it is more logical to address the problems as high priority. Also, they subjectively felt that the weighted word occurrence without lemmatization produces a better flowed summaries. However, the TF-IDF method might also be good to find customers with unique experience, but in general, this is not something the business would like to see often. As for the drawback, they deemed the summaries to be quite difficult to understand due to the grammatical and semantic problems. This is to be expected from extractive summarisation of short unrelated feedback data. A further exploration to improve the summaries is a possible direction for future works.

To sum up, the extractive summarisation method has its own pros and cons. For the pros, it is easier to implement than the abstractive summarisation, which results in a lightweight model and faster processing time. In contrast, the cons for this method is that the readability of the summaries is an obvious problem. Moreover, it is quite difficult to evaluate the quality using scoring metric as the summaries contain the exact same match as sentences from the reference.

5.2 Deployment

For the deployment for business use, a Tableau dashboard is created. Tableau is chosen because it is the standard reporting tool in the company. Ideally, the process will use a task scheduler to run the Python scripts, which process the data, everyday at a certain time, the data are stored in a database, and Tableau will connect to the database and refresh the data everyday. For this prototype dashboard in this project, the process will be just running the scripts and output excel/csv files to

be fed on the dashboard. While Tableau itself provides a way to connect to REST APIs, it is not ideal for business use as it will take longer time to load the data in the dashboard. The dashboard contains information on the feedback dataset, along with the additional sentiment and feedback summary data. The dashboard is split into three tabs: overview (Figure 5.3), feedback (Figure 5.4), and summary (Figure 5.5).

The overview dashboard is meant to give the statistics on the feedback dataset. It first shows the number of feedback by each company, followed by the breakdown by the ratings. As a business, it is also important to compare the percentage of those ratings to have a more insightful information on the company's performance. Finally, there is the percentage of feedback, broken down by the sentiment. This will quickly show the comparison between each company easily.

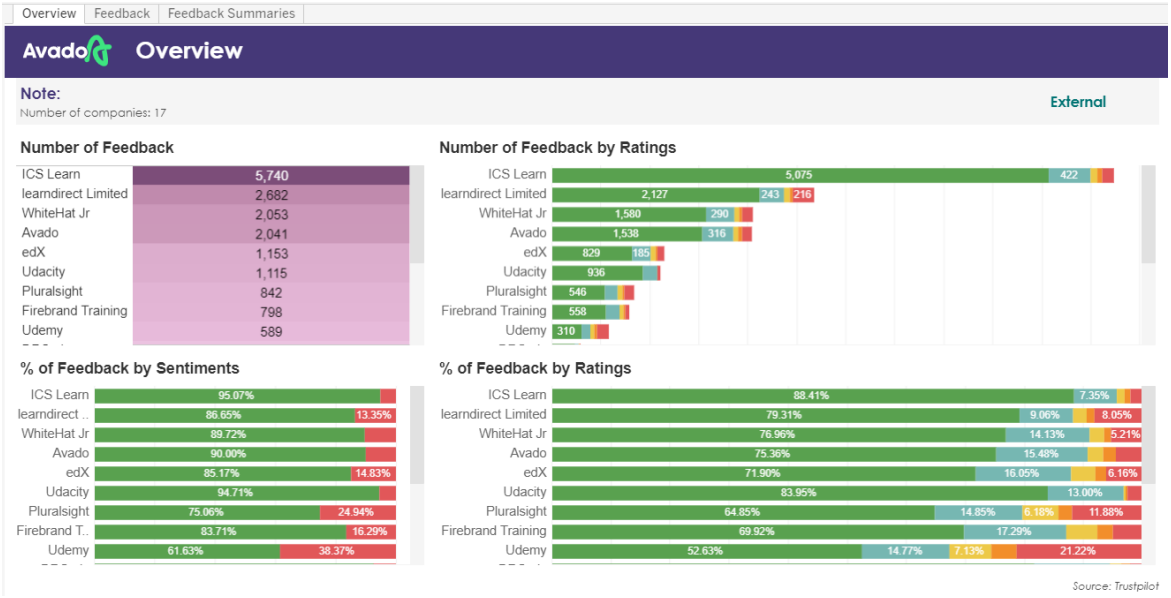


Figure 5.3: Tableau Dashboard: Overview

The feedback dashboard provides access to the feedback data, sorted by the date published. The feedback body text, the company, and the ratings are added as well. The colours on the ratings depict the sentiment classified for the respective feedback. In other words, a rating 5 feedback with red colour means that it is classified as negative/bad feedback. Four filters are provided as well. User can manipulate the data shown on the dashboard by changing the range of date published, the list of companies, the number of ratings, and the predicted sentiment. This is particularly useful to quickly get the list of feedback for use cases.

The summary dashboard is pretty straightforward. It shows the three different summaries of a company: the overall, good, and bad. The set of summaries used is the weighted word occurrence method result based on the evaluation of the previous section. For a full list of the summaries, refer to Appendix B.

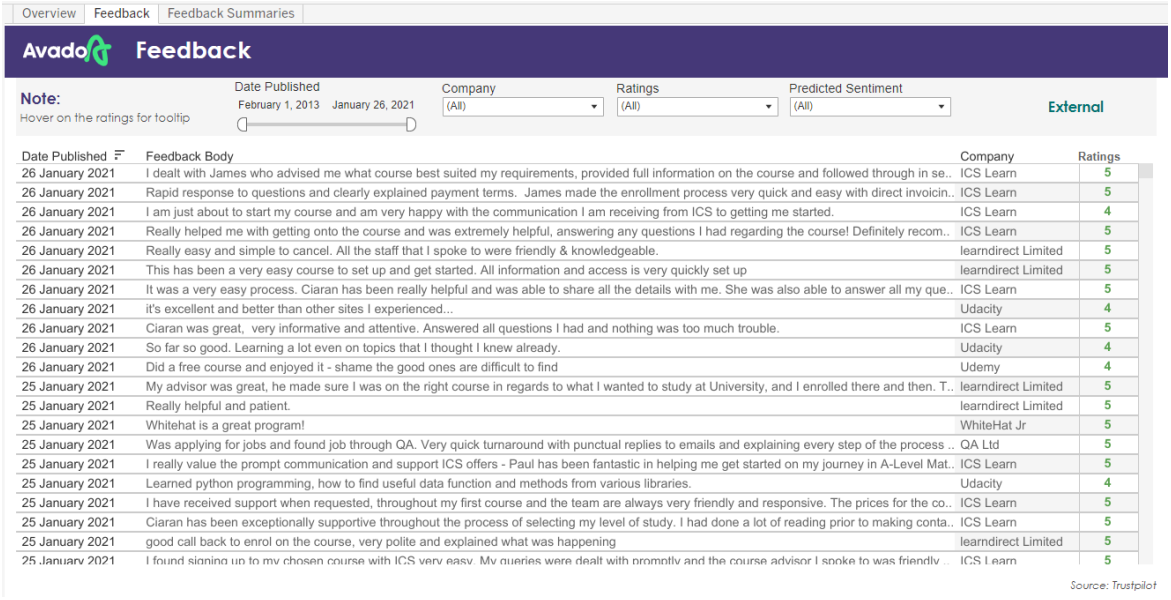


Figure 5.4: Tableau Dashboard: Feedback

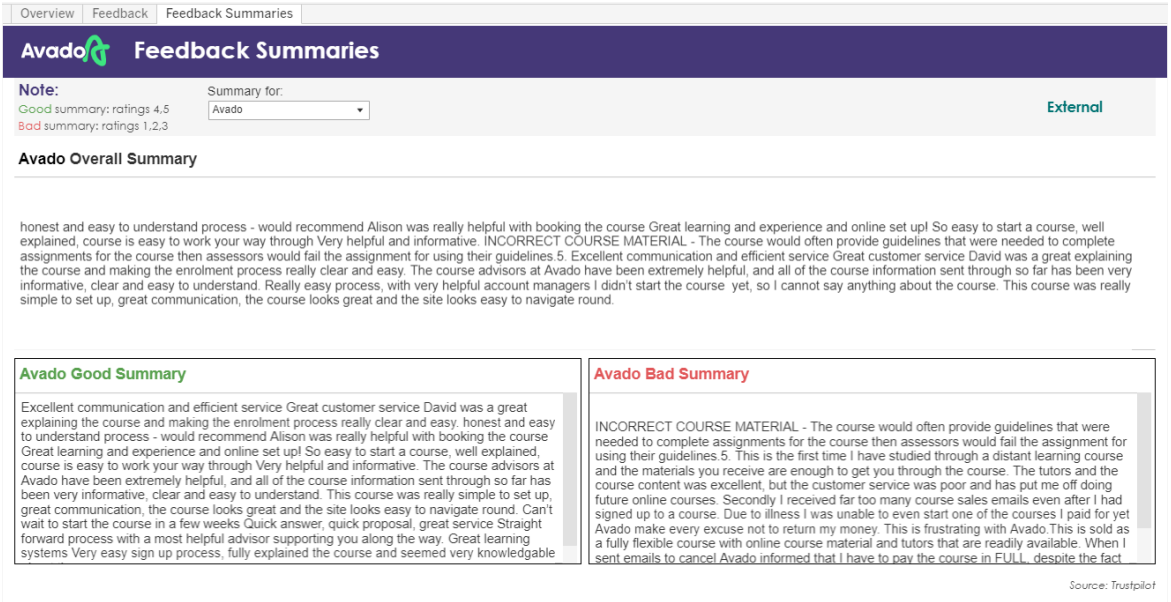


Figure 5.5: Tableau Dashboard: summary

Chapter 6

Conclusion and Future Works

6.1 Conclusion

Analysing feedback data has become a crucial part within a business to understand their customers and also make improvements. There are many information that can be extracted from the data, however this is an arduous task if done manually. In the first place, reading through the feedback one by one takes a very long time; this also happens because the number of feedback will keep increasing. Moreover, it is difficult to read them from the data source website only; it lacks advance filtering and other information that may help quicken the process. Thus, it is better to have the data scraped and processed automatically. Two of many information that can be extracted are the sentiment/polarity and the summary.

With the sentiment information, it is easy to find out customers' satisfaction towards the products/services. Sentiment in feedback can be classified as positive and negative. Positive feedback can also be used as marketing campaign material, appreciation, and ambassador offering. On the other hand, negative feedback can be used to improve/fix the problems mentioned and add more features on the products/services. Sentiment classification is a popular NLP task with huge development and research in the area. To build a classifier, input data and the labels are needed with the input feedback text data have to be transformed into machine readable format. Few methods that can be used include the bag of words and TF-IDF methods, which transform the sentences into sparse matrices of word positions based on the vocabulary corpus; and also the word embedding, which is able to give contextual/word similarity information. The labels in this case are either 1 (positive) or 0 (negative); this can be easily achieved by transforming the feedback ratings.

The training phase for sentiment classification involves building several model with different combination of data representation, machine learning algorithm, deep learning architecture, and the pretrained word embeddings. The nature of feedback data is that it is imbalanced with ratio of positive and negative feedback is approximately 10:1. By theory, training a model with imbalanced dataset yields a biased model towards the majority. To deal with this problem, resampling (upsampling & downsampling) technique is applied to the dataset. Nevertheless, models trained using both balanced and imbalanced dataset are compared and evaluated together. Indeed,

the unbiased model is able to predict more feedback sentiment correctly. Machine learning and deep learning algorithms used in this project are logistic regression, simple neural network, and convolutional neural network. In terms of performance scores, all algorithms show promising result with only slight differences between them. More hyperparameters tuning may show more contrast in the models performance.

Feedback summaries are also important information to get a quick overview/trend within the data. In this project, extractive summarisation technique is chosen, which creates a summary by taking the most representative sentences from the data without changing the context. The scoring method uses weighted word occurrence and TF-IDF, which takes word frequencies as the base for calculation; and also word embedding method with word similarity score added on top. Moreover, the scoring is applied to the sentences with only basic cleaning and with added lemmatization process. The evaluation is done both objectively and subjectively. Objective evaluation calculates the precision, recall, and f1 scores of the summary. However, it does not really help understand the quality as with the extractive method, the precision will always be 1. Subjective evaluation is done by asking domain experts for their inputs. The domain experts here range from technical to non-technical people, such as marketing/business people. They argue that the summaries with weighted word occurrences are better as they are more representatives.

To conclude, both sentiment classification and summaries can be both automated to give faster information than by manual reading. There are many ways to improve the results although the current results are also acceptable for initial business use. As for deployment, a Tableau dashboard is created to give more advance filtering functions, more centralised data, and easier access. It also contains many other information that can be found naturally within the dataset.

6.2 Future Works

There are several direction for future works in both sentiment classification and text summarisation for feedback data. These directions include ways to improve current results and also, new methods/techniques. For the sentiment classification, firstly, tuning more hyperparameters might give better scores. A second promising direction is to tune the feature extraction, such as selective lemmatization, specific stopwords filtering, and training examples/features selection. As indicated from the results, the features from good and bad feedback are distinctive and so, improving them will give better results. Next, in this project the words are processed one by one (unigram), however, processing multiple words at once, such as bigram, trigram, etc., may give the model the ability to capture more sequential information between words such that it is context aware. There are also other pretrained word embedding models, such as ELMo and BERT, which are trained using even bigger and richer corpus. While in this project, the focus of deep learning is CNN, in the NLP domain there are also other emerging architectures, such as RNN. There's also possibilities of creating new architecture by combining existing ones. In any of deep learning

applications, with more data the model usually performs better. Hence, to get more feedback data from other sources may also improve the result. Other improvement that can be made is expanding the prediction class with an 'unsure'/'neutral' label, which is for the feedback with classification score near the borderline. A multi-class classification is also interesting and it can be done by changing the class label into the number of ratings.

For the text summarisation, with the extractive method, it is likely that the summaries are difficult to understand due to grammatical and semantical errors. An improvement for these summaries would be a really nice addition for the user; it can be by either grouping the sentences by sentiment polarity, using connector words to give more natural context, etc. As for any extractive method, a direction would be to use more complex formalisation of the scoring, sorting, and selecting methods. Besides the extractive summarisation technique, there is also the abstractive summarisation technique, which generates more human-like summaries. Another direction for future work is to have more complex evaluation for the summaries quality, for example by comparing generated summaries with human written summaries.

Bibliography

[1] M. Allahyari, S. A. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. Text summarization techniques: A brief survey. *CoRR*, abs/1707.02268, 2017. pages 14

[2] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018. pages 13

[3] A. Brazinskas, M. Lapata, and I. Titov. Unsupervised opinion summarization as copycat-review generation. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5151–5169. Association for Computational Linguistics, 2020. pages 14

[4] K. R. Chowdhary. *Natural Language Processing*. Springer, 2020. pages 7

[5] G. G. Chowdhury. Natural language processing. *Annu. Rev. Inf. Sci. Technol.*, 37(1):51–89, 2003. pages 7

[6] C. N. dos Santos and M. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In J. Hajic and J. Tsujii, editors, *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 69–78. ACL, 2014. pages 6, 13, 16

[7] Y. Goldberg. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.*, 57:345–420, 2016. pages 7

[8] Y. Goldberg. *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2017. pages 7

[9] I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. pages 7

[10] D. Gupta, P. Lenka, H. Bedi, A. Ekbal, and P. Bhattacharyya. IITP at IJCNLP-2017 task 4: Auto analysis of customer feedback using CNN and GRU network. In C. Liu, P. Nakov, and N. Xue, editors, *Proceedings of the IJCNLP 2017, Shared Tasks, Taipei, Taiwan, November 27 - December 1, 2017, Shared Tasks*, pages 184–193. Asian Federation of Natural Language Processing, 2017. pages 6, 15

[11] A. Jacovi, O. S. Shalom, and Y. Goldberg. Understanding convolutional neural networks for text classification. In T. Linzen, G. Chrupala, and A. Alishahi, editors, *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 56–65. Association for Computational Linguistics, 2018. pages 13

[12] P. K. Jain and R. Pamula. A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews. *CoRR*, abs/2008.10282, 2020. pages 14

[13] R. Johnson and T. Zhang. Effective use of word order for text categorization with convolutional neural networks. In R. Mihalcea, J. Y. Chai, and A. Sarkar, editors, *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 103–112. The Association for Computational Linguistics, 2015. pages 13, 16

[14] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665. The Association for Computer Linguistics, 2014. pages 13

[15] A. Khan, M. A. Gul, M. Zareei, R. B. Rajesh, A. Zeb, M. Naeem, Y. Saeed, and N. Salim. Movie review summarization using supervised learning and graph-based ranking algorithm. *Comput. Intell. Neurosci.*, 2020:7526580:1–7526580:14, 2020. pages 14

[16] S. Kim and E. H. Hovy. Determining the sentiment of opinions. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*, 2004. pages 2, 6, 16

[17] Y. Kim. Convolutional neural networks for sentence classification. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL, 2014. pages 13

[18] O. Klymenko, D. Braun, and F. Matthes. Automatic text summarization: A state-of-the-art review. In J. Filipe, M. Smialek, A. Brodsky, and S. Hammoudi, editors, *Proceedings of the 22nd International Conference on Enterprise Information Systems, ICEIS 2020, Prague, Czech Republic, May 5-7, 2020, Volume 1*, pages 648–655. SCITEPRESS, 2020. pages 2, 14

[19] Y. Lecun and Y. Bengio. *Convolutional networks for images, speech, and time-series*. MIT Press, 1995. pages 12, 13

[20] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 396–404. Morgan Kaufmann, 1989. pages vii, 12

[21] E. D. Liddy. *Natural Language Processing*. Marcel Decker, Inc., 2001. pages 7

[22] B. Liu. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012. pages 2, 6

[23] C. McCormick. Word2vec tutorial - the skip-gram model. `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/`. Accessed: 2020-12-29. pages 10

[24] C. McCormick. Word2vec tutorial part 2 - negative sampling. `http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/`. Accessed: 2020-12-30. pages 10

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In Y. Bengio and Y. LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. pages vii, 10, 11

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013. pages 10

[27] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In K. Knight, H. T. Ng, and K. Oflazer, editors, *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 115–124. The Association for Computer Linguistics, 2005. pages 6

[28] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Philadelphia, PA, USA, July 6-7, 2002*, pages 79–86, 2002. pages 2, 6, 16

[29] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. pages 10

[30] A. Rajaraman and J. D. Ullman. *Data Mining*, page 1–17. Cambridge University Press, 2011. pages 9

[31] N. Shrestha and F. Nasoz. Deep learning sentiment analysis of amazon.com reviews and ratings. *CoRR*, abs/1904.04096, 2019. pages 6, 15

[32] P. D. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 417–424. ACL, 2002. pages 6

[33] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 352–357. The Association for Computer Linguistics, 2015. pages 13, 15

[34] M. Yang, Q. Qu, Y. Shen, Q. Liu, W. Zhao, and J. Zhu. Aspect and sentiment aware abstractive review summarization. In E. M. Bender, L. Derczynski, and P. Isabelle, editors, *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1110–1120. Association for Computational Linguistics, 2018. pages 14

[35] S. Yang, L. Cui, J. Xie, and Y. Zhang. Making the best use of review summary for sentiment analysis. In D. Scott, N. Bel, and C. Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 173–184. International Committee on Computational Linguistics, 2020. pages 14

[36] L. Zhang, S. Wang, and B. Liu. Deep learning for sentiment analysis : A survey. *CoRR*, abs/1801.07883, 2018. pages 6, 15

[37] X. Zhang and Y. LeCun. Text understanding from scratch. *CoRR*, abs/1502.01710, 2015. pages 15

[38] X. Zhang, J. J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657, 2015. pages 13, 15

[39] Y. Zhang, R. Jin, and Z. Zhou. Understanding bag-of-words model: a statistical framework. *Int. J. Mach. Learn. Cybern.*, 1(1-4):43–52, 2010. pages 8

Appendix A

Competitors List

Company	Trustpilot URL
Avado	https://uk.trustpilot.com/review/avadolearning.com
BPP	https://uk.trustpilot.com/review/www.bpp.com
Coursera	https://uk.trustpilot.com/review/coursera.org
Deloitte	https://uk.trustpilot.com/review/deloitte.com
DPG plc	https://uk.trustpilot.com/review/www.dpgplc.co.uk
edx	https://uk.trustpilot.com/review/www.edx.org
Firebrand Training	https://uk.trustpilot.com/review/firebrand.training
FutureLearn	https://uk.trustpilot.com/review/www.futurelearn.com
General Assembly	https://uk.trustpilot.com/review/generalassemb.ly
ICS Learn	https://uk.trustpilot.com/review/icslearn.co.uk
LearnDirect	https://uk.trustpilot.com/review/www.learndirect.com
Linkedin Learning	https://uk.trustpilot.com/review/uk.linkedin.com
PluralSight	https://uk.trustpilot.com/review/pluralsight.com
QA ltd	https://uk.trustpilot.com/review/qa.com
Udacity	https://uk.trustpilot.com/review/udacity.com
Udemy	https://uk.trustpilot.com/review/udemy.com
Whitehat Jr	https://uk.trustpilot.com/review/whitehatjr.com

Table A.1: Avado Competitors List

Appendix B

Feedback Summarisation Results

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
Good feedback	Really excited to work on with this course Good Course & Helpful staff.Very easy and stress-free sign up, all information provided to make easy to set up and login. Really helpful, easy to enrol, changed my course before I started, this was no hassle at all... Great course, really informative, loads of help. Keep up the good work! so far so good everything about the learn direct is perfect Great experience,great start up and materials.really happy Lovely course and very helpful customer service. Highly recommend. honest and easy to understand process - would recommend Alison was really helpful with booking the course Great learning and experience and online set up! Thoroughly enjoyed doing my course via Avado The course material was excellent, the portal looked very professional and I had great tutor support throughout my course.
Bad feedback	INCORRECT COURSE MATERIAL - The course would often provide guidelines that were needed to complete assignments for the course then assessors would fail the assignment for using their guidelines.5. The course highlights once you get a free nail kit with the course after paying at least 50% for the course which is fine by me. Daughter found course information on this company's website to be in-correct and misleading; customer service advisor also gave in-correct course information. This is the first time I have studied through a distant learning course and the materials you receive are enough to get you through the course. It was a really good course and the instructor was flexible with being able to complete the required course work on time. Biggest mistake I made was going with this course provider , I had to pay about £600 for my course plus pay for exam fees every time . Material, course content, late exam I liked the training delivery and also the set up however the course did feel a little rushed.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
Avado	Thoroughly enjoyed doing my course via Avado The course material was excellent, the portal looked very professional and I had great tutor support throughout my course. honest and easy to understand process - would recommend Alison was really helpful with booking the course Great learning and experience and online set up! The course advisors at Avado have been extremely helpful, and all of the course information sent through so far has been very informative, clear and easy to understand. So easy to start a course, well explained, course is easy to work your way through Very helpful and informative. Excellent communication and efficient service Great customer service David was a great explaining the course and making the enrolment process really clear and easy. INCORRECT COURSE MATERIAL - The course would often provide guidelines that were needed to complete assignments for the course then assessors would fail the assignment for using their guidelines.5. Really easy process, with very helpful account managers I didn't start the course yet, so I cannot say anything about the course.
Avado Good	honest and easy to understand process - would recommend Alison was really helpful with booking the course Great learning and experience and online set up! Excellent communication and efficient service Great customer service David was a great explaining the course and making the enrolment process really clear and easy. The course advisors at Avado have been extremely helpful, and all of the course information sent through so far has been very informative, clear and easy to understand. So easy to start a course, well explained, course is easy to work your way through Very helpful and informative. This course was really simple to set up, great communication, the course looks great and the site looks easy to navigate round. Great learning systems Very easy sign up process, fully explained the course and seemed very knowledgable about the course. Avado staff were very helpful in explaining the course sign up process and how the course is structured.
Avado Bad	INCORRECT COURSE MATERIAL - The course would often provide guidelines that were needed to complete assignments for the course then assessors would fail the assignment for using their guidelines.5. This is the first time I have studied through a distant learning course and the materials you receive are enough to get you through the course. Secondly I received far too many course sales emails even after I had signed up to a course. This is frustrating with Avado.This is sold as a fully flexible course with online course material and tutors that are readily available. Ruth Walczak, studying online CIPD level 5 in HR Management the support team have a very poor response I am a current student of Avado studying the CIPD diploma. When I sent emails to cancel Avado informed that I have to pay the course in FULL, despite the fact that I attended no classes and used no materials! Poor customer feedback and student support Avado so friendly and made me feel at ease when deciding what course to do.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
BPP	To anyone doing ACA exams and thinking of studying with BPP, especially towards the later or more harder ACA exams, I would strongly advise you to not use BPP. Which takes even more time and there is always a risk you may never get this second feedback due to how poor BPP is run operationally. There were also many other parts of the course that were overlooked, this has happened with other courses too at BPP I have been on at BPP. BPP give false hope to students (inc. letting students with a 2.2 onto the course, who have almost no chance of getting a TC at any City firms). To anyone reading this review I was hoping it would give you some insight into how my experience studying for my ACA exams at BPP has been. 7 phone calls waiting on hold for a long time for them to finally send my study material, by which time I had two lectures without any material. The library facilities are very poor and the tutors unresponsive to correspondence - emails seem to get conveniently 'lost' in the BPP system!
BPP Good	N/A
BPP Bad	To anyone doing ACA exams and thinking of studying with BPP, especially towards the later or more harder ACA exams, I would strongly advise you to not use BPP. Which takes even more time and there is always a risk you may never get this second feedback due to how poor BPP is run operationally. There were also many other parts of the course that were overlooked, this has happened with other courses too at BPP I have been on at BPP. BPP give false hope to students (inc. letting students with a 2.2 onto the course, who have almost no chance of getting a TC at any City firms). To anyone reading this review I was hoping it would give you some insight into how my experience studying for my ACA exams at BPP has been. 7 phone calls waiting on hold for a long time for them to finally send my study material, by which time I had two lectures without any material. The library facilities are very poor and the tutors unresponsive to correspondence - emails seem to get conveniently 'lost' in the BPP system!
Coursera	Some Courses are good, some are really bad.I enrolled in some courses with my college email id to get premium courses free. I would say that one Coursera course corresponds roughly to half of one high-level university course. I know a lot of work goes into creating courses, but without considered course delivery and implementation the course is little more than chunked information. Used to be good when they offered course materials free, but not you can even see the materials for only one free week. I was particularly interested in photography courses, and there are a lot of them.If I have free time, I use it to view the course on this platform. I tried a free course just to learn the info, and I'm so glad I don't have to take a "real" course from them. I have previously considered to study the UOL computer science degree course that UOL had liaise with Coursera.Just recently I tried to test out Coursera Free trial.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
Coursera Good	Really enjoy my marketing courses on Coursera The coursera is an excellent learning platform. I love the big selection of online courses and the fact that different universities offer their courses on their website. Coursera is an amazing organization that provides hundreds of different courses on a variety of different topics that you can take online for free! I really like Coursera courses, it's just a great way to learn new skills without leaving your home. It's amazing how many courses are available and I think I've been tapping around in at least 25 courses by now. I was particularly interested in photography courses, and there are a lot of them.If I have free time, I use it to view the course on this platform. I would say that one Coursera course corresponds roughly to half of one high-level university course.
Coursera Bad	Used to be good when they offered course materials free, but not you can even see the materials for only one free week. Bad, bad, bad... terrible company, won't offer me a refund even though it just renewed, dont use They charged me double for the same course, without my permission!Scam!! I know a lot of work goes into creating courses, but without considered course delivery and implementation the course is little more than chunked information. Any subscription that starts with a free trial is subsequently non-refundable, so unless you finish the course in 7 days and cancel, your money's gone. Some Courses are good, some are really bad.I enrolled in some courses with my college email id to get premium courses free. I tried a free course just to learn the info, and I'm so glad I don't have to take a "real" course from them. I took a free trial course and thereafter discovered I was being charged £60 a month subscription without my knowledge.
Deloitte	The most dishonest unprofessional company I've ever been involved with Very bad company Great study on private equity valuations. We were working together on tax matter. It was a great experience. They are very reliable and updated. They are reliable and professional. I've had the opportunity to work with them.
Deloitte Good	Great study on private equity valuations. We were working together on tax matter. They are very reliable and updated. It was a great experience. They are reliable and professional. I've had the opportunity to work with them.
Deloitte Bad	The most dishonest unprofessional company I've ever been involved with Very bad company

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
DPG plc	Sean was really helpful and responded to my questions promptly Amazing service !!! Easy, friendly, efficient - fab Advisor was helpful and responded really promptly! Great service, easy to use and great advisors, what more could anyone ask for Rodin was quick to respond to any questions I had and was helpful and informative. Very helpful support as I have already achieved some modules so needed information Sean was very helpful and the course was a great price ! Great service Knowledgable- kind - friendly Really helpful, reassuring and encouraging discussions with DPG following my initial enquiry about continuing with my CIPD studies. Quick and Easy to sign up with DPG Easy enrolment process, looking forward to starting my HR course. thanks DPG :) Super helpful, knowledgeable and easy to enrol, highly recommend The DPG team were always happy to help and were quick to respond. I haven't started my course yet be the advice and service I have been provided with when choosing and purchasing a course was great.
DPG plc Good	Great service, easy to use and great advisors, what more could anyone ask for Rodin was quick to respond to any questions I had and was helpful and informative. Great service Knowledgable- kind - friendly Really helpful, reassuring and encouraging discussions with DPG following my initial enquiry about continuing with my CIPD studies. thanks DPG :) Super helpful, knowledgeable and easy to enrol, highly recommend The DPG team were always happy to help and were quick to respond. Very helpful support as I have already achieved some modules so needed information Sean was very helpful and the course was a great price ! Quick and Easy to sign up with DPG Easy enrolment process, looking forward to starting my HR course. Really looking forward to getting started The Sales Manager (Calire Smeaton) was great help, answered all my questions with smile, Thank you :) Excellent communication, quick start. Friendly, clear and good guidance and very efficiently arranged my registration to start the level 5 HR Diploma course Just registered for CIPD Level 5 Diploma in HRM.
DPG plc Bad	I enquired about workshop and exam days prior to booking the course and paying my huge costs as I knew that I would be getting married. I would still like to attend the March group and feel that I should not have to suffer due to the poor service of the consultants representing your company. Unfortunately, their 3 weeks period of waiting to receive assessment results is really frustrating especially as other providers are able to mark assessment within just a few hours! I purchased the level 5 diploma and after I completed the course, I was issued with the certificate. The system also isn't the best and I had issues from time to time with it and had to call up and request support with errors. The worst experience in a course, read before enrol, please! I've got so far as to sign up for a course and that process has been satisfactory.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
edx	It was great experience of learning at edx with current updated educationcal system I have done 2 courses via edX and have had a very positive experience. EdX is a fantastic learning platform After ten years of not having studied an online course, I've decided to join a certification program over the Edx platform. hands down It was a good experience Great experience, learned a lot I had a great experience with my edx courses. edX is a great way to keep on learning new skills Great e-learning platform with great quality content! I have experience to join edx really gud nd nice im soo happy for that thnks soo much edx Great Course. Edx is the future world of learning .. good luck EDX, the best way to base your opinion on knowledge. Great learning experience, course material is easy to follow and supporting materials help with further understanding Edx has been my website of choice for personal development.
edx Good	It was great experience of learning at edx with current updated educationcal system I have done 2 courses via edX and have had a very positive experience. edX is a great way to keep on learning new skills Great e-learning platform with great quality content! hands down It was a good experience Great experience, learned a lot I had a great experience with my edx courses. EdX is a fantastic learning platform After ten years of not having studied an online course, I've decided to join a certification program over the Edx platform. I have experience to join edx really gud nd nice im soo happy for that thnks soo much edx Great Course. hii,am fell down in mathematics,this edx give best material.ok.am feel like fly with mathematics.edx provide best material with application and management time High quality videos, great course content. Edx is the future world of learning .. good luck EDX, the best way to base your opinion on knowledge.
edx Bad	Dishonest business practicesI paid for a verified certificate for a course (300usd) and I did not have time to complete the course so I did not pass. It was a really good course and the instructor was flexible with being able to complete the required course work on time. Other online learning platforms are better than edX as it allows people to work at their own pace for a certificate, even after missed course deadlines. It is amusing that a computer science course had problems showing the problems.The technical support must be improved or I will choose not to take courses on EdX. HardvardX confirmed I completed the course and sent my info to edX, but edX insisted I work with HarvardX to resolve. The last course I attempted - got half way through then - bammo - couldn't submit work for assessment unless I paid for a Verified Certificate.Come on edX. I even tested other courses (work fine) and I even created a test-account, enrolled to this course and could successfully click "Upgrade to Verified".

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
Firebrand Training	Firebrand do offer good courses and accelerated training, I have had 3 courses with Firebrand over the course of my apprenticeship. Excellent training course. Good trainer and environment Excellent training and I would highly recommend Firebrand training. I can for sure say the Firebrand course I attended was the best training course I have been on. Best Training company ever, nice and friendly staff, thank you An excellent training facility.Firebrand are my first choice for training due to location and quality of trainers. Good Tutors, Nice classrooms, Good course and revision sessions Very pleased with the last training undertaken at the wyboston lakes training centre. Material, course content, late exam I liked the training delivery and also the set up however the course did feel a little rushed.
Firebrand Training Good	Good trainer and environment Excellent training and I would highly recommend Firebrand training. Excellent training course. Best Training company ever, nice and friendly staff, thank you An excellent training facility.Firebrand are my first choice for training due to location and quality of trainers. Good Tutors, Nice classrooms, Good course and revision sessions Very pleased with the last training undertaken at the wyboston lakes training centre. Good quality fast training Access to class rooms, catering, tutoring and accommodation made for a great experience...I would definitely Firebrand again in the future. Great course, great teaching, materials could have been sent out earlier in advance great location with good resources, would recommend to anyone i know for future needs. The best way to pass CISSP Excellent accelerated training from instructors who know their stuff at Firebrand's dedicated training centre.
Firebrand Training Bad	this course is good if u have worked with Routers & Switches before and just want the Certification. rse did feel a little rushed. Firebrand do offer good courses and accelerated training, I have had 3 courses with Firebrand over the course of my apprenticeship. I felt the course was more geared to passing the exam instead of learning the content.My recommendation if you are attending a Firebrand course. Other training providers offer anything between 3 to 5 days for such a course, whereas Firebrand only trained us for 1.5 days (the afternoon is for the exam). I asked Firebrand to send me the books for the course 6 months before my course date so I could study them. Hello I did a SCCM course with Firebrand which is normally a 9 day course if you want to learn it properly. Course is hard for new people with 0 knowledge,
FutureLearn	For those of us lucky enough to have about four hours a week to spend learning, future learn offers many many courses in all subjects. I previously did such courses in other platforms, my experience, I really regret signing up the course. The courses my daughter has done on it taught her valuable practical skills through her course tasks. I have completed about 4 free courses on future learn relating to education and environment. upon upgrade the some courses they will denied you to get the certificate with argument (you did not pass the test). They are refusing to refund this amount because I continued a free course which I had started prior to subscribing. FutureLearn offer many free online courses.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
FutureLearn Good	For those of us lucky enough to have about four hours a week to spend learning, future learn offers many many courses in all subjects. I have completed about 4 free courses on future learn relating to education and environment. The courses my daughter has done on it taught her valuable practical skills through her course tasks. FutureLearn offer many free online courses. I have found all courses professional, well presented and allow for discussion amongst fellow students about the topic presented. I have done over 10 courses here and the quality is pretty high while the information is very easy to understand. I really love the culture and language courses here, especially those ones provided by the British Council.
FutureLearn Bad	I previously did such courses in other platforms, my experience, I really regret signing up the course. upon upgrade the some courses they will denied you to get the certificate with argument (you did not pass the test). They are refusing to refund this amount because I continued a free course which I had started prior to subscribing. Unfortunately if YOU HAVE TAKEN THE TEST OR RECIEVED A DIGITAL / ORIGINAL PAPER CERTIFICATE you can not claim back the money!!! Paid for a course did it and passed test. Go to any market broker and tell them that you would like a job because you know what the definition of their job is. Guys It is Important to be absolutely sure what you get out of the courses you do before paying in this website.
General Assembly	I wasn't a student, I was a teacher here.
General Assembly Good	I wasn't a student, I was a teacher here.
General Assembly Bad	N/A
ICS Learn	Really excited to work on with this course Good Course & Helpful staff.Very easy and stress-free sign up, all information provided to make easy to set up and login. My course advisor (Anne Marie) was fantastic and very helpful, all my questions were answered well and promptly.I will be recommending ICS for sure and without hesitation.Thank you, ICS. I have now enrolled on an ICS course and would have no hesitation in recommending ICS and Melissa to anyone looking for a distance learning course. Would always recommend ICS learn Easy booking process & Colin was really helpful Very helpful and informative. I called/emailed ICS many times enquiring about the course, I always spoke to Ciaran, he was very helpful and gave clear, useful information about the course. Would highly recommend Very informative, friendly and gave good advice ICS Learn is a brilliant course, the tutors are very helpful and respond fast. Very helpful adviser, easy monthly payment plan for a course I have wanted to do for years I had some queries before joining the CIPD course with ICS.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
ICS Learn Good	Really excited to work on with this course Good Course & Helpful staff.Very easy and stress-free sign up, all information provided to make easy to set up and login. Very easy, knowledgeable and helpful experience Very helpful and provided all the information I needed Really helpful the team is very friendly and helpful. My course advisor (Anne Marie) was fantastic and very helpful, all my questions were answered well and promptly.I will be recommending ICS for sure and without hesitation.Thank you, ICS. Would always recommend ICS learn Easy booking process & Colin was really helpful Very helpful and informative. I have now enrolled on an ICS course and would have no hesitation in recommending ICS and Melissa to anyone looking for a distance learning course. Would highly recommend Very informative, friendly and gave good advice ICS Learn is a brilliant course, the tutors are very helpful and respond fast. I called/emailed ICS many times enquiring about the course, I always spoke to Ciaran, he was very helpful and gave clear, useful information about the course.
ICS Learn Bad	I have paid tuition fees to ICS therefore ICS must take responsibility to ensure that students get their certificate upon completion of the course. Biggest mistake I made was going with this course provider , I had to pay about £600 for my course plus pay for exam fees every time . I have been studying with ICS Learn for two months now and I must say the amount of errors in the course material is really poor. I would give this course a 1* The course material is easy to follow and understand as well as being interesting and engaging. All was explained correctly about paying the course but was left not knowing how to go forward after paying my money as in starting the course. I paid the money to quit the course just to get rid of this improper, unprofessional, poorly structured, demotivating and disappointing course and centre. I have paid nearly £1800 for this course, I have reviewed the course material once and instantly knew I wouldn't be able to do it.
LearnDirect	I did my course a while back, very efficient service, everyone was helpful and got through my course very quickly Had to cancel my course due to financial problems. Really helpful, easy to enrol, changed my course before I started, this was no hassle at all... Great course, really informative, loads of help. Looking forward to completing my course. Brilliant course very helpful customer service team would recommend thanks Course content was outdated causing me to have to re-do a lot of the work. so far so good everything about the learn direct is perfect Great experience,great start up and materials.really happy Lovely course and very helpful customer service. Really helpful and efficient Thankyou x Awesome training courses would highly recommend and so easy to use Absolute easy to learn full support given throughout the course. It's great to know that someone from learn direct are always available to answer any questions Brilliant course very informative and great to study and do the course .

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
LearnDirect Good	Really helpful, easy to enrol, changed my course before I started, this was no hassle at all... Great course, really informative, loads of help. I did my course a while back, very efficient service, everyone was helpful and got through my course very quickly Had to cancel my course due to financial problems. so far so good everything about the learn direct is perfect Great experience,great start up and materials.really happy Lovely course and very helpful customer service. Looking forward to completing my course. Thank you. Only just started the course but very easy to assess the course website is very easy to use Really enjoying my experience so far! It's great to know that someone from learn direct are always available to answer any questions Brilliant course very informative and great to study and do the course .
LearnDirect Bad	The course highlights once you get a free nail kit with the course after paying at least 50% for the course which is fine by me. Daughter found course information on this company's website to be in-correct and misleading; customer service advisor also gave in-correct course information. What would make the difference rather than just taking your money, Learn direct could run a course on customer service and listening skills then go on it themselves. The course included written theory only, with poor diagrams and nothing in the way of videos or examples (essential I would say on a massage course)! Here is a copy of the email:I can confirm that we have received your request to cancel your course, and thus your course cancellation has been approved. Enjoy doing the course but teachers take long to reply and they dont help out as much I signed up a month ago and did not receive my course. Asked for a call back nothing ...booked online appointment nothing back terrible service I enrolled on 2 A levels and 1 maths course about 10 days ago.
Linkedin Learning	Premium service helps even more I also experienced the missing contact possibility for customer service. I improved my CV many times with no success (even paid for it)....Fully utilising LinkedIn made the difference. When a paid service gets renewed you have to send a reminder message!Feels like a scam: not even a thank you email after getting a new payment from you. Years ago, I thought LinkedIn was quite good.Two months ago, I would have said it was Facebook with a business bias - i.e. If LinkedIn wants me to pay for it, provide 99.9% decent content, with some relevant ads, and I might want to pay a small sum to remove the ads. I will avoid any LinkedIn paid service from now on like the plague. And also, had you had customer service via email, you wouldn't have had this review at this point.
Linkedin Learning Good	I improved my CV many times with no success (even paid for it)....Fully utilising LinkedIn made the difference. indeed, cv library, always provide your LinkedIn page, so they realise you are endorsed and recommended. Most people do not how to get the most of LinkedIn. Premium service helps even more Great network !!!!! Focus on skills, request skills endorsements and recommendations. Great for networking and reaching out to old work mates Most important features is CV tool.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
Linkedin Learning Bad	When a paid service gets renewed you have to send a reminder message!Feels like a scam: not even a thank you email after getting a new payment from you. If LinkedIn wants me to pay for it, provide 99.9% decent content, with some relevant ads, and I might want to pay a small sum to remove the ads. I have spent an hour out of my valuable time trying to find an email address or way of contacting customer support with no success. Years ago, I thought LinkedIn was quite good.Two months ago, I would have said it was Facebook with a business bias - i.e. And also, had you had customer service via email, you wouldn't have had this review at this point. I will avoid any LinkedIn paid service from now on like the plague. You're supposed to email a generic address and wait for however long it takes for someone to come back to you (on email only, they won't call you).
PluralSight	As a software developer and the technology change very fast and pluralsight helped me to learn fast also the latest technologies Pluralsight has excellent courses and great tutors. I was trainsignal user so i expect more system administration courses, currently it appears more application development courses are released I really love the courses available on Pluralsight. The only problem is that I have not enough time to watch all courses :-) High quality courses, great authors.I Recommend Pluralsight to anyone who is interested in development. Pluralsight is great learning portal to get started to any technological courses you would like to pursue and get insight on that technology. Good courses with valuable content I had more than two years taking courses in Pluralsight. Great content to learn and all my free time is now dedicated to go through Pluralsight courses which is awesome. Great training for developers and IT Pros The best source of tech courses Pluralsight has it all, from learning graphics, development or operations.
PluralSight Good	As a software developer and the technology change very fast and pluralsight helped me to learn fast also the latest technologies Pluralsight has excellent courses and great tutors. The only problem is that I have not enough time to watch all courses :-) High quality courses, great authors.I Recommend Pluralsight to anyone who is interested in development. I was trainsignal user so i expect more system administration courses, currently it appears more application development courses are released I really love the courses available on Pluralsight. Excellent courses for my needs as developer Pluralsight is awesome, for me the video courses hits the right spot with clear concise and well presented videos on the subject. Pluralsight is great learning portal to get started to any technological courses you would like to pursue and get insight on that technology. Good courses with valuable content I had more than two years taking courses in Pluralsight. Great training for developers and IT Pros The best source of tech courses Pluralsight has it all, from learning graphics, development or operations.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
PluralSight Bad	1. Removing ITIL contents from Pluralsight really made me to thing that courses are not permanent in pluralsight and it will go away anytime. For example I am looking for ”mobile development courses” but Pluralsight does not help me to choose courses from beginner to expert. Pluralsight provides extremely unfriendly support, it’s good to know there are many alternatives that work just as good or even better that do value their customers. I am currently trying to get my money back and have already cancelled my 30 day trial through my work account and will look at other training providers. PluralSight did NOT notify me of an annual subscription renewal prior to charging me, like other companies who undertake ethical business practices.
QA ltd	Value for Money!I have completed multiple AXELOS Foundation & Practitioner courses with both QA and also alternative training providers and would highly recommend QA for choice. I have taken 2 courses with QA (P3O and PRINCE2) both really enjoyable 1 week intense course (each). Course OK, typical mass produced training, delivered with the sole aim of passing the exam and avoiding any broader discussion of the topic: 3*. QA kept me informed and answered any queries that I raised promptly.I would highly recommend completing training with them. Good quality training - good trainers and well designed course materials. No answer from admin BCS Business Analysis Practitioner Certificate classroom course & exam. Unfortunately I cannot recommend this organisation.Update (14th oct): I’ve sent you my email Would recommend avoiding QA for Apprenticeships.
QA ltd Good	Value for Money!I have completed multiple AXELOS Foundation & Practitioner courses with both QA and also alternative training providers and would highly recommend QA for choice. I have taken 2 courses with QA (P3O and PRINCE2) both really enjoyable 1 week intense course (each). Good quality training - good trainers and well designed course materials. QA kept me informed and answered any queries that I raised promptly.I would highly recommend completing training with them. I had a great experience taking the Learn to Code-Using JavaScript course. Overall my experience of their courses has been far superior to cheaper alternative companies offering the same courses. Normally I find remote learning a bit challenging, but it worked really well.
QA ltd Bad	No answer from admin BCS Business Analysis Practitioner Certificate classroom course & exam. Exam results delayed by 5 weeks as QA left them in an unstaffed office (blamed Covid): -1*. Unfortunately I cannot recommend this organisation.Update (14th oct): I’ve sent you my email Would recommend avoiding QA for Apprenticeships. I recently attended a 3 day BCS Foundation Certificate in Business Analyst provided by this company recently. Numerous emails sent with no reply with a claim that they’ll get back to you within 48 hrs. Terrible workplace and mixed messages about support available any references to this were only on the phone so nothing in writing. Unprofessional teachers, worse training programme ever.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
Udacity	Great course! It's a great program, I've learned more and more useful information that makes me feel like I'm progressing towards a professionalism The program's great so far. thnks for every thing The Data Analyst nanodegree program is designed well with a good practice through the course which is the perfect way to learn, highly recommended. Thank you! It was awesome Great program Good program Amazing course. I am also thankful for the project reviewer, who encouraged me to learn new skills and explore best practices for Android Development.Thanks Udacity ;) Great program. It was so good..knowledgeable With my experience of learning from different sources, Udacity is really top with clear explanation and real time project work.
Udacity Good	Udacity is the Best It was an awesome experience with Udacity, I took a great decision when I choose Udacity for my React Project Learning. Thank you! It's a great program, I've learned more and more useful information that makes me feel like I'm progressing towards a professionalism The program's great so far. thnks for every thing The Data Analyst nanodegree program is designed well with a good practice through the course which is the perfect way to learn, highly recommended. I am also thankful for the project reviewer, who encouraged me to learn new skills and explore best practices for Android Development.Thanks Udacity ;) Great program. It was awesome Great program Good program Amazing course. Great course!
Udacity Bad	I took one course 3 years ago and didn't like it, I took another course recently but they are just getting worse. Course: DevOps I have recently completed two Udacity "nanodegree" courses: UX Design and Product Management. The refund policy: You have only two days to decide if you want to continue doing the course or not. I wanted to browse the course without the fear of not being able to get the full refund. Ultimately, as interesting as the course seems, the inability to receive help made me ask for a refund. The support just cared about their kpis to just make people click through and finish the course while promising to help afterwards. I have done a previous IT degree - the course is FAR TOO EXPENSIVE and the quality of teaching for the paid courses are terrible.
Udemy	Sriranga Amazing, I can't get enough courses Some courses are very expensive. I have been using Udemy for long time and am very happy with the courses and the customer service.The courses are for lifetime, always updated and never expire. I have done many courses on udemy, some courses are good and some are bad, that isn't udemys fault. I think Udemy is getting too crowded with cheap marketing as it's getting to difficult to find a really good, valuable course Their courses are incredibly cheap!!!

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
Udemy Good	Sriranga Amazing, I can't get enough courses Thank you to Udemy for a professional fabulous looking site that works. The Udemy Courses are very good, I can say that Udemy is the Internet University, where everybody have to go in order to learn whatever at very cheaper prices. I have done many courses on udemy, some courses are good and some are bad, that isn't udemys fault. I have been using Udemy for long time and am very happy with the courses and the customer service.The courses are for lifetime, always updated and never expire. I am personally benefitted from the udemy courses by Experienced Instructors who are delivering their online courses.
Udemy Bad	I think Udemy is getting too crowded with cheap marketing as it's getting to difficult to find a really good, valuable course Their courses are incredibly cheap!!! Don't waste your time putting a course on udemy as you will get fake reviews before you even start from competitors and udemy will do nothing about it. I looked elsewhere and found a course by 'quicked courses' that was free and actually better than the paid one by Udemy!!! I have taken about 5 course on Udemy and only one graphics design course stood out. Just think about this, my mate put on sale course on Udemy, and from every sale they got only 5 Euro, and Udemy take the rest! Also, sometimes when access the same course using different browsers in private / incognito mode, I get different prices for the SAME COURSE. Great website/platform but the course quality is just not good enough especially with other companies giving free online courses Pretty much what you would expect from a facebook ad.
Whitehat Jr	Amazing platform for kids to learn code...I am the coding instructor in this platform... Work culture is really good WhiteHatJr. Whitehat Jr is the best platform for kids where they can learn coding right from the very small ages which probably gonna reflect a great future of the kids... New challanges and new learning opportunity, work from home, flexible timing, mentor guidance, best technical support team and relationship manager Working days are flexible and support system is excellent. She is very good teacher Whitehat Jr is good platform for the kids to learn coding and be ready in advance for future technologies. Whitehatjr is the best platform to work and best platform for the kids as well to learn coding here. Whitehatjr is one of the best online platform which provides opportunity to women to work and earn while staying at home with flexible working hours and highest paid renumeration. Best opportunity for computer science graduate for teaching profession, huge benefit of earning and flexi hours, complete work from home Great place to work with as a teacher.

Dataset	Weighted Word Occurrence (Without Lemmatization) Summary
Whitehat Jr Good	Definitely, the best paying teaching job if you work hard and can understand computers and basic coding. Great to work with WhiteHat Jr as it gives a nice platform to educate students about coding and make out great programmers. WhiteHat Jr provides you with flexible work timings- you can choose your work schedule, good place to learn and share/ educate young minds with the knowledge you perceive. New challanges and new learning opportunity, work from home, flexible timing, mentor guidance, best technical support team and relationship manager Working days are flexible and support system is excellent. Amazing platform for kids to learn code...I am the coding instructor in this platform... Work culture is really good WhiteHatJr. Whitehatjr is one of the best online platform which provides opportunity to women to work and earn while staying at home with flexible working hours and highest paid renumeration. Whitehatjr is the best platform to work and best platform for the kids as well to learn coding here.
Whitehat Jr Bad	I figured they wanted to make sure the student was going to show up and the teacher's time will not be wasted.My child enjoyed the class. I enrolled my daughter for the trial class and at the end of the class she said she doesn't like coding. The overall experience of WhiteHat Jr coding program is rated as average as the entire experience is dependent on the kind of teacher gets allocated to the student. We need the student or child to understand why and what he is doing ... it's not class to class teaching which is important... In the time since, I have received four emails, five text messages, and three phone calls reminding me of the upcoming class. Are you still want to "suggest" how my feedback should look?It's very annoying that a company like WhiteHat Jr pretends to control the way customer leave feedback.Wake up guys! First of all, we got several messages before the class even started all from different random numbers labeled "spam likely" on my phone-not a good sign.

Table B.1: Feedback Summarisation Results Using Weighted Word Occurrence Method (Without Lemmatization)

Appendix C

Data Collection Script

Data Collection

In [5]:
```
import re
from bs4 import BeautifulSoup
import mechanize
import http.cookiejar as cookielib
import json
import pandas as pd
import time
from pandas.io.json import json_normalize

cookiejar =cookielib.LWPCookieJar()
```

First, identifies the Trustpilot URL for each company

In [ ]:
```
competitor_url = {
    'Avado':'https://uk.trustpilot.com/review/avadolearning.com',
    'ICS_Learn':'https://uk.trustpilot.com/review/icslearn.co.uk',
    'DPG_plc':'https://uk.trustpilot.com/review/www.dpgplc.co.uk',
    'QA_ltd':'https://uk.trustpilot.com/review/qa.com',
    'Whitehat_Jr':'https://uk.trustpilot.com/review/whitehatjr.com',
    'Firebrand_Training':'https://uk.trustpilot.com/review/firebrand.training',
    'Udemy':'https://uk.trustpilot.com/review/udemy.com',
    'Babington':'https://uk.trustpilot.com/review/babington.co.uk',
    'Baltic_Training':'https://uk.trustpilot.com/review/balticapprenticeships.com',
    'BPP':'https://uk.trustpilot.com/review/www.bpp.com',
    'Bright_Network':'https://uk.trustpilot.com/review/brightnetwork.co.uk',
    'Coursera':'https://uk.trustpilot.com/review/coursera.org',
    'Degreed':'https://uk.trustpilot.com/review/degreed.com',
    'Deloitte':'https://uk.trustpilot.com/review/deloitte.com',
    'Econsultancy':'https://uk.trustpilot.com/review/econsultancy.com',
    'edx':'https://uk.trustpilot.com/review/www.edx.org',
    'FutureLearn':'https://uk.trustpilot.com/review/www.futurelearn.com',
    'General_Assembly':'https://uk.trustpilot.com/review/generalassemb.ly',
    'Hyper_Island':'https://uk.trustpilot.com/review/hyperisland.com',
    'INSEAD':'https://uk.trustpilot.com/review/insead.edu',
    'LearnDirect':'https://uk.trustpilot.com/review/www.learndirect.com',
    'Linkedin_Learning':'https://uk.trustpilot.com/review/uk.linkedin.com',
    'McKinsey':'https://uk.trustpilot.com/review/mckinsey.com',
    'PluralSight':'https://uk.trustpilot.com/review/pluralsight.com',
    'Udacity':'https://uk.trustpilot.com/review/udacity.com'
}
```

Setup scraping script

In [3]:
```
def setup_soup_page(url):
    global br
    br = mechanize.Browser()
    br.set_cookiejar(cookiejar)
    br.set_handle_robots(False)
    br.open(url)
    response1= br.response()
    html=response1.read()
    #print(br.title())

    soup = BeautifulSoup(html,'html.parser')
    return soup

def get_reviews_from_page(soup):
    scripts=soup.find_all("script")
    for script in scripts:
        if script.get('data-business-unit-json-ld')=="":
```

```
            #print(script)
            review_script_json=script.string.strip()
            break
    review_data = json.loads(review_script_json)
    return review_data[0]['review']

def open_next_page(br_old_page):
    global br
    links=br_old_page.links()
    is_next_page=[ x for x in links if x.text=='Next page']
    if len(is_next_page) == 0:
        return
    next_page_link= is_next_page[0]
    response=br_old_page.follow_link(next_page_link)
    html=response.read()
    print(br_old_page.title())
    soup = BeautifulSoup(html,'html.parser')

    br=br_old_page

    return soup
```

For every company, iterate the scraping process. Save the data into a csv for each company.

In [6]:

```
for key in competitor_url:
    data=[]
    soup=setup_soup_page(competitor_url[key])
    data.extend(get_reviews_from_page(soup))
    soup=open_next_page(br)
    if soup != None:
        data.extend(get_reviews_from_page(soup))
        name=competitor_url[key].split('/')[-1]
        num_of_pages=int(re.search('Read Customer Service Reviews of {} \| ([0-9]*)

        for i in range(3,num_of_pages+1):
            if i % 20 == 0:
                time.sleep(5)
            soup=open_next_page(br)
            data.extend(get_reviews_from_page(soup))

    if len(data) == 0:
        print('No data:', key)
        continue

    df_temp=pd.DataFrame(data)
    drop_columns=['author','itemReviewed','publisher','reviewRating']
    df_meat=df_temp.copy().drop(columns=drop_columns)

    dfs=[]
    for column in drop_columns:
        dfs.append(json_normalize(df_temp[column]).add_prefix(column+'_'))

    df=pd.concat([df_meat]+dfs, axis=1).drop(columns=['reviewRating_worstRating','re
    df.to_csv('Dataset/competitors/trustpilot_reviews_{}.csv'.format(key))
    print(key)
```

```
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 2 of 103
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 3 of 103
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 4 of 103
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 5 of 103
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 6 of 103
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 7 of 103
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 8 of 103
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 9 of 103
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 10 of 103
Avado Reviews | Read Customer Service Reviews of avadolearning.com | 11 of 103
```

Appendix D

Data Understanding and Preparation Script

Data Understanding & Preparation

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag
import spacy
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
files = [
    'trustpilot_reviews_Avado.csv',
    'trustpilot_reviews_BPP.csv',
    'trustpilot_reviews_Coursera.csv',
    'trustpilot_reviews_Deloitte.csv',
    'trustpilot_reviews_DPG_plc.csv',
    'trustpilot_reviews_edx.csv',
    'trustpilot_reviews_Firebrand_Training.csv',
    'trustpilot_reviews_FutureLearn.csv',
    'trustpilot_reviews_General_Assembly.csv',
    'trustpilot_reviews_ICS_Learn.csv',
    'trustpilot_reviews_LearnDirect.csv',
    'trustpilot_reviews_Linkedin_Learning.csv',
    'trustpilot_reviews_PluralSight.csv',
    'trustpilot_reviews_QA_ltd.csv',
    'trustpilot_reviews_Udacity.csv',
    'trustpilot_reviews_Udemy.csv',
    'trustpilot_reviews_Whitehat_Jr.csv'
]
```

Gather all feedback into one dataframe.

In [3]:
```
df = pd.DataFrame()
for file in files:
    df_temp = pd.read_csv('Dataset/competitors/{}'.format(file))
    df = df.append(df_temp, ignore_index=True)
```

In [4]:
```
df.head()
```

Out[4]:

	Unnamed: 0	@type	datePublished	headline	inLanguage	reviewBody	author_@type
0	0	Review	2021-01-15T13:50:34+00:00	When speaking to one advisors on...	en	When speaking to one of the advisors on the ph...	Person
1	1	Review	2021-01-15T10:45:08+00:00	CIPD Level 5 Diploma	en	I enrolled today to study my CIPD Level 5 Dipl...	Person

	Unnamed: 0	@type	datePublished	headline	inLanguage	reviewBody	author_@type	
2	2	Review	2021-01-12T23:13:19+00:00	Great!	en	I'm enjoying the course that I'm doing. The st...	Person	
3	3	Review	2021-01-08T20:26:16+00:00	Very disappointed with the service	en	Very disappointed with the service I have rece...	Person	
4	4	Review	2021-01-06T04:51:35+00:00	Superb Digital Marketing course at Avado	en	The entire course module at Avado is very enga...	Person	ima

1. Data Understanding

1.1. Describe Data

Describe the data that has been acquired, including the format of the data, the quantity of data (for example, the number of records and fields in each table), the identities of the fields, and any other surface features which have been discovered. Evaluate whether the data acquired satisfies the relevant requirements.

In [5]:
```
df.shape
```

Out[5]:
```
(17575, 17)
```

The data is formatted as a csv file with 17575 rows(observations) and 17 columns(features).

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17575 entries, 0 to 17574
Data columns (total 17 columns):
Unnamed: 0                  17575 non-null int64
@type                       17575 non-null object
datePublished               17575 non-null object
headline                    17575 non-null object
inLanguage                  17575 non-null object
reviewBody                  17575 non-null object
author_@type                17575 non-null object
author_image                2465 non-null object
author_name                 17574 non-null object
author_url                  17575 non-null object
itemReviewed_@type          17575 non-null object
itemReviewed_name           17575 non-null object
publisher_@type             17575 non-null object
publisher_name              17575 non-null object
publisher_sameAs            17575 non-null object
reviewRating_@type          17575 non-null object
reviewRating_ratingValue    17575 non-null int64
dtypes: int64(2), object(15)
memory usage: 2.3+ MB
```

```
In [7]: df.nunique()
```

```
Out[7]: Unnamed: 0                  5740
        @type                          1
        datePublished              17556
        headline                   14915
        inLanguage                     1
        reviewBody                 17459
        author_@type                   1
        author_image                2457
        author_name                14385
        author_url                 17541
        itemReviewed_@type             1
        itemReviewed_name             17
        publisher_@type                1
        publisher_name                 1
        publisher_sameAs               1
        reviewRating_@type             1
        reviewRating_ratingValue       5
        dtype: int64
```

Initially, the features are as follow:

- Unnamed: 0: integer. Auto-generated index from scraping process.
- @type: string. The type of data/information in the review HTML block, which is 'Review'.
- datePublished: string. The date and time the review was published/submitted.
- headline: string. The head/summary of the review.
- inLanguage: string. The detected language used in the review.
- reviewBody: string. The body of the review.
- author_@type: string. The type of data/information in the author HTML block, which is 'Person'.
- author_image: string. The profile picture image of the author of the review, given as a URL to the image.
- author_name: string. The name of the author.
- author_url: string. The URL to the author Trustpilot profile.
- itemReviewed_@type: string. The type of data/information in the target company HTML block, which is 'Thing'.
- itemReviewed_name: string. The name of the target company.
- publisher_@type: string. The type of data/information in the publisher HTML block, which is 'Organization'.
- publisher_name: string. The name of the publisher, in this case is 'Trustpilot'.
- publisher_sameAs: string. The URL to the publisher, in this case is 'https://uk.trustpilot.com'.
- reviewRating_@type: string. The type of data/information in the ratings HTML block, which is 'Rating'.
- reviewRating_ratingValue: integer. The number of ratings given to the reviews with range between 1 and 5 inclusive.

For columns with '@type' in it, the value of each of the column is the same. For example, column 'author_@type' has only one value 'Person' for the whole dataset. Other than this, the columns 'publisher_name' and 'publisher_sameAs' have the same value for the whole dataset as well because in this case, the publisher is only 'Trustpilot'.

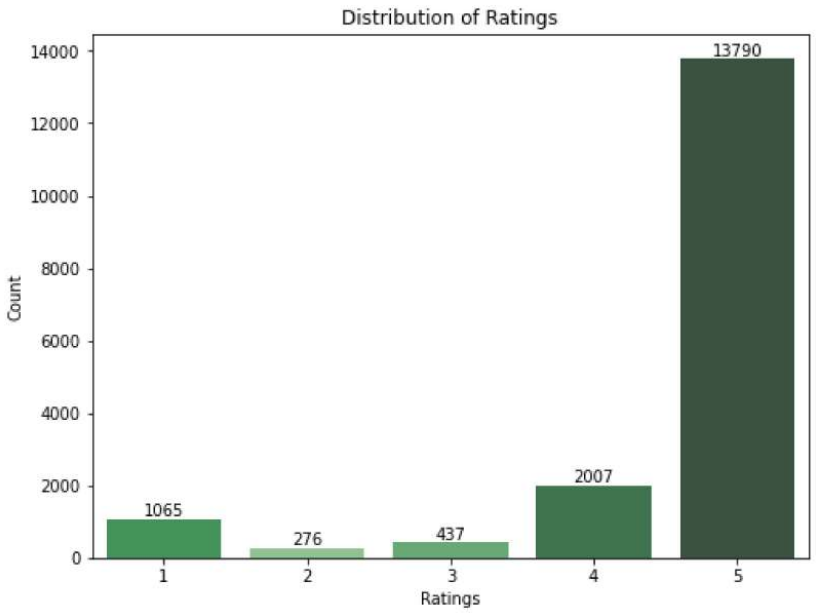
The dataset fulfills the requirements needed for this project with focus on columns 'headline', 'reviewBody', and 'reviewRating_ratingValue', which make up a complete customer feedback.

1.2. Explore Data

Describe results of this task, including first findings or initial hypothesis and their impact on the remainder of the project. If appropriate, include graphs and plots to indicate data characteristics that suggest further examination of interesting data subsets.

In [8]:
```
# Ratings count
f, ax = plt.subplots(figsize=(8,6))

groupedvalues=df.groupby('reviewRating_ratingValue').count().reset_index()

pal = sns.color_palette("Greens_d", len(groupedvalues))
pal.reverse()
rank = groupedvalues["Unnamed: 0"].argsort().argsort()

g = sns.countplot(data=df,
                  x="reviewRating_ratingValue",
                  palette=np.array(pal[::-1])[rank],
                  ax=ax
                 )

for index, row in groupedvalues.iterrows():
    g.text(row.name,row["Unnamed: 0"], round(row["Unnamed: 0"],2), color='black', ha

ax.set_title('Distribution of Ratings')
ax.set_ylabel('Count')
ax.set_xlabel('Ratings')
```

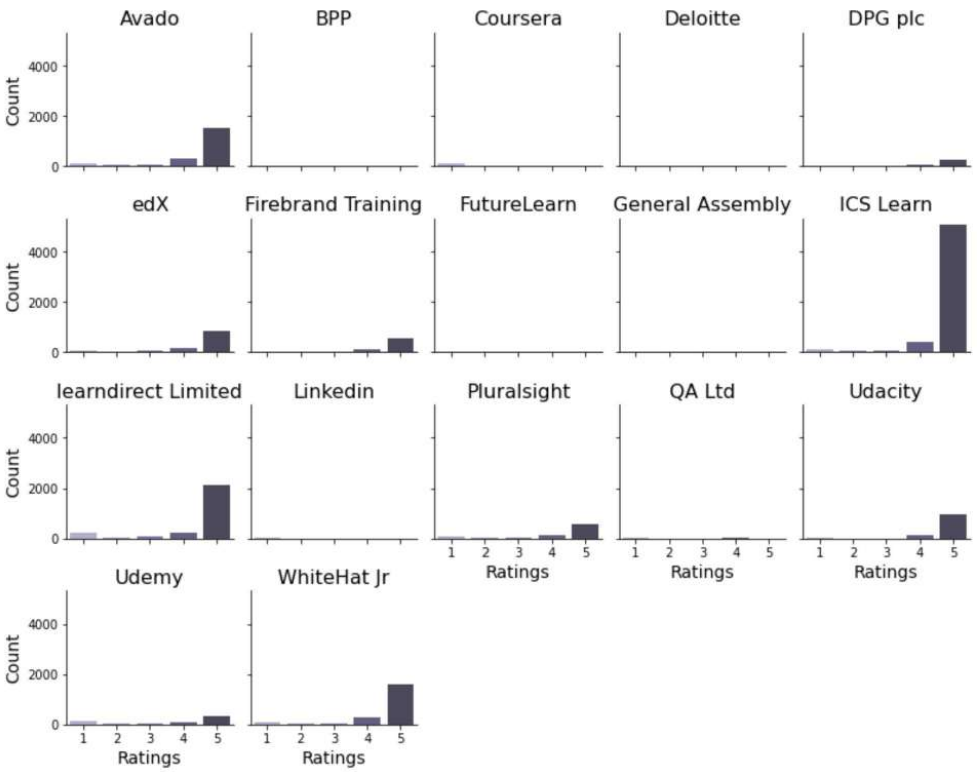
Out[8]: Text(0.5, 0, 'Ratings')



In [9]:
```
# Ratings by company
pal = sns.color_palette("Purples_d", 5)

g = sns.FacetGrid(df, # initialise FacetGrid, this will not draw the plots
                  col="itemReviewed_name",
                  height=2.2,
                  aspect=1,
                  col_wrap=5,
                  margin_titles=True
                 )
g.map(sns.countplot, "reviewRating_ratingValue", palette=pal) # Draw the plot using
```

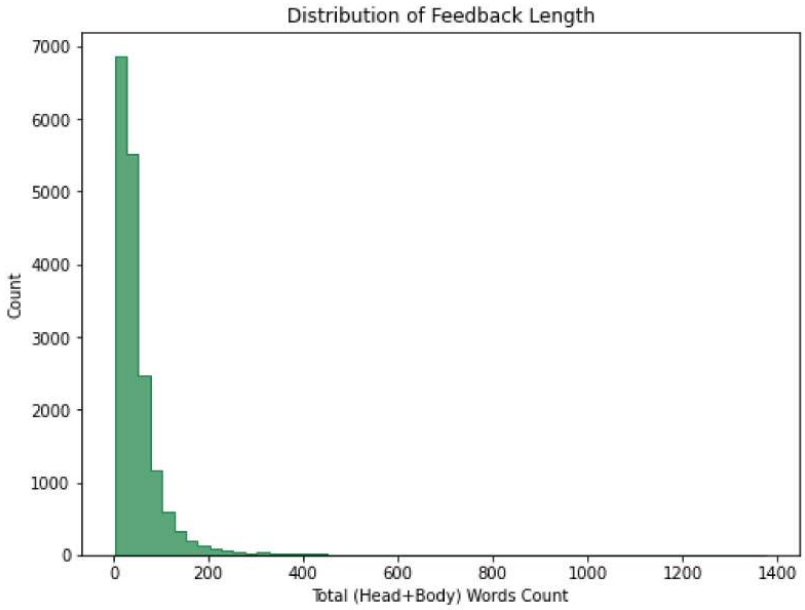
```
g.set_axis_labels("Ratings", "Count", fontsize=14)
g.set_titles(col_template="{col_name}", row_template="{row_name}", size=16)
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x2079f2d8688>



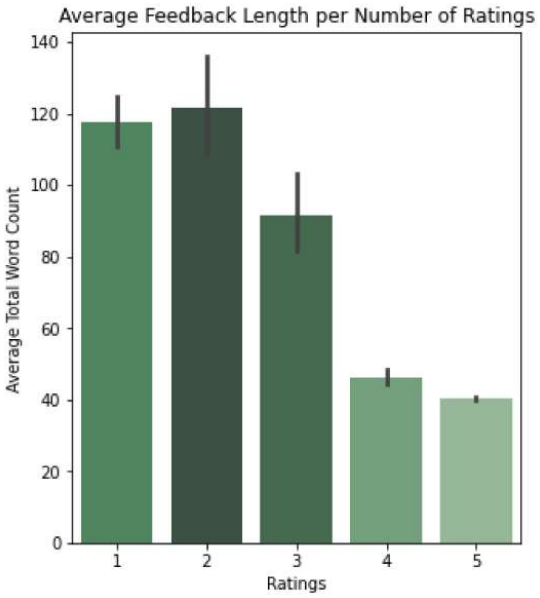
In [5]:
```
# Total (Head + Body) length by words
df['head_word_count'] = df['headline'].apply(lambda x: len(str(x).split()))
df['body_word_count'] = df['reviewBody'].apply(lambda x: len(str(x).split()))
df['total_word_count'] = df['head_word_count'] + df['body_word_count']

f, ax = plt.subplots(figsize=(8,6))

pal = sns.color_palette("Greens_d")
pal.reverse()

sns.histplot(data=df,
             x="total_word_count",
             stat="count", # 'count', 'frequency', 'density', 'probability'
             bins=20, # number of bins
             binwidth=25, # width of each bins. Override bins parameter
             element='step', # 'bars', 'step', 'poly'. Univariate only
             #palette="dark", # 'deep', 'muted', 'bright', 'pastel', 'dark', 'colorb
             #legend=True, # True or False
             color='#278a50',
             ax=ax
            )

ax.set_title('Distribution of Feedback Length')
ax.set_ylabel('Count')
ax.set_xlabel('Total (Head+Body) Words Count')
```

Out[5]: Text(0.5, 0, 'Total (Head+Body) Words Count')



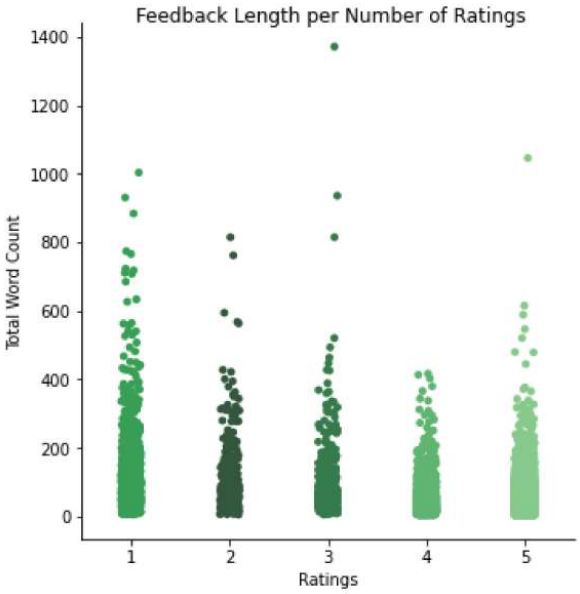
In [11]:
```
# Average Total (Head + Body) Length by ratings
f, ax = plt.subplots(figsize=(5,5.85))

groupedvalues=df.groupby('reviewRating_ratingValue').count().reset_index()

pal = sns.color_palette("Greens_d", len(groupedvalues))
rank = groupedvalues["Unnamed: 0"].argsort().argsort()

sns.barplot(data=df,
                    x="reviewRating_ratingValue",
                    y="total_word_count",
                    saturation=0.5, # float 0 to 1. Large patches often look better w
                    orient=None, # 'v' or 'h'. Use to resolve ambiguity
                    palette=np.array(pal[::-1])[rank], # 'deep', 'muted', 'bright', '
                    ax=ax
                   )

ax.set_title('Average Feedback Length per Number of Ratings')
ax.set_ylabel('Average Total Word Count')
ax.set_xlabel('Ratings')
```

Out[11]:
```
Text(0.5, 0, 'Ratings')
```



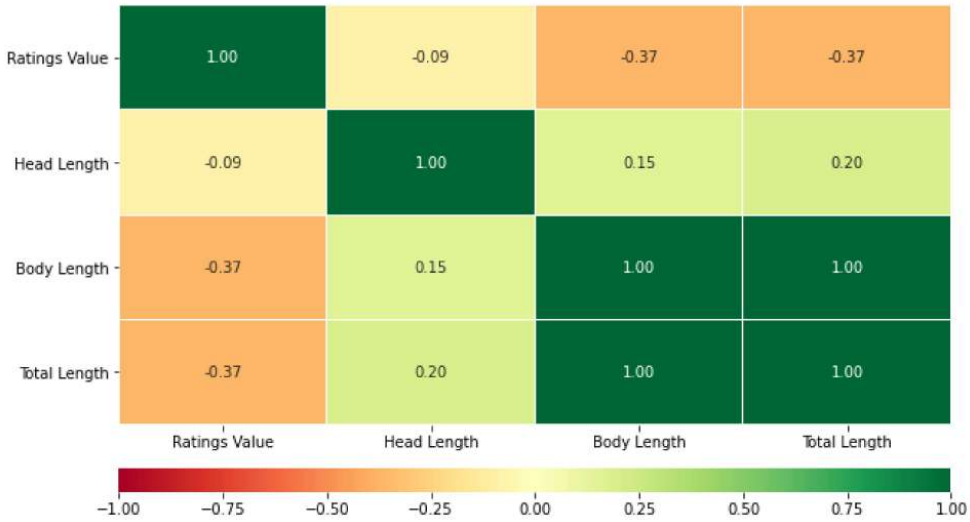
In [12]:
```
# Total (Head + Body) length by ratings

groupedvalues=df.groupby('reviewRating_ratingValue').count().reset_index()

pal = sns.color_palette("Greens_d", len(groupedvalues))
rank = groupedvalues["Unnamed: 0"].argsort().argsort()

g = sns.catplot(data=df,
                    x="reviewRating_ratingValue",
                    y="total_word_count",
                    kind="strip", # 'strip', 'swarm', 'box', 'violin', 'boxen', 'poin
                    height=5, # height in inches for each facet
                    aspect=1, # ratio of facet. Width = height * aspect
                    palette=np.array(pal[::-1])[rank] # 'deep', 'muted', 'bright', 'p
                   )


g.fig.suptitle("Feedback Length per Number of Ratings", x=0.55, y=1)
plt.subplots_adjust(top=0.97)
g.set_axis_labels("Ratings", "Total Word Count")
```

Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x207a3350148>
```



In [21]:
```
grid_kws = {"height_ratios": (.9, .05), "hspace": .2} # set ratio of axes grid
f, (ax, cbar_ax) = plt.subplots(2, gridspec_kw=grid_kws, figsize=(10,6))

df_sliced = df[["reviewRating_ratingValue", "head_word_count", "body_word_count", "t
df_sliced.columns = ['Ratings Value', 'Head Length', 'Body Length', 'Total Length']
g=sns.heatmap(data=df_sliced.corr(),
              vmin=-1, # min value of colormap
              vmax=1, # max value of colormap
              annot=True, # If True, write data value in each cell
              fmt=".2f", # String formatting code for annotation
              linewidths=0.1, # Width of the lines that will divide each cell
              linecolor="White", # Color of the lines that will divide each cell
              cmap="RdYlGn", # The mapping from data values to color space
              cbar=True, # Whether to draw a color bar
              cbar_ax=cbar_ax, # Which axis to draw the colorbar, if not set, draw in
              cbar_kws={"orientation": "horizontal"},
              ax=ax
             )
g.set_yticklabels(g.get_yticklabels(), rotation = 0)
```

Out[21]:
```
[Text(0, 0.5, 'Ratings Value'),
 Text(0, 1.5, 'Head Length'),
 Text(0, 2.5, 'Body Length'),
 Text(0, 3.5, 'Total Length')]
```



Suggestion: need to wait for sentiment classification and see the correlation with the sentiment value as well.

1.3. Verify Data Quality

List the results of the data quality verification; if quality problems exist, list possible solutions. Solutions to data quality problems generally depend heavily on both data and business knowledge.

In [14]:
```
# Missing Values: return a dataframe with number of missing values for each feature
def check_missing_value(df):
    total = df.isnull().sum().sort_values(ascending=False)
    percent = (df.isnull().sum()/df.isnull().count()*100).sort_values(ascending=Fals
    missing_values_table = pd.concat([total, percent], axis=1, keys=['Total', 'Perce
    pd.set_option('display.max_rows', None)

    return missing_values_table
```

In [15]:
```
check_missing_value(df)
```

Out[15]:

	Total	Percent
author_image	15110	85.974395
author_name	1	0.005690
total_word_count	0	0.000000
body_word_count	0	0.000000
@type	0	0.000000
datePublished	0	0.000000
headline	0	0.000000
inLanguage	0	0.000000
reviewBody	0	0.000000
author_@type	0	0.000000
author_url	0	0.000000

	Total	Percent
itemReviewed_@type	0	0.000000
itemReviewed_name	0	0.000000
publisher_@type	0	0.000000
publisher_name	0	0.000000
publisher_sameAs	0	0.000000
reviewRating_@type	0	0.000000
reviewRating_ratingValue	0	0.000000
head_word_count	0	0.000000
Unnamed: 0	0	0.000000

Th missing values in author image and name can be ignored as this project will not use both features.

In [16]:
```
#Check input errors
np.sort(df['reviewRating_ratingValue'].unique())
```

Out[16]:
```
array([1, 2, 3, 4, 5], dtype=int64)
```

It is confirmed that the ratings value between 1 and 5 only, inclusive.

Now, manually check if there's any discrepancies in the head/body of feedback. It seems that the headline is automatically filled with several words from the body if the reviewer did not fill in manually. One characteristic of these kind of feedback head is the symbol ... attached at the end of the head. There some seemingly random feedback such as 'Xggv', emoticons such as ':)', shorten word such as 'Fab' from fabulous, typos such as 'consise' (should be 'concise').

2. Data Preparation

2.1. Data Selection

In [6]:
```
df.drop(['Unnamed: 0', '@type', 'inLanguage', 'author_@type', 'author_name', 'author
         'itemReviewed_@type', 'publisher_@type', 'publisher_name', 'publisher_sameA
         'reviewRating_@type'], axis=1, inplace=True)
```

The selected columns are chosen simply because they are needed for the project. 'headline', 'reviewBody', and 'reviewRating_ratingValue' make the whole feedback structure. 'itemReviewed_name' is needed to give comparison for business insights purpose. 'datePublished' is also included for indexing/sorting purpose. The excluded columns might be useful, but they can be explored further as a future work direction.

In [7]:
```
df.head()
```

Out[7]:

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head
0	2021-01-15T13:50:34+00:00	When speaking to one advisors on...	When speaking to one of the advisors on the ph...	Avado	3	

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head
1	2021-01-15T10:45:08+00:00	CIPD Level 5 Diploma	I enrolled today to study my CIPD Level 5 Dipl...	Avado	5	
2	2021-01-12T23:13:19+00:00	Great!	I'm enjoying the course that I'm doing. The st...	Avado	5	
3	2021-01-08T20:26:16+00:00	Very disappointed with the service	Very disappointed with the service I have rece...	Avado	2	
4	2021-01-06T04:51:35+00:00	Superb Digital Marketing course at Avado	The entire course module at Avado is very enga...	Avado	5	

2.2. Data Cleaning

In [8]:
```
# remove unwanted characters, numbers and symbols
df['headline_clean'] = df['headline'].apply(lambda text: " ".join(re.findall("[a-zA-
df['reviewBody_clean'] = df['reviewBody'].apply(lambda text: " ".join(re.findall("[a

# make entire text lowercase
df['headline_clean'] = [r.lower() for r in df['headline_clean']]
df['reviewBody_clean'] = [r.lower() for r in df['reviewBody_clean']]

stop_words = stopwords.words('english')

def remove_stopwords(rev):
    rev_new = " ".join([i for i in rev if i not in stop_words])
    return rev_new

# remove stopwords from the text
df['headline_clean'] = [remove_stopwords(r.split()) for r in df['headline_clean']]
df['reviewBody_clean'] = [remove_stopwords(r.split()) for r in df['reviewBody_clean'
```

In [9]:
```
df['reviewBody_clean'].iloc[1]
```

Out[9]:
```
'enrolled today study cipd level diploma hr management long time analyzing learning
providers even comparison excel decided study avado cheryl ball amazing support deci
sion time patient candid great knowledge avado offers looking forward start studies
melissa g'
```

In [10]:
```
df['reviewBody_clean'].iloc[10]
```

Out[10]:
```
'brilliant blend different learning styles including group work online materials liv
e webinars really enjoyed great collaborate learn fellow squares'
```

In [11]:
```
# Lemmatize the words in the feedback based on the pos tag
def get_simple_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
```

```
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    final_text = []
    for i in text.split():
        pos = pos_tag([i.strip()])
        word = lemmatizer.lemmatize(i.strip(),get_simple_pos(pos[0][1]))
        final_text.append(word)
    return " ".join(final_text)

df['lemmatized_headline'] = df['headline_clean'].apply(lemmatize_words)
df['lemmatized_reviewBody'] = df['reviewBody_clean'].apply(lemmatize_words) # withou
df['lemmatized_reviewBody_original'] = df['reviewBody'].apply(lemmatize_words) # sti
```

In [20]:
```
df['lemmatized_reviewBody'].iloc[1]
```

Out[20]:
```
'enrol today study cipd level diploma hr management long time analyze learn provider
even comparison excel decide study avado cheryl ball amaze support decision time pat
ient candid great knowledge avado offer look forward start study melissa g'
```

In [28]:
```
df['lemmatized_reviewBody'].iloc[10]
```

Out[28]:
```
'brilliant blend different learn style include group work online material live webin
ars really enjoy great collaborate learn fellow square'
```

In [12]:
```
df['lemmatized_reviewBody_original'].iloc[10]
```

Out[12]:
```
"A brilliant blend of different learn style include group work, online material and
live webinars. I really enjoy this, and it be great to collaborate and learn with my
fellow 'squares'"
```

2.3. Data Construction

In [13]:
```
# word count
df['head_word_count'] = df['headline'].apply(lambda x: len(str(x).split()))
df['body_word_count'] = df['reviewBody'].apply(lambda x: len(str(x).split()))
df['total_word_count'] = df['head_word_count'] + df['body_word_count']

# combine head and body
df['lemmatized_feedback'] = df['lemmatized_headline'] + ' ' + df['lemmatized_reviewB

# convert rating to binary class
def sentiment_rating(rating):
    # Replacing ratings of 1,2,3 with 0 (not good) and 4,5 with 1 (good)
    if(int(rating) == 1 or int(rating) == 2 or int(rating) == 3):
        return 0
    else:
        return 1

df['overall_rating'] = df['reviewRating_ratingValue'].apply(sentiment_rating)
```

In [14]:
```
df.head()
```

Out[14]:

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head
0	2021-01-15T13:50:34+00:00	When speaking to one advisors on...	When speaking to one of the advisors on the ph...	Avado	3	
1	2021-01-15T10:45:08+00:00	CIPD Level 5 Diploma	I enrolled today to study my CIPD Level 5 Dipl...	Avado	5	
2	2021-01-12T23:13:19+00:00	Great!	I'm enjoying the course that I'm doing. The st...	Avado	5	
3	2021-01-08T20:26:16+00:00	Very disappointed with the service	Very disappointed with the service I have rece...	Avado	2	
4	2021-01-06T04:51:35+00:00	Superb Digital Marketing course at Avado	The entire course module at Avado is very enga...	Avado	5	

2. Final Dataset

In [15]:
```
df.to_csv('Dataset/trustpilot_reviews.csv', index=False)
```

Appendix E

Sentiment Classification Script

The sentiment classification script attached here is for the unbiased model using balanced dataset. For the biased model using imbalanced dataset, everything is the same, except for the resampling part.

Sentiment Classification

Quick links:

In [1]:
```
def reset_seed():
    # disable GPU
    #os.environ["CUDA_DEVICE_ORDER"] = "PCI_BUS_ID"
    os.environ["CUDA_VISIBLE_DEVICES"] = ""

    # reproducibility
    seed_value=111
    os.environ['PYTHONHASHSEED']=str(seed_value)

    np.random.seed(seed_value)
    python_random.seed(seed_value)
    tf.random.set_seed(seed_value)

    config = tf.compat.v1.ConfigProto(intra_op_parallelism_threads=1, inter_op_paral
    sess = tf.compat.v1.Session(graph=tf.compat.v1.get_default_graph(), config=confi
    tf.compat.v1.keras.backend.set_session(sess)
```

In [3]:
```
import os
import numpy as np
import random as python_random
import tensorflow as tf
reset_seed()
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import imblearn
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from keras import backend as K
from tensorflow import keras # have to be tensorflow 2.2.0 or keras 2.3.1
#import keras
from keras.models import Sequential
from keras import layers, callbacks
```

```
from keras.backend import clear_session
from sklearn.model_selection import RandomizedSearchCV
from keras.preprocessing.text import Tokenizer
from keras.wrappers.scikit_learn import KerasClassifier
from keras.preprocessing.sequence import pad_sequences
from keras import metrics
from gensim.models import KeyedVectors
#from keras.models import load_model
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
df = pd.read_csv('Dataset/trustpilot_reviews.csv')
```

In [5]:
```
df.head()
```

Out[5]:

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head
0	2021-01-15T13:50:34+00:00	When speaking to one advisors on...	When speaking to one of the advisors on the ph...	Avado	3	
1	2021-01-15T10:45:08+00:00	CIPD Level 5 Diploma	I enrolled today to study my CIPD Level 5 Dipl...	Avado	5	
2	2021-01-12T23:13:19+00:00	Great!	I'm enjoying the course that I'm doing. The st...	Avado	5	
3	2021-01-08T20:26:16+00:00	Very disappointed with the service	Very disappointed with the service I have rece...	Avado	2	
4	2021-01-06T04:51:35+00:00	Superb Digital Marketing course at Avado	The entire course module at Avado is very enga...	Avado	5	

In [6]:
```
# function to map the training log into line chart
plt.style.use('ggplot')

def plot_training_log(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    x = range(1, len(acc) + 1)

    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    plt.plot(x, acc, 'b', label='Training acc')
    plt.plot(x, val_acc, 'r', label='Validation acc')
    plt.title('Training and validation accuracy')
```

```
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(x, loss, 'b', label='Training loss')
    plt.plot(x, val_loss, 'r', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()
```

1. Data Preprocessing

Back to top

Deal with imbalanced dataset.

In [7]:
```
# split dataset for binary classification using the overall rating
# split first to make sure the test set is untouched
x_train,x_test,y_train,y_test = train_test_split(df['lemmatized_feedback'], df['over
                                                 random_state = 1000) #reproducibili

# reset index
x_train.reset_index(drop=True, inplace=True)
y_train.reset_index(drop=True, inplace=True)
x_test.reset_index(drop=True, inplace=True)
y_test.reset_index(drop=True, inplace=True)
```

In [8]:
```
# Check the distribution of the label
train_good = x_train[y_train[y_train == 1].index]
train_bad = x_train[y_train[y_train == 0].index]

print('Training set size =', len(x_train))
print('Good feedback training set size =', len(train_good))
print('Bad feedback training set size =', len(train_bad))

test_good = x_test[y_test[y_test == 1].index]
test_bad = x_test[y_test[y_test == 0].index]

print('\nTest set size =', len(x_test))
print('Good feedback test set size =', len(test_good))
print('Bad feedback test set size =', len(test_bad))
```

```
Training set size = 14060
Good feedback training set size = 12615
Bad feedback training set size = 1445

Test set size = 3515
Good feedback test set size = 3182
Bad feedback test set size = 333
```

The training set is imbalanced. To make sure the information loss is minimal when downsampling the majority class, the minority class is upsampled first.

In [9]:
```
ros = RandomOverSampler(sampling_strategy={0:2890},random_state=1000)
x_ros, y_ros = ros.fit_resample(x_train.values.reshape(-1, 1), y_train)
```

In [10]:
```
# Check the distribution of the label
train_good = x_ros[y_ros[y_ros == 1].index]
train_bad = x_ros[y_ros[y_ros == 0].index]

print('Training set size =', len(x_ros))
print('Good feedback training set size =', len(train_good))
print('Bad feedback training set size =', len(train_bad))
```

```
Training set size = 15505
```

```
Good feedback training set size = 12615
Bad feedback training set size = 2890
```

In [11]:
```
rus = RandomUnderSampler(random_state=1000)
x_train, y_train = rus.fit_resample(x_ros.reshape(-1, 1), y_ros)
```

In [12]:
```
# Check the distribution of the label
train_good = x_train[y_train[y_train == 1].index]
train_bad = x_train[y_train[y_train == 0].index]

print('Training set size =', len(x_train))
print('Good feedback training set size =', len(train_good))
print('Bad feedback training set size =', len(train_bad))
```

```
Training set size = 5780
Good feedback training set size = 2890
Bad feedback training set size = 2890
```

In [13]:
```
x_train = pd.DataFrame(x_train)
x_train = x_train[0]
```

1.1. Data representation: Bag of words

In [14]:
```
# Bag of words: sparse matrix with all zeros except for a few denoting the word
# ngram_range: tuple (min_n, max_n)

count_vectorizer = CountVectorizer(min_df=0, ngram_range=(1,1)) #unigram
cv_train = count_vectorizer.fit_transform(x_train)
cv_test = count_vectorizer.transform(x_test)

print('Bag of Words Train:', cv_train.shape) # (dataset size, size of the vocabulary
print('Bag of Words Test:', cv_test.shape)

print(x_train[1])
print(cv_train[1])
```

```
Bag of Words Train: (5780, 7217)
Bag of Words Test: (3515, 7217)
course purchase sign process get far sign course process satisfactory nothing beyond
contact often delayed despite email query something post purchase yet response
  (0, 5797)     2
  (0, 2707)     1
  (0, 1471)     2
  (0, 5423)     1
  (0, 5076)     2
  (0, 4976)     2
  (0, 2409)     1
  (0, 5594)     1
  (0, 4325)     1
  (0, 699)      1
  (0, 1372)     1
  (0, 4407)     1
  (0, 1669)     1
  (0, 1736)     1
  (0, 2093)     1
  (0, 5115)     1
  (0, 5935)     1
  (0, 4853)     1
  (0, 7194)     1
```

1.2. Data representation: TF-IDF

In [15]:
```
# Weighted bag of words (TF-IDF)
# ngram_rangetuple (min_n, max_n)
```

```
tfidf_vectorizer = TfidfVectorizer(min_df=0, ngram_range=(1,1)) #unigram
tfidf_train = tfidf_vectorizer.fit_transform(x_train)
tfidf_test = tfidf_vectorizer.transform(x_test)

print('TF-IDF Train:', tfidf_train.shape)
print('TF-IDF Test:', tfidf_test.shape)

print(x_train[1])
print(tfidf_train[1])
```

```
TF-IDF Train: (5780, 7217)
TF-IDF Test: (3515, 7217)
course purchase sign process get far sign course process satisfactory nothing beyond
contact often delayed despite email query something post purchase yet response
  (0, 7194)     0.1832667505565071
  (0, 4853)     0.23491525434984517
  (0, 5935)     0.19316792338588662
  (0, 5115)     0.20123324530795014
  (0, 2093)     0.13116701982766407
  (0, 1736)     0.2161093938155354
  (0, 1669)     0.2725854013730652
  (0, 4407)     0.22693043803385754
  (0, 1372)     0.14700265822439096
  (0, 699)      0.2531988592368892
  (0, 4325)     0.16764606891511744
  (0, 5594)     0.3020880336250264
  (0, 2409)     0.1631615775480476
  (0, 4976)     0.30133447997994245
  (0, 5076)     0.41357150178967894
  (0, 5423)     0.16037550231704278
  (0, 1471)     0.14575224240832105
  (0, 2707)     0.1027404834059262
  (0, 5797)     0.3043750389991771
```

2. Baseline Model: Logistic Regression

Back to top

In [15]:

```
# Bag of Words
# Define the model
log_reg_classifier = LogisticRegression()
log_reg_bow = log_reg_classifier.fit(cv_train, y_train)

# Predict
y_bow_predict = log_reg_bow.predict(cv_test)

# Get mean accuracy score
log_reg_bow_score = accuracy_score(y_test, y_bow_predict)
print('Logistic Regression BoW score = ', log_reg_bow_score)

# Print classification report
report = classification_report(y_test, y_bow_predict, target_names=['0','1'])
print('\nBag of Words Classification Report:')
print(report)
```

```
Logistic Regression BoW score =  0.9342816500711237

Bag of Words Classification Report:
              precision    recall  f1-score   support

           0       0.61      0.86      0.71       333
           1       0.98      0.94      0.96      3182

    accuracy                           0.93      3515
   macro avg       0.80      0.90      0.84      3515
```

```
weighted avg       0.95      0.93      0.94      3515
```

For imbalanced dataset, using precision and recall as the metrics are better than using the false positive rate because of the large number of true negative, making the fpr to be small.

In [16]:
```
df_lr = pd.DataFrame()
df_lr['x_test'] = x_test
df_lr['y_test'] = y_test
df_lr['y_predict'] = y_bow_predict

tp = len(df_lr[(df_lr['y_test'] == 0) & (df_lr['y_test'] == df_lr['y_predict'])])
fn = len(df_lr[(df_lr['y_test'] == 0) & (df_lr['y_test'] != df_lr['y_predict'])])
fp = len(df_lr[(df_lr['y_test'] == 1) & (df_lr['y_test'] != df_lr['y_predict'])])

print(tp, fn, fp)
```

```
287 46 185
```

In [ ]:
```
#result = pd.DataFrame()
#result['Feedback'] = x_test
#result['Label Sentiment'] = y_test
#result['Predicted Sentiment'] = y_bow_predict
#
#ratings = []
#for feedback in x_test:
#    ratings.append(df.loc[df['lemmatized_feedback'] == feedback]['reviewRating_rati
#
#result['Rating'] = ratings
```

Classification report:

- The recall means the ability to find all positive samples: tp / (tp + fn)
- The precision will be "how many are correctly classified among that class": tp / (tp + fp)
- The f1-score is the harmonic mean between precision & recall
- The support is the number of occurence of the given class in your dataset
- Precision and recall is highly used for imbalanced dataset because in an highly imbalanced dataset, a 99% accuracy can be meaningless.

In [17]:
```
# TD-IDF
# Define the model
log_reg_classifier = LogisticRegression()
log_reg_tfidf = log_reg_classifier.fit(tfidf_train, y_train)

# Predict
y_tfidf_predict = log_reg_tfidf.predict(tfidf_test)

# Get mean accuracy score
log_reg_tfidf_score = accuracy_score(y_test, y_tfidf_predict)
print('Logistic Regression TF-IDF score = ', log_reg_tfidf_score)

# Print classification report
report = classification_report(y_test, y_tfidf_predict, target_names=['0','1'])
print('\nTF-IDF Classification Report:')
print(report)
```

```
Logistic Regression TF-IDF score =  0.9314366998577525

TF-IDF Classification Report:
```

```
              precision    recall  f1-score   support

           0       0.59      0.90      0.71       333
           1       0.99      0.93      0.96      3182

    accuracy                           0.93      3515
   macro avg       0.79      0.92      0.84      3515
weighted avg       0.95      0.93      0.94      3515
```

In [18]:
```
df_lr = pd.DataFrame()
df_lr['x_test'] = x_test
df_lr['y_test'] = y_test
df_lr['y_predict'] = y_tfidf_predict

tp = len(df_lr[(df_lr['y_test'] == 0) & (df_lr['y_test'] == df_lr['y_predict'])])
fn = len(df_lr[(df_lr['y_test'] == 0) & (df_lr['y_test'] != df_lr['y_predict'])])
fp = len(df_lr[(df_lr['y_test'] == 1) & (df_lr['y_test'] != df_lr['y_predict'])])

print(tp, fn, fp)
```

```
300 33 208
```

Both models perform similarly. For class 0, the low precision means that there are many false positives. But the very high recall is a good thing as the model is able to gather more data. The result is also weird in a sense that the precision is much lower than the model trained using imbalanced dataset. However, after checking the prediction manually, it turns out that this model can predict better, hence more false positives and lower precision.

3. Baseline Model: Neural Network

Back to top

Will use the bag of words representation of training set.

In [19]:
```
# define the model: 1 layer neural network
reset_seed()
clear_session()
input_dim = cv_train.shape[1]

model = Sequential()
model.add(layers.Dense(10, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy

model.summary()
```

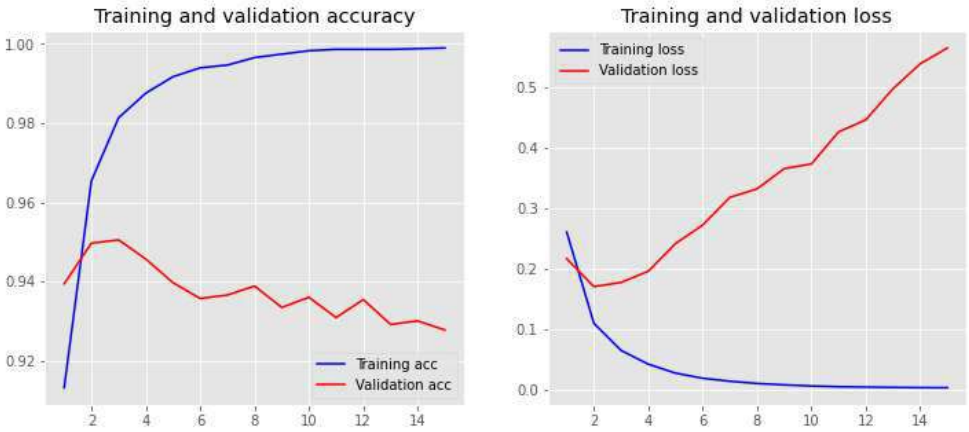
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                72180
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 72,191
Trainable params: 72,191
Non-trainable params: 0
_________________________________________________________________
```

In [20]:
```
# train the model
training_log = model.fit(cv_train, y_train,
                         epochs=15,
                         verbose=False,
```

```
                          validation_data=(cv_test, y_test),
                          batch_size=5
                         )
```

In [21]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(cv_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(cv_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9990
Testing Accuracy:  0.9277
```



The model is overfitting as can be seen on increasing validation loss. The accuracy plot also shows an overfitting model. To reduce the overfitting, an early stopping mechanism is introduced.

In [22]:
```
# With early stopping
# define the model: 1 layer neural network
reset_seed()
clear_session()
input_dim = cv_train.shape[1]

model = Sequential()
model.add(layers.Dense(10, input_dim=input_dim, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy

# train the model
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(cv_train, y_train,
                         epochs=15,
                         verbose=False,
                         validation_data=(cv_test, y_test),
                         batch_size=5,
                         callbacks=[callback]
                        )
```

In [23]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(cv_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(cv_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

```
plot_training_log(training_log)
```

```
Training Accuracy: 0.9825
Testing Accuracy:  0.9496
```

In [24]:
```
# Predict
y_nn_predict = model.predict(cv_test)
y_nn_predict = [1 if x >= 0.5 else 0 for x in y_nn_predict]

# Print classification report
report = classification_report(y_test, y_nn_predict, target_names=['0','1'])
print('Neural Network Classification Report:')
print(report)
```

```
Neural Network Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.87      0.77       333
           1       0.99      0.96      0.97      3182

    accuracy                           0.95      3515
   macro avg       0.84      0.91      0.87      3515
weighted avg       0.96      0.95      0.95      3515
```

With early stopping, the model performs better than the logistic regression with better precision and recall for class 0.

4. Data Representation: Word Embedding

Back to top

4.1. Hardcoded Indexed Word Embedding

In [22]:
```
# Hardcoded indexed word embedding
tokenizer = Tokenizer(num_words=5000) # take most common num_words words as vocabula
tokenizer.fit_on_texts(x_train)

indexed_word_embedding_train = tokenizer.texts_to_sequences(x_train)
indexed_word_embedding_test = tokenizer.texts_to_sequences(x_test)

vocab_size = len(tokenizer.word_index) + 1 # plus 1 because index 0 is a reserved in

print(vocab_size)
print(x_train[1])
print(indexed_word_embedding_train[1])
```

```
7238
course purchase sign process get far sign course process satisfactory nothing beyond
contact often delayed despite email query something post purchase yet response
[1, 239, 67, 69, 2, 92, 67, 1, 69, 1703, 113, 832, 45, 468, 1109, 398, 12, 292, 243,
506, 239, 180, 81]
```

In [23]:
```
# padding with zeroes to make the embedding length the same
maxlen = 100 # remove long sentences
indexed_word_embedding_train = pad_sequences(indexed_word_embedding_train, padding='
indexed_word_embedding_test = pad_sequences(indexed_word_embedding_test, padding='po

print(indexed_word_embedding_train[1, :])
```

```
[   1  239   67   69    2   92   67    1   69 1703  113  832   45  468
 1109  398   12  292  243  506  239  180   81    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0]
```

In [29]:
```
# train with embedding as part of the model structure using neural network architect
reset_seed()
clear_session()
embedding_dim = 50

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           input_length=maxlen))
model.add(layers.Flatten()) # converting matrix to single array (remove all dimensio
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

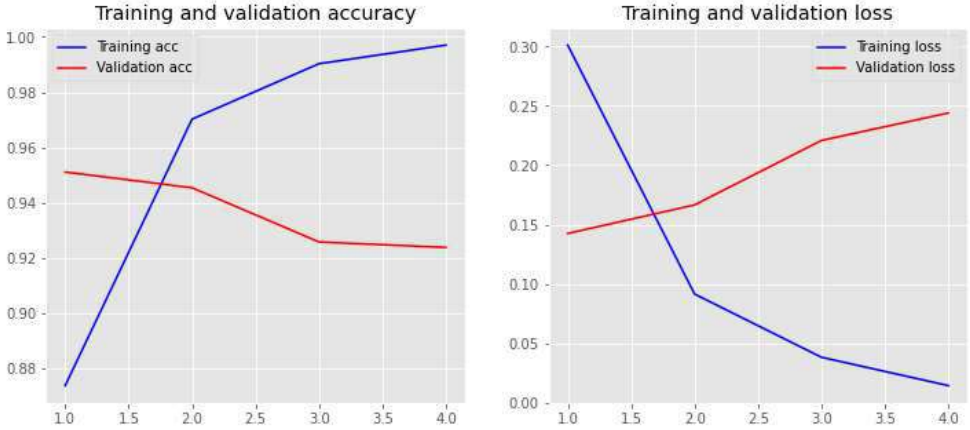
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
flatten_1 (Flatten)          (None, 5000)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                50010
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 411,921
Trainable params: 411,921
Non-trainable params: 0
_________________________________________________________________
```

In [30]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=15,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback]
                         )
```

In [31]:
```
# Plot the learning curve
```

```
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9625
Testing Accuracy:  0.9511
```



In [32]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('Indexed Word Embedding Without Pooling Classification Report:')
print(report)
```

```
Indexed Word Embedding Without Pooling Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.81      0.76       333
           1       0.98      0.97      0.97      3182

    accuracy                           0.95      3515
   macro avg       0.85      0.89      0.87      3515
weighted avg       0.95      0.95      0.95      3515
```

Using the hardcoded indexes word embedding with only positional information, the model has a good balance of precision and recall. As text data are sequential, it is better to have a model that is able to learn the sequence information rather than only learn each word position. To do this, a pooling layer is used, which will downsize the number of features. First, global max pooling layer is added, which will take the highest valued feature across all features.

In [33]:
```
# use pooling layer to improve feature learning
reset_seed()
clear_session()
embedding_dim = 50

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           input_length=maxlen))
model.add(layers.GlobalMaxPool1D())
model.add(layers.Dense(10, activation='relu'))
```

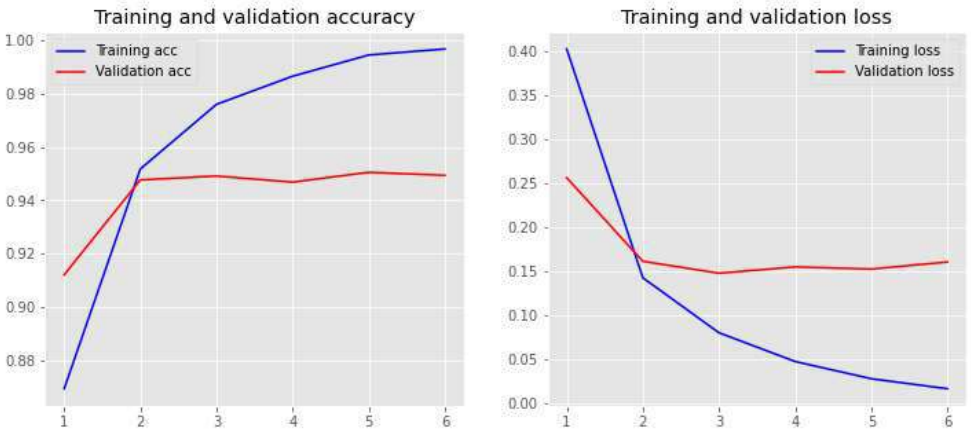
```
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                510
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 362,421
Trainable params: 362,421
Non-trainable params: 0
_________________________________________________________________
```

In [34]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=15,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback]
                         )
```

In [35]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9882
Testing Accuracy:  0.9491
```



In [36]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
```

```
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('Indexed Word Embedding With Global Max Pooling Classification Report:')
print(report)
```

```
Indexed Word Embedding With Global Max Pooling Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.90      0.77       333
           1       0.99      0.95      0.97      3182

    accuracy                           0.95      3515
   macro avg       0.83      0.93      0.87      3515
weighted avg       0.96      0.95      0.95      3515
```

The model shows an improvement (higher recall on class 0) compared to the model without pooling layer. Next is the max pooling layer, which will take the highest valued feature for every pool sized features.

In [37]:
```
# use pooling layer to improve feature learning
reset_seed()
clear_session()
embedding_dim = 50

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           input_length=maxlen))
model.add(layers.MaxPooling1D(pool_size=2))
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

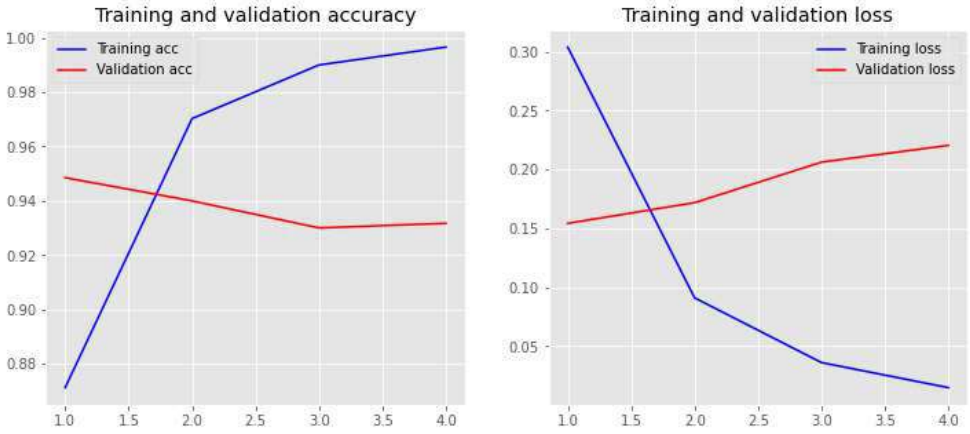
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 50, 50)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2500)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                25010
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 386,921
Trainable params: 386,921
Non-trainable params: 0
_________________________________________________________________
```

In [38]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=15,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback]
                         )
```

In [39]:
```
# Plot the learning curve
```

```
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9644
Testing Accuracy:  0.9485
```



In [40]:

```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('Indexed Word Embedding With Max Pooling Classification Report:')
print(report)
```

```
Indexed Word Embedding With Max Pooling Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.84      0.76       333
           1       0.98      0.96      0.97      3182

    accuracy                           0.95      3515
   macro avg       0.83      0.90      0.86      3515
weighted avg       0.95      0.95      0.95      3515
```

This model is very similar to the model with global max pooling and better than the model without pooling layer.

4.2. Pretrained Word Embedding: GloVe (without training)

Back to top

In [41]:

```
# load the embedding file, and assign the embedding vectors only to words available
def create_glove_embedding_matrix(filepath, word_index, embedding_dim):
    vocab_size = len(word_index) + 1  # plus 1 because index 0 is a reserved index
    embedding_matrix = np.zeros((vocab_size, embedding_dim))

    with open(filepath, encoding="utf8") as f:
        for line in f:
            word, *vector = line.split()
            if word in word_index:
                idx = word_index[word]
```

```
                embedding_matrix[idx] = np.array(vector, dtype=np.float32)[:embeddin

    return embedding_matrix
```

In [42]:
```
# Create the embedding matrix based on the current vocabulary in tokenizer.word_inde
embedding_dim = 50

embedding_matrix = create_glove_embedding_matrix(
    'Dataset/glove.6B.50d.txt',
    tokenizer.word_index, embedding_dim)
```

In [43]:
```
# Check the ratio of the vocabulary available in the embedding file
nonzero_elements = np.count_nonzero(np.count_nonzero(embedding_matrix, axis=1))
nonzero_elements / vocab_size
```

Out[43]:
```
0.9049461177120751
```

This means the GloVe embedding model covers 90% of the vocabulary in the dataset.

In [44]:
```
reset_seed()
clear_session()
embedding_dim = 50
maxlen = 100

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           weights=[embedding_matrix], # each word has the embedding
                           input_length=maxlen,
                           trainable=False)) # the dataset is not used as a part of
model.add(layers.GlobalMaxPool1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

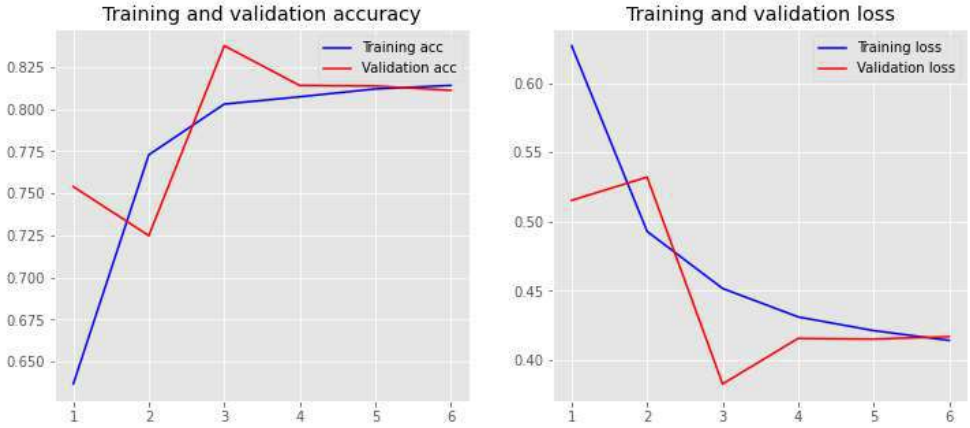
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                510
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 362,421
Trainable params: 521
Non-trainable params: 361,900
_________________________________________________________________
```

In [45]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=100,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback])
```

In [46]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
```

```
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.8102
Testing Accuracy:  0.8376
```



In [47]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('GloVe Word Embedding Without Training Classification Report:')
print(report)
```

```
GloVe Word Embedding Without Training Classification Report:
              precision    recall  f1-score   support

           0       0.34      0.76      0.47       333
           1       0.97      0.85      0.90      3182

    accuracy                           0.84      3515
   macro avg       0.66      0.80      0.69      3515
weighted avg       0.91      0.84      0.86      3515
```

Very poor performance in terms of spotting all class 0. This might be expected as the embedding model is not trained with the dataset vocabulary. The learning curves show that the validation accuracy is higher and validation loss is lower. This might be because of the unintentional 10% dropout on the vocabulary as the GloVe embedding model only recognises 90% of the dataset vocabulary. In this case, the training set becomes more difficult for the model to learn (thus, lower accuracy), but it is easier for the validation set as all vocabularies are present (as the 10% vocabularies now have the weights of the embedding from backpropagation).

4.3. Pretrained Word Embedding: GloVe (with training)

Back to top

In [50]:
```
reset_seed()
clear_session()
embedding_dim = 50
```

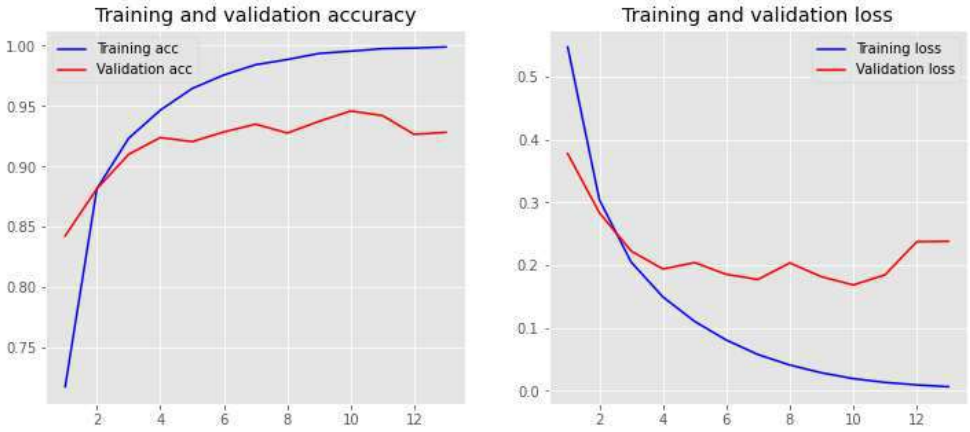
```
model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           weights=[embedding_matrix],
                           input_length=maxlen,
                           trainable=True))
model.add(layers.GlobalMaxPool1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                510
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 362,421
Trainable params: 362,421
Non-trainable params: 0
_________________________________________________________________
```

In [51]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=100,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback])
```

In [52]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9969
Testing Accuracy:  0.9459
```



In [53]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('GloVe Word Embedding With Training Classification Report:')
print(report)
```

```
GloVe Word Embedding With Training Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.81      0.74       333
           1       0.98      0.96      0.97      3182

    accuracy                           0.95      3515
   macro avg       0.83      0.88      0.85      3515
weighted avg       0.95      0.95      0.95      3515
```

Indeed, with the dataset being trained together within the embedding model, the performance improves alot. The model is on par with the baseline models. The learning curves show more 'normal' behaviour because although there are 10% vocabularies dropout, they are trained together in the embedding layer.

4.4. Pretrained Word Embedding: Word2Vec (without training)

Back to top

In [ ]:
```
# ONLY NEED TO RUN ONCE: convert the binary file to text file
# Load the word2vec pretrained model
model = KeyedVectors.load_word2vec_format('Dataset/GoogleNews-vectors-negative300.bi

# save model in ASCII (word2vec) format
model.wv.save_word2vec_format('Dataset/embedding_word2vec.txt', binary=False)
```

In [56]:
```
# load the embedding file, and assign the embedding vectors only to words available
def create_word2vec_embedding_matrix(filepath, word_index, embedding_dim):
    vocab_size = len(word_index) + 1  # plus 1 because index 0 is a reserved index
    embedding_matrix = np.zeros((vocab_size, embedding_dim))

    with open(filepath, encoding="utf8") as f:
        for line in f:
            word, *vector = line.split()
            if word in word_index:
```

```
                idx = word_index[word]
                embedding_matrix[idx] = np.array(vector, dtype=np.float32)[:embeddin

    return embedding_matrix
```

In [57]:
```
# Create the embedding matrix based on the current vocabulary in tokenizer.word_inde
embedding_dim = 50

embedding_matrix = create_word2vec_embedding_matrix(
    'Dataset/embedding_word2vec.txt',
    tokenizer.word_index, embedding_dim)
```

In [58]:
```
# Check the ratio of the vocabulary available in the embedding file
nonzero_elements = np.count_nonzero(np.count_nonzero(embedding_matrix, axis=1))
nonzero_elements / vocab_size
```

Out[58]:
```
0.8912683061619232
```

In [59]:
```
reset_seed()
clear_session()
embedding_dim = 50
maxlen = 100

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           weights=[embedding_matrix], # each word has the embedding
                           input_length=maxlen,
                           trainable=False)) # the dataset is not used as a part of
model.add(layers.GlobalMaxPool1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

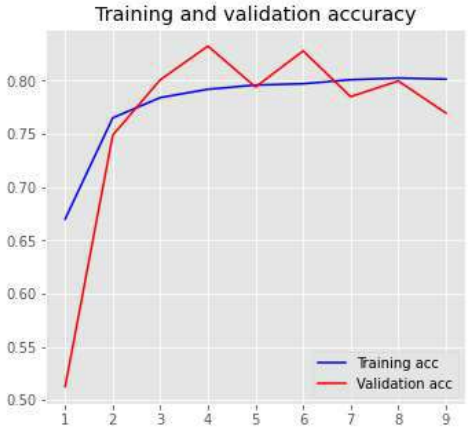
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                510
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 362,421
Trainable params: 521
Non-trainable params: 361,900
_________________________________________________________________
```

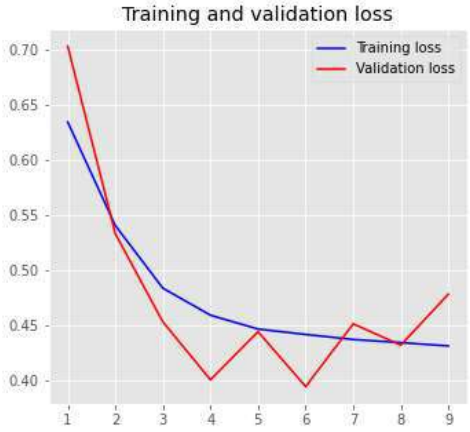
In [60]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=100,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback])
```

In [61]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
```

```
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.8054
Testing Accuracy:  0.8279
```





In [62]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('Word2Vec Without Training Classification Report:')
print(report)
```

```
Word2Vec Without Training Classification Report:
              precision    recall  f1-score   support

           0       0.32      0.72      0.44       333
           1       0.97      0.84      0.90      3182

    accuracy                           0.83      3515
   macro avg       0.64      0.78      0.67      3515
weighted avg       0.91      0.83      0.86      3515
```

Similar to the GloVe model without training, the model performs poorly and the learning curves show 'abnormal' behaviour that might be caused by the 11% vocabularies dropout in the word2vec model.

4.5. Pretrained Word Embedding: Word2Vec (with training)

Back to top

In [63]:
```
reset_seed()
clear_session()
embedding_dim = 50
maxlen = 100

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           weights=[embedding_matrix], # each word has the embedding
```

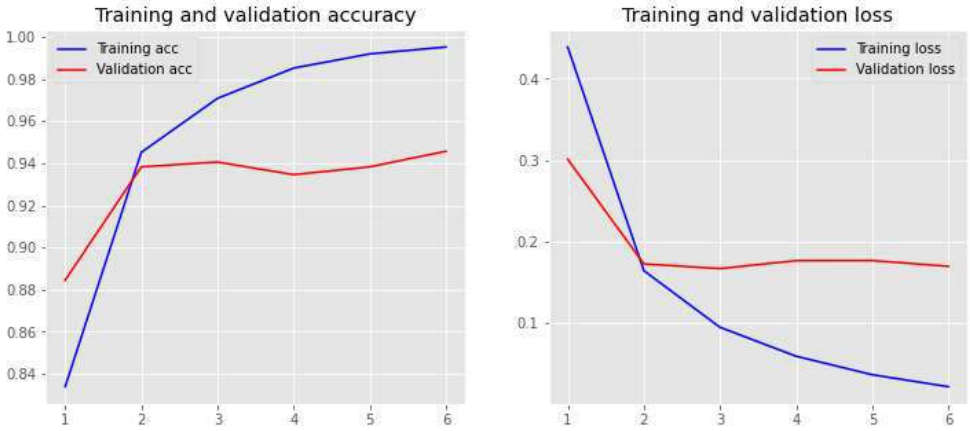
```
                           input_length=maxlen,
                           trainable=True)) # the dataset is not used as a part of e
model.add(layers.GlobalMaxPool1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                510
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 362,421
Trainable params: 362,421
Non-trainable params: 0
_________________________________________________________________
```

In [64]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=100,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback])
```

In [65]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9855
Testing Accuracy:  0.9405
```



In [66]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
```

```
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('Word2Vec With Training Classification Report:')
print(report)
```

```
Word2Vec With Training Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.90      0.74       333
           1       0.99      0.95      0.97      3182

    accuracy                           0.94      3515
   macro avg       0.81      0.92      0.85      3515
weighted avg       0.95      0.94      0.95      3515
```

The model works well and comparable with the GloVe embedding model and other baseline models. The model can detect more class 0 although with a slightly lower precision.

5. Convolutional Neural Network

Back to top

5.1. CNN without pretrained embedding

In [67]:
```
reset_seed()
clear_session()
embedding_dim = 50
maxlen = 100

model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Conv1D(128, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

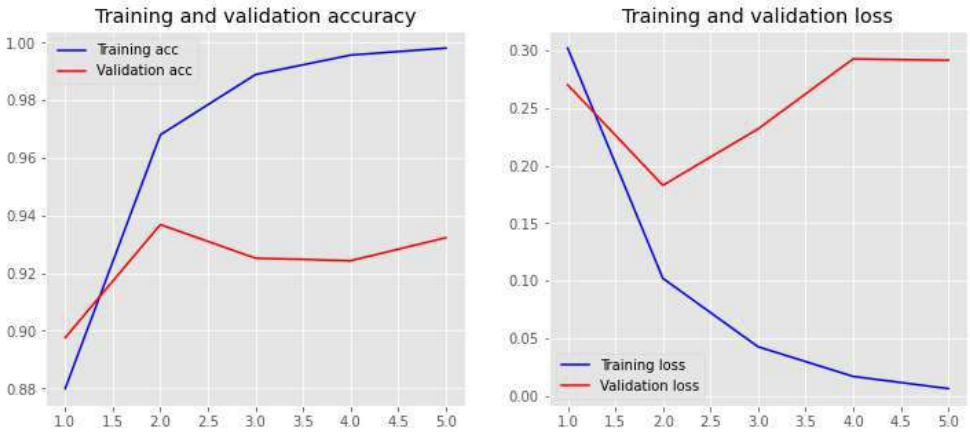
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 96, 128)           32128
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 395,329
Trainable params: 395,329
Non-trainable params: 0
_________________________________________________________________
```

In [68]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=50,
```

```
                        verbose=False,
                        validation_data=(indexed_word_embedding_test, y_test),
                        batch_size=10,
                        callbacks=[callback])
```

In [69]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9886
Testing Accuracy:  0.9368
```



In [70]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('CNN Classification Report:')
print(report)
```

```
CNN Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.88      0.72       333
           1       0.99      0.94      0.96      3182

    accuracy                           0.94      3515
   macro avg       0.80      0.91      0.84      3515
weighted avg       0.95      0.94      0.94      3515
```

A decent model that is comparable with the logistic regression models.

5.2. CNN with GloVe embedding

Back to top

In [16]:
```
# load the embedding file, and assign the embedding vectors only to words available
def create_glove_embedding_matrix(filepath, word_index, embedding_dim):
    vocab_size = len(word_index) + 1  # plus 1 because index 0 is a reserved index
    embedding_matrix = np.zeros((vocab_size, embedding_dim))
```

```
    with open(filepath, encoding="utf8") as f:
        for line in f:
            word, *vector = line.split()
            if word in word_index:
                idx = word_index[word]
                embedding_matrix[idx] = np.array(vector, dtype=np.float32)[:embeddin

    return embedding_matrix
```

In [72]:
```
# Create the embedding matrix based on the current vocabulary in tokenizer.word_inde
embedding_dim = 50

embedding_matrix = create_glove_embedding_matrix(
    'Dataset/glove.6B.50d.txt',
    tokenizer.word_index, embedding_dim)
```

In [73]:
```
reset_seed()
clear_session()
embedding_dim = 50
maxlen = 100

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           weights=[embedding_matrix], # each word has the embedding
                           input_length=maxlen,
                           trainable=True)) # the dataset is not used as a part of e
model.add(layers.Conv1D(128, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 96, 128)           32128
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 395,329
Trainable params: 395,329
Non-trainable params: 0
_________________________________________________________________
```

In [74]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=50,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback])
```

In [75]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
```

```
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9979
Testing Accuracy:  0.9425
```

In [76]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('CNN with GloVe Word Embedding Classification Report:')
print(report)
```

```
CNN with GloVe Word Embedding Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.85      0.74       333
           1       0.98      0.95      0.97      3182

    accuracy                           0.94      3515
   macro avg       0.82      0.90      0.85      3515
weighted avg       0.95      0.94      0.95      3515
```

With GloVe embedding, it is better than the model without pretrained embedding, and performs as well as other baseline models. Compared with the neural network with GloVe embedding, this model has a higher precision with a comparatively still good recall, which is an advantage as the model recognises more class 0 data.

5.3. CNN with Word2Vec embedding

Back to top

In [77]:
```
# load the embedding file, and assign the embedding vectors only to words available
def create_word2vec_embedding_matrix(filepath, word_index, embedding_dim):
    vocab_size = len(word_index) + 1  # plus 1 because index 0 is a reserved index
    embedding_matrix = np.zeros((vocab_size, embedding_dim))

    with open(filepath, encoding="utf8") as f:
        for line in f:
            word, *vector = line.split()
```

```
            if word in word_index:
                idx = word_index[word]
                embedding_matrix[idx] = np.array(vector, dtype=np.float32)[:embeddin

    return embedding_matrix
```

In [78]:
```
# Create the embedding matrix based on the current vocabulary in tokenizer.word_inde
embedding_dim = 50

embedding_matrix = create_word2vec_embedding_matrix(
    'Dataset/embedding_word2vec.txt',
    tokenizer.word_index, embedding_dim)
```

In [79]:
```
reset_seed()
clear_session()
embedding_dim = 50
maxlen = 100

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           weights=[embedding_matrix], # each word has the embedding
                           input_length=maxlen,
                           trainable=True)) # the dataset is not used as a part of e
model.add(layers.Conv1D(128, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

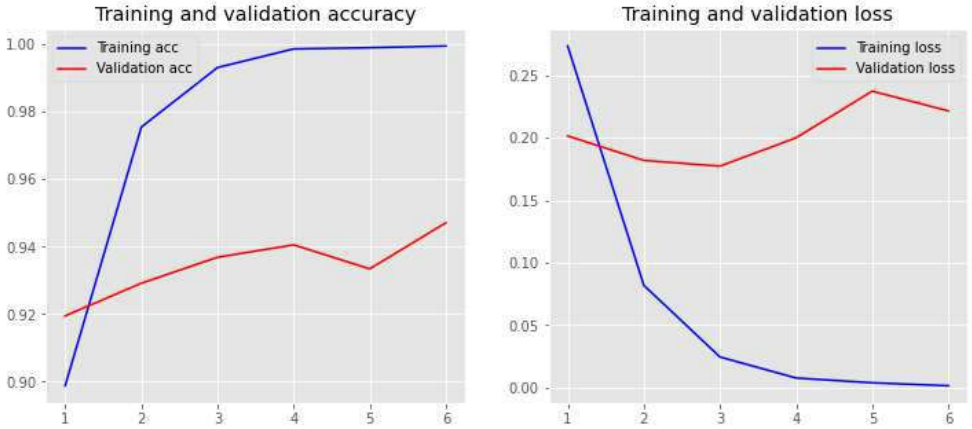
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 96, 128)           32128
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 395,329
Trainable params: 395,329
Non-trainable params: 0
_________________________________________________________________
```

In [80]:
```
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_bes
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=50,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback])
```

In [81]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
```

```
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9988
Testing Accuracy:  0.9368
```



In [82]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('CNN with Word2Vec Classification Report:')
print(report)
```

```
CNN with Word2Vec Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.89      0.73       333
           1       0.99      0.94      0.96      3182

    accuracy                           0.94      3515
   macro avg       0.80      0.91      0.85      3515
weighted avg       0.95      0.94      0.94      3515
```

5.4. Hyperparameter Tuning

Back to top

Based on the experiment, the CNN with GloVe embedding model is quite good to be used as the production model. Then, parameter tuning the model to find the best combination of parameters and see if it can improve the performance.

In [38]:
```
# Define a function that return a Keras model
def create_model(num_filters, kernel_size, embedding_dim, maxlen):
    # Hardcoded indexed word embedding
    tokenizer = Tokenizer(num_words=5000) # take most common num_words words as voca
    tokenizer.fit_on_texts(x_train)

    word_index = tokenizer.word_index

    # Create the embedding matrix based on the current vocabulary in tokenizer.word_
    vocab_size = len(word_index) + 1

    embedding_matrix = create_glove_embedding_matrix(
```

```
        'Dataset/glove.6B.50d.txt',
        tokenizer.word_index, embedding_dim)

    model = Sequential()
    model.add(layers.Embedding(input_dim=vocab_size, # Size of the vocabulary, i.e.
                               output_dim=embedding_dim, # Dimension of the dense embedd
                               weights=[embedding_matrix], # each word has the initial e
                               input_length=maxlen, # Length of input sequences, when it
                               trainable=True)) # the dataset is not used as a part of e
    model.add(layers.Conv1D(num_filters,
                            kernel_size,
                            activation='relu'))
    model.add(layers.GlobalMaxPooling1D())
    model.add(layers.Dense(10, activation='relu'))
    model.add(layers.Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model
```

In [39]:
```
# setting the parameter dictionary
param_grid = dict(num_filters=[32, 64, 128],
                  kernel_size=[3, 5, 7],
                  embedding_dim=[50],
                  maxlen=[100])
```

In [40]:
```
# Initiate the model, run hyperparameter tuning
reset_seed()
model = KerasClassifier(build_fn=create_model, #construct, compile and return a mode
                        epochs=20,
                        batch_size=10,
                        verbose=False
                       )

search_res = RandomizedSearchCV(estimator=model,
                                param_distributions=param_grid,
                                cv=5, # cross validate the training set to cover mor
                                verbose=1,
                                n_iter=5, # number of parameter settings that are sa
                                random_state=1
                               )

callback = keras.callbacks.EarlyStopping(monitor='loss', patience=3, restore_best_we
model_result = search_res.fit(indexed_word_embedding_train, y_train, callbacks=[call
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
```

In [41]:
```
# Evaluate testing set
test_accuracy = search_res.score(indexed_word_embedding_test, y_test)

output_file = 'output.txt'
with open(output_file, 'a') as f:
    s = ('Best Accuracy : '
         '{:.4f} \n{} \nTest Accuracy : {:.4f}')
    output_string = s.format(
        model_result.best_score_,
        model_result.best_params_,
        test_accuracy)
    print(output_string)
    f.write(output_string)
```

```
Best Accuracy : 0.9488
{'num_filters': 64, 'maxlen': 100, 'kernel_size': 3, 'embedding_dim': 50}
Test Accuracy : 0.9440
```

Now that the best combination of parameters has been found, train a new model with this set of parameters.

In [42]:
```
#tokenizer = Tokenizer(num_words=5000) # take most common num_words words as vocabul
#tokenizer.fit_on_texts(x_train)
#
#word_index = tokenizer.word_index
#vocab_size = len(word_index) + 1

reset_seed()
clear_session()
embedding_dim = 50
maxlen = 100

embedding_matrix = create_glove_embedding_matrix(
            'Dataset/glove.6B.50d.txt',
            tokenizer.word_index, embedding_dim)

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           weights=[embedding_matrix], # each word has the embedding'
                           input_length=maxlen,
                           trainable=True)) # the dataset is not used as a part of em
model.add(layers.Conv1D(64, 3, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

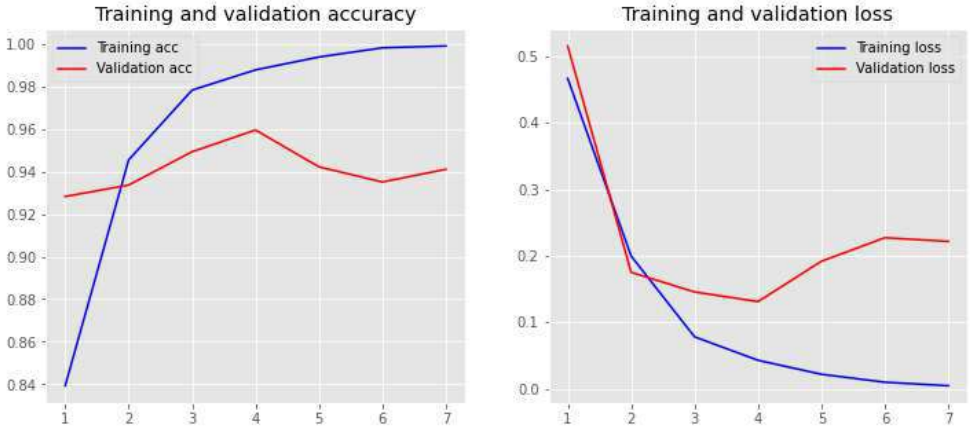
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 50)           361900
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 98, 64)            9664
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 11
=================================================================
Total params: 372,225
Trainable params: 372,225
Non-trainable params: 0
_________________________________________________________________
```

In [43]:
```
callback = callbacks.EarlyStopping(monitor='val_loss', patience=3, restore_best_weig
training_log = model.fit(indexed_word_embedding_train, y_train,
                         epochs=50,
                         verbose=False,
                         validation_data=(indexed_word_embedding_test, y_test),
                         batch_size=10,
                         callbacks=[callback])
```

In [44]:
```
# Plot the learning curve
loss, accuracy = model.evaluate(indexed_word_embedding_train, y_train, verbose=False
print("Training Accuracy: {:.4f}".format(accuracy))

loss, accuracy = model.evaluate(indexed_word_embedding_test, y_test, verbose=False)
```

```
print("Testing Accuracy:  {:.4f}".format(accuracy))

plot_training_log(training_log)
```

```
Training Accuracy: 0.9905
Testing Accuracy:  0.9596
```



In [45]:
```
# Predict
y_indexed_word_embedding_predict = model.predict(indexed_word_embedding_test)
y_indexed_word_embedding_predict = [1 if x >= 0.5 else 0 for x in y_indexed_word_emb

# Print classification report
report = classification_report(y_test, y_indexed_word_embedding_predict, target_name
print('Best Parameters CNN With GloVe Embedding Classification Report:')
print(report)
```

```
Best Parameters CNN With GloVe Embedding Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.82      0.79       333
           1       0.98      0.97      0.98      3182

    accuracy                           0.96      3515
   macro avg       0.88      0.90      0.89      3515
weighted avg       0.96      0.96      0.96      3515
```

In [46]:
```
df_lr = pd.DataFrame()
df_lr['x_test'] = x_test
df_lr['y_test'] = y_test
df_lr['y_predict'] = y_indexed_word_embedding_predict

tp = len(df_lr[(df_lr['y_test'] == 0) & (df_lr['y_test'] == df_lr['y_predict'])])
fn = len(df_lr[(df_lr['y_test'] == 0) & (df_lr['y_test'] != df_lr['y_predict'])])
fp = len(df_lr[(df_lr['y_test'] == 1) & (df_lr['y_test'] != df_lr['y_predict'])])
print(tp, fn, fp)
```

```
272 61 81
```

In [47]:
```
model.save('Model/CNN_glove_unbiased.h5')
```

Make all predictions:

In [49]:
```
df_all = pd.read_csv('Dataset/trustpilot_reviews.csv')
```

In [51]:
```
# Hardcoded indexed word embedding
tokenizer = Tokenizer(num_words=5000) # take most common num_words words as vocabula
tokenizer.fit_on_texts(x_train)
```

```
indexed_word_embedding_all = tokenizer.texts_to_sequences(df['lemmatized_feedback'])

vocab_size = len(tokenizer.word_index) + 1 # plus 1 because index 0 is a reserved in

print(vocab_size)
print(x_train[1])
print(indexed_word_embedding_all[1])
```

```
7238
course purchase sign process get far sign course process satisfactory nothing beyond
contact often delayed despite email query something post purchase yet response
[101, 50, 339, 66, 350, 36, 101, 50, 339, 265, 390, 211, 5, 3, 172, 34, 1616, 212, 3
6, 98, 4221, 3010, 160, 15, 382, 5, 463, 7, 234, 98, 108, 55, 146, 19, 36, 558, 112
1]
```

In [53]:
```
# padding with zeroes to make the embedding length the same
maxlen = 100 # remove long sentences
indexed_word_embedding_all = pad_sequences(indexed_word_embedding_all, padding='post

print(indexed_word_embedding_all[1, :])
```

```
[ 101   50  339   66  350   36  101   50  339  265  390  211    5    3
  172   34 1616  212   36   98 4221 3010  160   15  382    5  463    7
  234   98  108   55  146   19   36  558 1121    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0]
```

In [54]:
```
prediction = model.predict(indexed_word_embedding_all)
prediction = [1 if x >= 0.5 else 0 for x in prediction]
```

In [55]:
```
df_all['predicted'] = prediction
```

In [57]:
```
df_all.to_csv('Dataset/trustpilot_reviews.csv', index=False)
```

Appendix F

Feedback Summarisation Script

The feedback summarisation script here is the shortened version. It includes all the codes, but with only limited results shown due to the long pages.

Feedback Summarisation

In [1]:
```
import pandas as pd
import numpy as np
from nltk.corpus import stopwords, wordnet
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk import pos_tag
from nltk.stem import WordNetLemmatizer
from rouge import Rouge
import re
from summarizer import Summarizer
stopwords = stopwords.words('english')
```

In [2]:
```
df = pd.read_csv('Dataset/trustpilot_reviews.csv')
```

In [3]:
```
df.head(2)
```

Out[3]:

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head_wo
0	2021-01-15T13:50:34+00:00	When speaking to one advisors on...	When speaking to one of the advisors on the ph...	Avado	3	
1	2021-01-15T10:45:08+00:00	CIPD Level 5 Diploma	I enrolled today to study my CIPD Level 5 Dipl...	Avado	5	

In [4]:
```
# prepare the data by splitting into good and bad reviews, then combining the
df_good = df[df['overall_rating']==1]
df_bad = df[df['overall_rating']==0]
```

In [5]:
```
# prepare the data by splitting into each company
companies = list(set(df['itemReviewed_name']))
df_companies = {'df_' + str(company):df[df['itemReviewed_name']==company] for compan
```

In [6]:
```
df_companies['df_Avado'].head(2)
```

Out[6]:

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head_wo
0	2021-01-15T13:50:34+00:00	When speaking to one advisors on...	When speaking to one of the advisors on the ph...	Avado	3	
1	2021-01-15T10:45:08+00:00	CIPD Level 5 Diploma	I enrolled today to study my CIPD Level 5 Dipl...	Avado	5	

In [7]:
```
# prepare the data by splitting into each company and further by good and bad review
for company in companies:
```

```
    name = 'df_{0}'.format(company)
    df_companies['{0}_good'.format(name)] = df_companies[name][df_companies[name]['o
    df_companies['{0}_bad'.format(name)] = df_companies[name][df_companies[name]['ov
```

In [8]:
```
df_companies['df_Avado_good'].head(2)
```

Out[8]:

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head_wo
1	2021-01-15T10:45:08+00:00	CIPD Level 5 Diploma	I enrolled today to study my CIPD Level 5 Dipl...	Avado	5	
2	2021-01-12T23:13:19+00:00	Great!	I'm enjoying the course that I'm doing. The st...	Avado	5	

In [9]:
```
df_companies['df_Avado_bad'].head(2)
```

Out[9]:

	datePublished	headline	reviewBody	itemReviewed_name	reviewRating_ratingValue	head
0	2021-01-15T13:50:34+00:00	When speaking to one advisors on...	When speaking to one of the advisors on the ph...	Avado	3	
3	2021-01-08T20:26:16+00:00	Very disappointed with the service	Very disappointed with the service I have rece...	Avado	2	

In [10]:
```
# Function: lemmatize the word based on the pos tag
def get_simple_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

lemmatizer = WordNetLemmatizer()
def lemmatize_word(word):
    pos = pos_tag([word.strip()])
    return lemmatizer.lemmatize(word.strip(),get_simple_pos(pos[0][1]))

def lemmatize_sentence(text):
    final_text = []
    for i in text.split():
        pos = pos_tag([i.strip()])
        word = lemmatizer.lemmatize(i.strip(),get_simple_pos(pos[0][1]))
        final_text.append(word)
    return " ".join(final_text)
```

1. Weighted Word Occurence Method

Count the occurence for each word in the document, take the highest occurence and divide each word's occurence with the highest occurence to get the weighted word. The score is calculated by summing the occurence scrore for each word in the sentence.

In [11]:
```
def weighted_occurence_summarisation(combined_cleaned_feedback, original_feedback_se
    '''
    combined_cleaned_feedback: a string of all cleaned feedback combined (with or wi
    original_feedback_sentences: an array of original feedback sentences
    n_sentence: n number of sentences to be taken to form the summary
    is_lemmatized: whether to lemmatize words in calculating the word occurrences an
    '''
    # calculate word occurences from the combined feedback text
    word_occurences = {}
    for word in word_tokenize(combined_cleaned_feedback):
        if word not in word_occurences.keys():
            word_occurences[word] = 1
        else:
            word_occurences[word] += 1

    # calculate weighted word occurence
    if len(word_occurences.values()) == 0:
        return 'N/A'
    highest_occurence = max(word_occurences.values())

    for word in word_occurences.keys():
        word_occurences[word] = (word_occurences[word]/highest_occurence)

    # iterates each sentence and give it a score
    sentence_scores = {}
    for sent in original_feedback_sentences:
        for word in word_tokenize(sent.lower()):
            if is_lemmatized:
                word = lemmatize_word(word)
            if word in word_occurences.keys():
                if len(sent.split(' ')) < 30: # to filter out sentences
                    if sent not in sentence_scores.keys():
                        sentence_scores[sent] = word_occurences[word]
                    else:
                        sentence_scores[sent] += word_occurences[word]

    # sort based on the score value
    sorted_sentences = dict(sorted(sentence_scores.items(), key=lambda item: item[1]

    # take n number of sentences
    summary = []
    n = 0
    for key, value in sorted_sentences.items():
        if n == n_sentence: break
        if value <= 5: # filter out too high score to avoid short generic feedback,
            summary.append(key)
            n += 1

    return ' '.join(summary)
```

1.1. Weighted Word from Whole Dataset by Sentiment (no lemmatization)

In [38]:
```
good_feedback_combined = ' '.join(str(body) for body in df_good['reviewBody_clean'].
```

```
reference = ' '.join(str(body).replace('\n','') for body in df_good['reviewBody'].to
sentences = sent_tokenize(reference)
summary = weighted_occurence_summarisation(good_feedback_combined, sentences, 7, Fal

# recall: how much the words in the reference appeared in the summaries
# precision: how much the words in the summary appeared in the reference
# f1 score: 2*(precision*recall)/(precision+recall)

rouge = Rouge()
score_rouge = rouge.get_scores(summary, reference)

print(score_rouge[0]['rouge-1'])
print(summary)
```

```
{'f': 0.0004809778440702294, 'p': 1.0, 'r': 0.00024054677327420764}
Really excited to work on with this course Good Course & Helpful staff.Very easy and
stress-free sign up, all information provided to make easy to set up and login. Real
ly helpful, easy to enrol, changed my course before I started, this was no hassle at
all... Great course, really informative, loads of help. Keep up the good work! so fa
r so good everything about the learn direct is perfect Great experience,great start
up and materials.really happy Lovely course and very helpful customer service. Highl
y recommend. honest and easy to understand process - would recommend Alison was real
ly helpful with booking the course Great learning and experience and online set up!
Thoroughly enjoyed doing my course via Avado The course material was excellent, the
portal looked very professional and I had great tutor support throughout my course.
```

In [39]:

```
bad_feedback_combined = ' '.join(str(body) for body in df_bad['reviewBody_clean'].to
reference = ' '.join(str(body).replace('\n','') for body in df_bad['reviewBody'].to_
sentences = sent_tokenize(reference)
summary = weighted_occurence_summarisation(bad_feedback_combined, sentences, 7, Fals

# recall: how much the words in the reference appeared in the summaries
# precision: how much the words in the summary appeared in the reference
# f1 score: 2*(precision*recall)/(precision+recall)

rouge = Rouge()
score_rouge = rouge.get_scores(summary, reference)

print(score_rouge[0]['rouge-1'])
print(summary)
```

```
{'f': 0.0018506572292130937, 'p': 1.0, 'r': 0.0009261856499441642}
INCORRECT COURSE MATERIAL - The course would often provide guidelines that were need
ed to complete assignments for the course then assessors would fail the assignment f
or using their guidelines.5. The course highlights once you get a free nail kit with
the course after paying at least 50% for the course which is fine by me. Daughter fo
und course information on this company's website to be in-correct and misleading; cu
stomer service advisor also gave in-correct course information. This is the first ti
me I have studied through a distant learning course and the materials you receive ar
e enough to get you through the course. It was a really good course and the instruct
or was flexible with being able to complete the required course work on time. Bigges
t mistake I made was going with this course provider , I had to pay about £600 for m
y course plus pay for exam fees every time . Material, course content, late exam I l
iked the training delivery and also the set up however the course did feel a little
rushed.
```

1.2. Weighted Word from Each Company (no lemmatization)

In [40]:

```
companies = list(set(df['itemReviewed_name']))

for company in companies:
    print('Summary for {0}:'.format(company))
    df_name = 'df_{0}'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name]['review
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    summary = weighted_occurence_summarisation(feedback_combined, sentences, 7, Fals
```

```
    # recall: how much the words in the reference appeared in the summaries
    # precision: how much the words in the summary appeared in the reference
    # f1 score: 2*(precision*recall)/(precision+recall)

    rouge = Rouge()
    score_rouge = rouge.get_scores(summary, reference)

    print(score_rouge[0]['rouge-1'])
    print(summary, '\n')
```

```
Summary for BPP:
{'f': 0.0967494231898788, 'p': 1.0, 'r': 0.05083378160301237}
To anyone doing ACA exams and thinking of studying with BPP, especially towards the
later or more harder ACA exams, I would strongly advise you to not use BPP. Which ta
kes even more time and there is always a risk you may never get this second feedback
due to how poor BPP is run operationally. There were also many other parts of the co
urse that were overlooked, this has happened with other courses too at BPP I have be
en on at BPP. BPP give false hope to students (inc. letting students with a 2.2 onto
the course, who have almost no chance of getting a TC at any City firms). To anyone
reading this review I was hoping it would give you some insight into how my experien
ce studying for my ACA exams at BPP has been. 7 phone calls waiting on hold for a lo
ng time for them to finally send my study material, by which time I had two lectures
without any material. The library facilities are very poor and the tutors unresponsi
ve to correspondence - emails seem to get conveniently 'lost' in the BPP system!

Summary for Udemy:
{'f': 0.007707871587155137, 'p': 1.0, 'r': 0.0038688461166457105}
Sriranga Amazing, I can't get enough courses Some courses are very expensive. I have
been using Udemy for long time and am very happy with the courses and the customer s
ervice.The courses are for lifetime, always updated and never expire. I have done ma
ny courses on udemy, some courses are good and some are bad, that isn't udemys faul
t. I think Udemy is getting too crowded with cheap marketing as it's getting to diff
icult to find a really good, valuable course Their courses are incredibly cheap!!!

Summary for Coursera:
{'f': 0.025379915149703555, 'p': 1.0, 'r': 0.012853062519834974}
Some Courses are good, some are really bad.I enrolled in some courses with my colleg
e email id to get premium courses free. I would say that one Coursera course corresp
onds roughly to half of one high-level university course. Used to be good when they
offered course materials free, but not you can even see the materials for only one f
ree week. I know a lot of work goes into creating courses, but without considered co
urse delivery and implementation the course is little more than chunked information.
I was particularly interested in photography courses, and there are a lot of them.If
I have free time, I use it to view the course on this platform. I tried a free cours
e just to learn the info, and I'm so glad I don't have to take a "real" course from
them. This review is based on Coursera as a course provider relating to consumer saf
ety, and not the academic quality of the courses.

Summary for edX:
{'f': 0.008395596188540513, 'p': 1.0, 'r': 0.004215493928628241}
It was great experience of learning at edx with current updated educationcal system
I have done 2 courses via edX and have had a very positive experience. EdX is a fant
astic learning platform After ten years of not having studied an online course, I've
decided to join a certification program over the Edx platform. hands down It was a g
ood experience Great experience, learned a lot I had a great experience with my edx
courses. edX is a great way to keep on learning new skills Great e-learning platform
with great quality content! I have experience to join edx really gud nd nice im soo
happy for that thnks soo much edx Great Course. Edx is the future world of learning
.. good luck EDX, the best way to base your opinion on knowledge. Great learning exp
erience, course material is easy to follow and supporting materials help with furthe
r understanding Edx has been my website of choice for personal development.

Summary for Udacity:
{'f': 0.006409738429550177, 'p': 1.0, 'r': 0.003215173457257106}
Great course! It's a great program, I've learned more and more useful information th
at makes me feel like I'm progressing towards a professionalism The program's great
so far. thnks for every thing The Data Analyst nanodegree program is designed well w
ith a good practice through the course which is the perfect way to learn, highly rec
```

```
Summary for DPG plc:
{'f': 0.03487894918540155, 'p': 1.0, 'r': 0.01774900814366256}
Sean was really helpful and responded to my questions promptly Amazing service  E
asy, friendly, efficient - fab Advisor was helpful and responded really promptly! Gr
eat service, easy to use and great advisors, what more could anyone ask for Rodin wa
s quick to respond to any questions I had and was helpful and informative. Very help
ful support as I have already achieved some modules so needed information Sean was v
ery helpful and the course was a great price ! Great service Knowledgable- kind - fr
iendly Really helpful, reassuring and encouraging discussions with DPG following my
initial enquiry about continuing with my CIPD studies. Quick and Easy to sign up wit
h DPG Easy enrolment process, looking forward to starting my HR course. thanks DPG
:) Super helpful, knowledgeable and easy to enrol, highly recommend The DPG team wer
e always happy to help and were quick to respond. I haven't started my course yet be
the advice and service I have been provided with when choosing and purchasing a cour
se was great.

Summary for Deloitte:
{'f': 0.44052863092627453, 'p': 1.0, 'r': 0.2824858757062147}
The most dishonest unprofessional company I've ever been involved with Very bad comp
any Great study on private equity valuations. We were working together on tax matte
r. It was a great experience. They are very reliable and updated. They are reliable
and professional. I've had the opportunity to work with them.

Summary for General Assembly:
{'f': 0.1978021960198044, 'p': 1.0, 'r': 0.10975609756097561}
I wasn't a student, I was a teacher here.
```

In [60]:

```python
# This cell is just for exploration
word_occurences = {}
feedback_combined = ' '.join(str(body) for body in df_companies['df_Avado']['reviewB
reference = ' '.join(str(body).replace('\n','') for body in df_companies['df_Avado']
sentences = sent_tokenize(reference)
for word in word_tokenize(feedback_combined):
    if word not in word_occurences.keys():
        word_occurences[word] = 1
    else:
        word_occurences[word] += 1

print('First 5 word occurrences:')
n = 5
for key, value in word_occurences.items():
    print(key, value)
    if n==0: break
    n-=1

highest_occurence = max(word_occurences.values())
print('\nHighest occurrence = ', highest_occurence)

for word in word_occurences.keys():
    word_occurences[word] = (word_occurences[word]/highest_occurence)

print('\nFirst 5 weighted word occurrences:')
n = 5
for key, value in word_occurences.items():
    print(key, value)
    if n==0: break
    n-=1

sentence_scores = {}
for sent in sentences:
    for word in word_tokenize(sent.lower()):
        if word in word_occurences.keys():
            if len(sent.split(' ')) < 30: # to filter out sentences
                if sent not in sentence_scores.keys():
                    sentence_scores[sent] = word_occurences[word]
```

```
                else:
                    sentence_scores[sent] += word_occurences[word]

print('\nTop high scored sentences:')
sorted_sentences = dict(sorted(sentence_scores.items(), key=lambda item: item[1], re
n = 17
for key, value in sorted_sentences.items():
    print(key, value)
    if n==0: break
    n-=1
```

```
First 5 word occurrences:
speaking 11
one 148
advisors 18
phone 135
felt 52
answers 30

Highest occurrence =  1249

First 5 weighted word occurrences:
speaking 0.008807045636509208
one 0.11849479583666933
advisors 0.014411529223378704
phone 0.10808646917534027
felt 0.041633306645316254
answers 0.02401921537229784

Top high scored sentences:
Thoroughly enjoyed doing my course via Avado The course material was excellent, the
portal looked very professional and I had great tutor support throughout my course.
4.835068054443555
honest and easy to understand process - would recommend Alison was really helpful wi
th booking the course Great learning and experience and online set up! 4.37870296236
9896
The course advisors at Avado have been extremely helpful, and all of the course info
rmation sent through so far has been very informative, clear and easy to understand.
4.267413931144916
So easy to start a course, well explained, course is easy to work your way through V
ery helpful and informative. 4.10568454763811
Excellent communication and efficient service Great customer service David was a gre
at explaining the course and making the enrolment process really clear and easy. 4.0
33626901521218
INCORRECT COURSE MATERIAL - The course would often provide guidelines that were need
ed to complete assignments for the course then assessors would fail the assignment f
or using their guidelines.5. 3.9607686148919137
Really easy process, with very helpful account managers I didn’t start the course  y
et, so I cannot say anything about the course. 3.943154523618895
This course was really simple to set up, great communication, the course looks great
and the site looks easy to navigate round. 3.7670136108887107
Great learning systems Very easy sign up process, fully explained the course and see
med very knowledgable about the course. 3.7157726180944755
Tutor support is a bit hit and miss Great student support, easy to use website and g
reat course design, very easy to keep track of what you have completed. 3.6901521216
97358
I am really enjoying my course and would not hesitate in recommending AVADO to those
who feel an online digital course is for them. 3.622097678142514
Avado staff were very helpful in explaining the course sign up process and how the c
ourse is structured. 3.578062449959968
I would definitely recommend Avado. 3.4539631705364293
I found it very easy to enrol on this course, with helpful advisors who would keep i
n touch whilst I sorted out the funding for my course. 3.425140112089672
Excellent course, great tutor with a really good support service and easy to use web
site. 3.3835068054443553
Can’t wait to start the course in a few weeks Quick answer, quick proposal, great se
rvice Straight forward process with a most helpful advisor supporting you along the
way. 3.351481184947958
Loretta made the 14 months course (alongside a full time job) very worthwhile.I woul
```

```
d highly recommend Avado to anyone wanting to take on distance learning from home!
3.2562049639711774
Very simple procedures extremely helpful and friendly staff I did level 2 with Avad
o, it was a really good course. 3.241793434747798
```

1.3. Weighted Word from Each Company and Sentiment (no lemmatization)

In [42]:
```
companies = list(set(df['itemReviewed_name']))

for company in companies:
    print('Summary for {0} - good feedback:'.format(company))
    df_name_good = 'df_{0}_good'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name_good]['r
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    summary = weighted_occurence_summarisation(feedback_combined, sentences, 7, Fals
    rouge = Rouge()
    if len(reference) > 0:
        score_rouge = rouge.get_scores(summary, reference)
        print(score_rouge[0]['rouge-1'])
    print(summary, '\n')

    print('Summary for {0} - bad feedback:'.format(company))
    df_name_bad = 'df_{0}_bad'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name_bad]['re
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    summary = weighted_occurence_summarisation(feedback_combined, sentences, 7, Fals
    rouge = Rouge()
    if len(reference) > 0:
        score_rouge = rouge.get_scores(summary, reference)
        print(score_rouge[0]['rouge-1'])
    print(summary, '\n')
```

```
Summary for BPP - good feedback:
N/A

Summary for BPP - bad feedback:
{'f': 0.0967494231898788, 'p': 1.0, 'r': 0.05083378160301237}
To anyone doing ACA exams and thinking of studying with BPP, especially towards the
later or more harder ACA exams, I would strongly advise you to not use BPP. Which ta
kes even more time and there is always a risk you may never get this second feedback
due to how poor BPP is run operationally. There were also many other parts of the co
urse that were overlooked, this has happened with other courses too at BPP I have be
en on at BPP. BPP give false hope to students (inc. letting students with a 2.2 onto
the course, who have almost no chance of getting a TC at any City firms). To anyone
reading this review I was hoping it would give you some insight into how my experien
ce studying for my ACA exams at BPP has been. 7 phone calls waiting on hold for a lo
ng time for them to finally send my study material, by which time I had two lectures
without any material. The library facilities are very poor and the tutors unresponsi
ve to correspondence - emails seem to get conveniently 'lost' in the BPP system!

Summary for Udemy - good feedback:
{'f': 0.013488551779766187, 'p': 1.0, 'r': 0.006790070063143327}
Sriranga Amazing, I can't get enough courses Thank you to Udemy for a professional f
abulous looking site that works. The Udemy Courses are very good, I can say that Ude
my is the Internet University, where everybody have to go in order to learn whatever
at very cheaper prices. I have done many courses on udemy, some courses are good and
some are bad, that isn't udemys fault. I have  been using Udemy for long time and am
very happy with the courses and the customer service.The courses are for lifetime, a
lways updated and never expire. I am personally benefitted from the udemy courses by
Experienced  Instructors who are delivering their online courses.

Summary for Udemy - bad feedback:
{'f': 0.018581906906371917, 'p': 1.0, 'r': 0.009378084896347482}
I think Udemy is getting too crowded with cheap marketing as it's getting to difficu
```

```
{'f': 0.04254804745012506, 'p': 1.0, 'r': 0.02173644559713474}
Great service, easy to use and great advisors, what more could anyone ask for Rodin
was quick to respond to any questions I had and was helpful and informative. Great s
ervice Knowledgable- kind - friendly Really helpful, reassuring and encouraging disc
ussions with DPG following my initial enquiry about continuing with my CIPD studies.
thanks DPG :) Super helpful, knowledgeable and easy to enrol, highly recommend The D
PG team were always happy to help and were quick to respond. Very helpful support as
I have already achieved some modules so needed information Sean was very helpful and
the course was a great price ! Quick and Easy to sign up with DPG Easy enrolment pro
cess, looking forward to starting my HR course. Really looking forward to getting st
arted The Sales Manager (Calire Smeaton) was great help, answered all my questions w
ith smile, Thank you :) Excellent communication, quick start. Friendly, clear and go
od guidance and very efficiently arranged  my registration to start the level 5 HR D
iploma course Just registered for CIPD Level 5 Diploma in HRM.

Summary for DPG plc - bad feedback:
{'f': 0.1871559615963939, 'p': 0.9935064935064936, 'r': 0.10330857528696827}
I enquired about workshop and exam days prior to booking the course and paying my hu
ge costs as I knew that I would be getting married. I would still like to attend the
March group and feel that I should not have to suffer due to the poor service of the
consultants representing your company. Unfortunately, their 3 weeks period of waitin
g to receive assessment results is really frustrating especially as other providers
are able to mark assessment within just a few hours! I purchased the level 5 diploma
and after I completed the course, I was issued with the certificate. The system also
isn’t the best and I had issues from time to time with it and had to call up and req
uest support with errors. The worst experience in a course, read before enrol, pleas
e! I've got so far as to sign up for a course and that process has been satisfactor
y.

Summary for Deloitte - good feedback:
{'f': 0.999999995, 'p': 1.0, 'r': 1.0}
Great study on private equity valuations. We were working together on tax matter. Th
ey are very reliable and updated. It was a great experience. They are reliable and p
rofessional. I've had the opportunity to work with them.

Summary for Deloitte - bad feedback:
{'f': 0.16993463896791833, 'p': 1.0, 'r': 0.09285714285714286}
The most dishonest unprofessional company I've ever been involved with Very bad comp
any

Summary for General Assembly - good feedback:
{'f': 0.1978021960198044, 'p': 1.0, 'r': 0.10975609756097561}
I wasn't a student, I was a teacher here.

Summary for General Assembly - bad feedback:
N/A
```

1.4. Weighted Word from Whole Dataset by Sentiment (with lemmatization)

In [48]:

```
good_feedback_combined = ' '.join(str(body) for body in df_good['lemmatized_reviewBo
reference = ' '.join(str(body).replace('\n','') for body in df_good['reviewBody'].to
sentences = sent_tokenize(reference)
summary = weighted_occurence_summarisation(good_feedback_combined, sentences, 7, Tru

# recall: how much the words in the reference appeared in the summaries
# precision: how much the words in the summary appeared in the reference
# f1 score: 2*(precision*recall)/(precision+recall)

rouge = Rouge()
score_rouge = rouge.get_scores(summary, reference)

print(score_rouge[0]['rouge-1'])
print(summary)
```

```
{'f': 0.0004426475157027678, 'p': 1.0, 'r': 0.00022137275511466934}
```

```
Excellent customer service. Interesting courses available Great choice of courses an
d very easy to enrol An easy to follow well thought out course and very informative.
Really excited to work on with this course Good Course & Helpful staff.Very easy and
stress-free sign up, all information provided to make easy to set up and login. I le
arned a lot from their copywriting course, creative course, marketing course and UI/
UX design course. Really helpful, easy to enrol, changed my course before I started,
this was no hassle at all... Great course, really informative, loads of help. Keep u
p the good work! Thoroughly enjoyed doing my course via Avado The course material wa
s excellent, the portal looked very professional and I had great tutor support throu
ghout my course.
```

In [49]:

```
bad_feedback_combined = ' '.join(str(body) for body in df_bad['lemmatized_reviewBody
reference = ' '.join(str(body).replace('\n','') for body in df_bad['reviewBody'].to_
sentences = sent_tokenize(reference)
summary = weighted_occurence_summarisation(bad_feedback_combined, sentences, 7, True

# recall: how much the words in the reference appeared in the summaries
# precision: how much the words in the summary appeared in the reference
# f1 score: 2*(precision*recall)/(precision+recall)

rouge = Rouge()
score_rouge = rouge.get_scores(summary, reference)

print(score_rouge[0]['rouge-1'])
print(summary)
```

```
{'f': 0.0018295262084558224, 'p': 1.0, 'r': 0.0009156006710876595}
INCORRECT COURSE MATERIAL - The course would often provide guidelines that were need
ed to complete assignments for the course then assessors would fail the assignment f
or using their guidelines.5. Some Courses are good, some are really bad.I enrolled i
n some courses with my college email id to get premium courses free. I know a lot of
work goes into creating courses, but without considered course delivery and implemen
tation the course is little more than chunked information. The course highlights onc
e you get a free nail kit with the course after paying at least 50% for the course w
hich is fine by me. I took one course 3 years ago and didn't like it, I took another
course recently but they are just getting worse. Firebrand do offer good courses and
accelerated training, I have had 3 courses with Firebrand over the course of my appr
enticeship. This is the first time I have studied through a distant learning course
and the materials you receive are enough to get you through the course.
```

1.5. Weighted Word from Each Company (with lemmatization)

In [46]:

```
companies = list(set(df['itemReviewed_name']))

for company in companies:
    print('Summary for {0}:'.format(company))
    df_name = 'df_{0}'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name]['lemmat
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    summary = weighted_occurence_summarisation(feedback_combined, sentences, 7, True

    # recall: how much the words in the reference appeared in the summaries
    # precision: how much the words in the summary appeared in the reference
    # f1 score: 2*(precision*recall)/(precision+recall)

    rouge = Rouge()
    score_rouge = rouge.get_scores(summary, reference)

    print(score_rouge[0]['rouge-1'])
    print(summary, '\n')
```

```
Summary for BPP:
{'f': 0.09626215986200023, 'p': 1.0, 'r': 0.05056481979558903}
BPP give false hope to students (inc. letting students with a 2.2 onto the course, w
ho have almost no chance of getting a TC at any City firms). Which takes even more t
ime and there is always a risk you may never get this second feedback due to how poo
```

```
ort as I have already achieved some modules so needed information Sean was very help
ful and the course was a great price ! Caitlin was very helpful in getting me succes
sfully enrolled onto my course, I'm looking forward to getting started with my studi
es.

Summary for Deloitte:
{'f': 0.44052863092627453, 'p': 1.0, 'r': 0.2824858757062147}
The most dishonest unprofessional company I've ever been involved with Very bad comp
any We were working together on tax matter. Great study on private equity valuation
s. It was a great experience. They are very reliable and updated. I've had the oppor
tunity to work with them. They are reliable and professional.

Summary for General Assembly:
{'f': 0.1978021960198044, 'p': 1.0, 'r': 0.10975609756097561}
I wasn't a student, I was a teacher here.
```

In [61]:

```
# This cell is just for exploration
word_occurences = {}
feedback_combined = ' '.join(str(body) for body in df_companies['df_Avado']['lemmati
reference = ' '.join(str(body).replace('\n','') for body in df_companies['df_Avado']
sentences = sent_tokenize(reference)
for word in word_tokenize(feedback_combined):
    if word not in word_occurences.keys():
        word_occurences[word] = 1
    else:
        word_occurences[word] += 1

print('First 5 word occurrences:')
n = 5
for key, value in word_occurences.items():
    print(key, value)
    if n==0: break
    n-=1

highest_occurence = max(word_occurences.values())
print('\nHighest occurrence = ', highest_occurence)

for word in word_occurences.keys():
    word_occurences[word] = (word_occurences[word]/highest_occurence)

print('\nFirst 5 weighted word occurrences:')
n = 5
for key, value in word_occurences.items():
    print(key, value)
    if n==0: break
    n-=1

sentence_scores = {}
for sent in sentences:
    for word in word_tokenize(sent.lower()):
        word = lemmatize_word(word)
        if word in word_occurences.keys():
            if len(sent.split(' ')) < 30: # to filter out sentences
                if sent not in sentence_scores.keys():
                    sentence_scores[sent] = word_occurences[word]
                else:
                    sentence_scores[sent] += word_occurences[word]

print('\nTop high scored sentences:')
sorted_sentences = dict(sorted(sentence_scores.items(), key=lambda item: item[1], re
n = 17
for key, value in sorted_sentences.items():
    print(key, value)
    if n==0: break
    n-=1
```

```
First 5 word occurrences:
speak 33
one 152
advisor 78
phone 144
felt 52
answer 259

Highest occurrence =  1314

First 5 weighted word occurrences:
speak 0.02511415525114155
one 0.1156773211567732
advisor 0.0593607305936073
phone 0.1095890410958904
felt 0.0395738203957382
answer 0.19710806697108066

Top high scored sentences:
Thoroughly enjoyed doing my course via Avado The course material was excellent, the
portal looked very professional and I had great tutor support throughout my course.
5.172754946727549
honest and easy to understand process - would recommend Alison was really helpful wi
th booking the course Great learning and experience and online set up! 4.40410958904
1097
So easy to start a course, well explained, course is easy to work your way through V
ery helpful and informative. 4.373668188736682
INCORRECT COURSE MATERIAL - The course would often provide guidelines that were need
ed to complete assignments for the course then assessors would fail the assignment f
or using their guidelines.5. 4.358447488584476
Excellent communication and efficient service Great customer service David was a gre
at explaining the course and making the enrolment process really clear and easy. 4.3
40943683409436
The course advisors at Avado have been extremely helpful, and all of the course info
rmation sent through so far has been very informative, clear and easy to understand.
4.254185692541856
Really easy process, with very helpful account managers I didn't start the course  y
et, so I cannot say anything about the course. 4.121004566210045
This course was really simple to set up, great communication, the course looks great
and the site looks easy to navigate round. 4.101978691019786
Can't wait to start the course in a few weeks Quick answer, quick proposal, great se
rvice Straight forward process with a most helpful advisor supporting you along the
way. 3.9893455098934556
Tutor support is a bit hit and miss Great student support, easy to use website and g
reat course design, very easy to keep track of what you have completed. 3.9307458143
07458
Great learning systems Very easy sign up process, fully explained the course and see
med very knowledgable about the course. 3.8356164383561646
I am really enjoying my course and would not hesitate in recommending AVADO to those
who feel an online digital course is for them. 3.7732115677321154
Avado staff were very helpful in explaining the course sign up process and how the c
ourse is structured. 3.664383561643836
Learning in this way will take a little while to get used to but Avado have made it
very easy to get started with online learning. 3.6598173515981736
Once you get started, it's really easy to navigate the virtual learning course and g
et stuck in the activities to aid your learning. 3.6164383561643834
Loretta made the 14 months course (alongside a full time job) very worthwhile.I woul
d highly recommend Avado to anyone wanting to take on distance learning from home!
3.5616438356164384
I found it very easy to enrol on this course, with helpful advisors who would keep i
n touch whilst I sorted out the funding for my course. 3.5365296803652964
This is the first time I have studied through a distant learning course and the mate
rials you receive are enough to get you through the course. 3.5091324200913245
```

1.6. Weighted Word from Each Company and Sentiment (with lemmatization)

In [47]:
```
companies = list(set(df['itemReviewed_name']))
```

```
for company in companies:
    print('Summary for {0} - good feedback:'.format(company))
    df_name_good = 'df_{0}_good'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name_good]['l
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    summary = weighted_occurence_summarisation(feedback_combined, sentences, 7, True
    rouge = Rouge()
    if len(reference) > 0:
        score_rouge = rouge.get_scores(summary, reference)
        print(score_rouge[0]['rouge-1'])
    print(summary, '\n')

    print('Summary for {0} - bad feedback:'.format(company))
    df_name_bad = 'df_{0}_bad'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name_bad]['le
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    summary = weighted_occurence_summarisation(feedback_combined, sentences, 7, True
    rouge = Rouge()
    if len(reference) > 0:
        score_rouge = rouge.get_scores(summary, reference)
        print(score_rouge[0]['rouge-1'])
    print(summary, '\n')
```

```
Summary for BPP - good feedback:
N/A

Summary for BPP - bad feedback:
{'f': 0.09626215986200023, 'p': 1.0, 'r': 0.05056481979558903}
BPP give false hope to students (inc. letting students with a 2.2 onto the course, w
ho have almost no chance of getting a TC at any City firms). Which takes even more t
ime and there is always a risk you may never get this second feedback due to how poo
r BPP is run operationally. For students booking themselves onto their courses, look
out for their terms and conditions if you do book courses with them as they can be q
uite unfavourable. There were also many other parts of the course that were overlook
ed, this has happened with other courses too at BPP I have been on at BPP. To anyone
reading this review I was hoping it would give you some insight into how my experien
ce studying for my ACA exams at BPP has been. 7 phone calls waiting on hold for a lo
ng time for them to finally send my study material, by which time I had two lectures
without any material. I paid for Audit and Assurance course material (ACCA) on 16th
Sep, I only received my books in 4 weeks time after several chasers.

Summary for Udemy - good feedback:
{'f': 0.014597286477108685, 'p': 1.0, 'r': 0.0073523051639131564}
I learned a lot from their copywriting course, creative course, marketing course and
UI/UX design course. Courses are varied - that's the beauty - real people creating r
eal course (mostly) and not everyone is an expert at creating courses but there's so
much to learn. I am always looking for new courses to register and learn @Udemy My e
xperience with Udemy all started when I was looking on how to make games without cod
ing. The learning courses were informative and in some courses you get quizzes and a
ssignment to program to use what you may have learned. I didn't complete the course
for a long time and ignored Udemy's emails because I did not want to pay the normal
course prices. The Udemy Courses are very good, I can say that Udemy is the Internet
University, where everybody have to go in order to learn whatever at very cheaper pr
ices. Now, with Udemy you buy a course and have the course, the instructor and fello
w students for a life time.

Summary for Udemy - bad feedback:
{'f': 0.019012439379285772, 'p': 1.0, 'r': 0.009597455303279587}
I think Udemy is getting too crowded with cheap marketing as it's getting to difficu
lt to find a really good, valuable course Their courses are incredibly cheap!!! Als
o, sometimes when access the same course using different browsers in private / incog
nito mode, I get different prices for the SAME COURSE. I looked elsewhere and found
a course by 'quicked courses' that was free and actually better than the paid one by
Udemy!!! I'm getting $1-2 per course sale.They claim that Apple charges 30% of the p
rice of the course, when purchased WITHIN the app.They lie and make stuff up. Great
```

```
Summary for DPG plc - good feedback:
{'f': 0.041837968151436485, 'p': 1.0, 'r': 0.021365938001729035}
Quick and Easy to sign up with DPG Easy enrolment process, looking forward to starti
ng my HR course. Great service, easy to use and great advisors, what more could anyo
ne ask for Rodin was quick to respond to any questions I had and was helpful and inf
ormative. thanks DPG :) Super helpful, knowledgeable and easy to enrol, highly recom
mend The DPG team were always happy to help and were quick to respond. Great service
Knowledgable- kind - friendly Really helpful, reassuring and encouraging discussions
with DPG following my initial enquiry about continuing with my CIPD studies. Very he
lpful support as I have already achieved some modules so needed information Sean was
very helpful and the course was a great price ! I haven't started my course yet be t
he advice and service I have been provided with when choosing and purchasing a cours
e was great. Friendly, clear and good guidance and very efficiently arranged  my reg
istration to start the level 5 HR Diploma course Just registered for CIPD Level 5 Di
ploma in HRM.

Summary for DPG plc - bad feedback:
{'f': 0.21566264867833507, 'p': 1.0, 'r': 0.12086428089128967}
I enquired about workshop and exam days prior to booking the course and paying my hu
ge costs as I knew that I would be getting married. I would still like to attend the
March group and feel that I should not have to suffer due to the poor service of the
consultants representing your company. Unfortunately, their 3 weeks period of waitin
g to receive assessment results is really frustrating especially as other providers
are able to mark assessment within just a few hours! I've got so far as to sign up f
or a course and that process has been satisfactory. The system also isn’t the best a
nd I had issues from time to time with it and had to call up and request support wit
h errors. I feel let down and disappointed, that the consultant did not have the cou
rtesy to call me regarding this and felt that an email was good enough! At no point
in time did any of the two consults advise that I could loose my place if the paper
work was not handed in.

Summary for Deloitte - good feedback:
{'f': 0.999999995, 'p': 1.0, 'r': 1.0}
Great study on private equity valuations. We were working together on tax matter. Th
ey are very reliable and updated. I've had the opportunity to work with them. It was
a great experience. They are reliable and professional.

Summary for Deloitte - bad feedback:
{'f': 0.16993463896791833, 'p': 1.0, 'r': 0.09285714285714286}
The most dishonest unprofessional company I've ever been involved with Very bad comp
any

Summary for General Assembly - good feedback:
{'f': 0.1978021960198044, 'p': 1.0, 'r': 0.10975609756097561}
I wasn't a student, I was a teacher here.

Summary for General Assembly - bad feedback:
N/A
```

2. TF-IDF Method

The difference with the weighted word occurences method should be that in TF-IDF, highly frequent words (excluding stopwords) in many sentences are penalised. Hence, the scoring algorithm is different (using TD-IDF method) which processes per sentence rather than per document.

In [50]:
```
def tf_idf_summarisation(combined_cleaned_feedback, original_feedback_sentences, n_s
    '''
    combined_cleaned_feedback: a string of all cleaned feedback combined (with or wi
    original_feedback_sentences: an array of original feedback sentences
    n_sentence: n number of sentences to be taken to form the summary
    is_lemmatized: whether to lemmatize words in calculating the word occurrences an
```

```
    original_lemmatized_feedback_sentences: an array of original lemmatized feedback
    '''

    # calculate inverse document frequency idf
    # IDF(w) = log(Total number of sentences / Number of sentences with word w in it
    word_idf = {}
    for word in word_tokenize(combined_cleaned_feedback):
        if word in word_idf.keys():
            continue
        contained_sentence = 0

        if is_lemmatized:
            for sent in original_lemmatized_feedback_sentences:
                if word in sent.lower():
                    contained_sentence += 1
        else:
            for sent in original_feedback_sentences:
                if word in sent.lower():
                    contained_sentence += 1

        if contained_sentence == 0: contained_sentence += 1 # smoothing, to avoid ze

        word_idf[word] = np.log(len(original_feedback_sentences)/contained_sentence)

    # calculate sentence score
    sentence_scores = {}

    for sent in original_feedback_sentences:
        sent_cleaned = " ".join(re.findall("[a-zA-Z]+", sent)).lower()
        sent_cleaned = " ".join([i for i in sent_cleaned.split() if i not in stopwor
        sentence_scores[sent] = calculate_sentence_score(sent_cleaned, word_idf, is_

    sorted_sentences = dict(sorted(sentence_scores.items(), key=lambda item: item[1]

    # take n number of sentences
    summary = []
    n = 0
    for key, value in sorted_sentences.items():
        if n == n_sentence: break
        if value <= 15: # filter out too high score to avoid short generic feedback,
            summary.append(key)
            n += 1

    return ' '.join(summary)

# calculate sentence score
def calculate_sentence_score(sentence, word_idf, is_lemmatized):
    sentence_score = 0
    tokenized_sentence = word_tokenize(sentence)
    for word in tokenized_sentence:
        # calculate term frequency tf
        # TF(w) = (Number of times word w appears in a sentence) / (Total number of
        tf = sentence.count(word)/len(tokenized_sentence)

        # get the IDF score from dictionary
        if is_lemmatized:
            word = lemmatize_word(word)
        if word in word_idf.keys():
            idf = word_idf[word]
        else:
            idf = 0

        sentence_score += tf*idf
    return sentence_score
```

2.1. TF-IDF from Whole Dataset by Sentiment (no lemmatization)

In [51]:
```
good_feedback_combined = ' '.join(str(body) for body in df_good['reviewBody_clean'].
reference = ' '.join(str(body).replace('\n','') for body in df_good['reviewBody'].to
sentences = sent_tokenize(reference)
summary = tf_idf_summarisation(good_feedback_combined, sentences, 7, False, None)

# recall: how much the words in the reference appeared in the summaries
# precision: how much the words in the summary appeared in the reference
# f1 score: 2*(precision*recall)/(precision+recall)

rouge = Rouge()
score_rouge = rouge.get_scores(summary, reference)

print(score_rouge[0]['rouge-1'])
print(summary)
```

```
{'f': 0.001198526914775933, 'p': 0.9971014492753624, 'r': 0.0005996238406255611}
The Kafka module covers the majority of components/tools used in Kafka ecosystem, th
e Teacher has a very good knowledge and excellent slides complemented by code exampl
es to explain how Kafka works.The project gave me an excellent overview about kafka
and related components of kafka ecosystem. !Leaders inspire other leaders to make mo
re leaders that’s the only way! It was a wonderful education journey…It was a wonder
ful education journey that will satisfy my needs and meet all my expectations,In Dig
ital Marketing Nanodegree they will teach all points and have a great practicing wit
h live monitoring for all the sectionsThank you Udacity It was a wonderful education
journey…It was a wonderful education journey that will satisfy my needs and meet all
my expectations,In Digital Marketing Nanodegree they will teach all points and have
a great practicing with live monitoring for all the sectionsThank you Udacity I jump
ed into the nanodegree course not knowing what to expect. My favorite course was Set
h Godin's "Seth Godin's Freelancer Course." Kylee and Prajwala work so well together
and Kylee misses Prajwala days they don’t work together. The only criticism I have i
s that on the Dashboard, when you click on a particular Playlist, instead of taking
you to that particular Playlist, it takes you to a list of all your Playlists and yo
u have to find the one you are looking for from that list. i learned following after
completing my fist module.• Understating the business problem and getting the right
data to solve that problem is the most important and toughest job.• Find an approach
using A.I to solve business problem to get the solution.• Come up with a strategy to
get the data related to that problem• Strategy to clean that data.• Model selection
process / approach.• Need to have a clear path to deploy your model.• Strategy how t
his model will integrate into business process.• Some sort of monitoring process in
production to check if your model is learning and acting the way it should be in pro
duction.
```

In [52]:
```
bad_feedback_combined = ' '.join(str(body) for body in df_bad['reviewBody_clean'].to
reference = ' '.join(str(body).replace('\n','') for body in df_bad['reviewBody'].to_
sentences = sent_tokenize(reference)
summary = tf_idf_summarisation(bad_feedback_combined, sentences, 7, False, None)

# recall: how much the words in the reference appeared in the summaries
# precision: how much the words in the summary appeared in the reference
# f1 score: 2*(precision*recall)/(precision+recall)

rouge = Rouge()
score_rouge = rouge.get_scores(summary, reference)

print(score_rouge[0]['rouge-1'])
print(summary)
```

```
{'f': 0.0017555641160730056, 'p': 1.0, 'r': 0.000878553245089893}
if instead of writing < i glyphicon glypohicon-world > < / i > you write  < i  glyph
icon glypohicon-world / > In my instance I only got 5 distinct instances of the icon
instead of the expected 1.
```

2.2. TF-IDF from Each Company (no lemmatization)

In [53]:
```
companies = list(set(df['itemReviewed_name']))

for company in companies:
    print('Summary for {0}:'.format(company))
    df_name = 'df_{0}'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name]['review
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    summary = tf_idf_summarisation(feedback_combined, sentences, 7, False, None)
    # recall: how much the words in the reference appeared in the summaries
    # precision: how much the words in the summary appeared in the reference
    # f1 score: 2*(precision*recall)/(precision+recall)

    rouge = Rouge()
    score_rouge = rouge.get_scores(summary, reference)

    print(score_rouge[0]['rouge-1'])
    print(summary, '\n')
```

```
Summary for BPP:
{'f': 0.26261682014857635, 'p': 1.0, 'r': 0.1511565357719204}
I then rang customer service who said that I had emailed the wrong email address and
should have emailed the first address which is why it never got booked, even though
I had emailed that address in the first place.There has been another occasion where
i emailed in and didn't get a response after 2 weeks.I have had a number of calls to
customer service where they have been rude and unhelpful, and in one case I was told
an issue would be rectified, I rang again a few days later (a lovely lady answered)
and she informed me that the previous call handler didn't even log an issue in the f
irst place.I have had issues with access to my hub account, which have taken 2 weeks
to sort, which meant i was 2 weeks behind on my course as i had no access to learnin
g materials; however they refused to move me to a course which had a later start dat
e, so I have been unable to qualify for pass assurance.On a number of times I have t
elephoned due to an urgent matter i have been told to 'email in with urgent as the r
eference and it will get sorted today' and then when I finally have a response two d
ays later, I am told I should have rang in instead of emailed because they cannot re
spond to urgent emails quick enough.I have been emailed the incorrect dates for my c
urrent revision course, which I am now unable to attend one of the days because of t
his.They are extremely slow in marking the 'Step 4 and 6' which are meant to aid you
r learning and give feedback to help you improve your written answers. The idea is t
hat you have feedback from Step 4 before you do Step 6, but I have even waited until
after the Step 6 deadline and my Step 4 still hadn't been marked.Overall, I wish it
would be simple enough to change apprenticeship providers because trying to deal wit
h customer service at BPP is extremely stressful and makes my whole experience compl
etely unenjoyable. I had emailed in to book to the correct email address, where they
informed me that I should email a different address. Avoid, avoid, avoid!They are gr
ossly incompetent and their Customer Service is something I have not come across bef
ore. Good for masochists who like to be belittled and treated like scum by peopole w
ho think they are special, but not so good for people wanting a professional qualifi
cation delivered by professionals. Happy to take your money but not happy to fulfill
their end of the bargain I thankfully only bought a book not an exam or an apprentic
eship, however I advice you to avoid avoid!I ONLY bought it because they said they h
ad an app which hasn’t been updated for the past 8 MONTHS!However it doesn’t open, y
ou can hardly download it.Not to mention you can’t even do exams in it!And to top it
off you can’t even use it on different devices at the same time!If i knew this befor
e I would not have spent my money on them.If you can avoid them by all means.😠😠😠
😠😠😠😠😠😠 Having decided to pursue ACCA after years out of education, I decided t
o book my CBAs at BPP Liverpool Street. Their marketing is just fluff.

Summary for Udemy:
{'f': 0.005257445986391671, 'p': 1.0, 'r': 0.0026356514169648903}
I'm seriously invested, and I'm seriously let down. My favorite course was Seth Godi
n's "Seth Godin's Freelancer Course." LOVE LOVE LOVE Udemy! Continued learning is ke
y to continued growth and brain functionality. When you find a gem you really find a
gem. The mobile browser website is barely functional and they have for some reason d
one the same with the mobile site as they have done with the mobile app and limited
what you can do on the mobile version considerably despite a large portion of their
audience enjoying courses from the comfort of their smartphones. PRICING METHODS- Th
e first negative aspect of Udemy is the pricing methods.
```

```
when trying to obtain any information or indeed obtaining their grievance procedure.
They are not prepared to give names and simply fob you off.I had a very serious matt
er to deal with that could have been sorted had they bothered to help me. Staff from
this Company are Handling the Administration of Tonik Energy and after 2 Months plu
s, have Still NOT transferred My Credit from Tonik to Scottish Power (forced onto by
OFGEM) !!! Great study on private equity valuations. We were working together on tax
matter. They are very reliable and updated.

Summary for General Assembly:
{'f': 0.999999995, 'p': 1.0, 'r': 1.0}
I taught the full-stack web-development course.I'd always thought that bootcamps wer
e a nice idea for getting a very basic level of competence, but were pretty pricey f
or the privilege for what you could get from an online course.So I was pleasantly su
rprised at how effective this teaching style was for ramping up students from basica
lly knowing nothing about coding to being a hireable junior developer.I would have h
ired my students I wasn't a student, I was a teacher here.
```

In [62]:

```
# This cell is just for exploration
word_idf = {}
feedback_combined = ' '.join(str(body) for body in df_companies['df_Avado']['reviewB
reference = ' '.join(str(body).replace('\n','') for body in df_companies['df_Avado']
sentences = sent_tokenize(reference)
for word in word_tokenize(feedback_combined):
    if word in word_idf.keys():
        continue
    contained_sentence = 0
    for sent in sentences:
        if word in sent.lower():
            contained_sentence += 1
    if contained_sentence == 0: contained_sentence += 1 # smoothing, to avoid zero d
    word_idf[word] = np.log(len(sentences)/contained_sentence)

print('First 5 IDF:')
n = 5
for key, value in word_idf.items():
    print(key, value)
    if n==0: break
    n-=1

sentence_scores = {}

for sent in sentences:
    sent_cleaned = " ".join(re.findall("[a-zA-Z]+", sent)).lower()
    sent_cleaned = " ".join([i for i in sent_cleaned.split() if i not in stopwords])
    sentence_scores[sent] = calculate_sentence_score(sent_cleaned, word_idf, False)

print('\nTop high scored sentences:')
sorted_sentences = dict(sorted(sentence_scores.items(), key=lambda item: item[1], re
n = 17
for key, value in sorted_sentences.items():
    print(key, value)
    if n==0: break
    n-=1
```

```
First 5 IDF:
speaking 6.078892503982828
one 2.0294819142399856
advisors 5.586416018885034
phone 3.4863551900024623
felt 4.564764771353053
answers 5.075590395119043

Top high scored sentences:
"If you're going to honour the twelve months free membership that you have promised,
tell me how you plan to reimburse me for the four months from July to October 2019 t
hat you offered but hasn't been included!Now lets move on to the HR Inform three mon
```

```
th free subscription - they didn't bother to add the essential caveat that this is i
n fact not free at all - if you want the three free months, you have to pay for a 12
month subscription and you will then get 3 of those months for free.Now lets move on
to the Avado learning platform. 10.461956470644427
Please, please be very careful. 9.453991220881743
There were spelling mistakes and other mistakes frequently. 9.148079485345031
I turned up to another exam and the exam had not been booked with the exam center, l
uckily they had a space and let me sit my exam. 8.97765175405021
I particularly had challenges with my local bank because of our local restrictions.
8.92006721291753
You sign in with your username and there is a tickbox option to remember your userna
me. 8.890271365600896
or it’s just a hoax. 8.4767877767812
But was mistaken again. 8.4767877767812
It is glitchy and temperamental. 8.4767877767812
Bravo! 8.4767877767812
Fast enrolement, friendly and knowledgeable enrolement personel. 8.345991235370931
They sell sell sell and say everything you want to hear, makes everything sound wond
erful.... the truth is- they do not have 1-2-1 support for you. 8.2346093057549
They are enthusiastic, not at all patronizing or scary. 8.110583680558495
It also enables you to be able to connect with people who you otherwise wouldn't be
able to connect with. 8.077774986391619
Most serious of all they are dodgy with the financial arrangement, I am in the proce
ss of making a complaint to the financial conduct authority. 7.78458967919782
Seemless! 7.783640596221253
To investigate. 7.783640596221253
TUTOR WAS RUDE - The tutor would say rude and dismissive comments and would favor ce
rtain students particular students of a certain gender. 7.779217638386946
```

2.3. TF-IDF from Each Company and Sentiment (no lemmatization)

In [54]:

```
companies = list(set(df['itemReviewed_name']))

for company in companies:
    print('Summary for {0} - good feedback:'.format(company))
    df_name_good = 'df_{0}_good'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name_good]['r
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    summary = tf_idf_summarisation(feedback_combined, sentences, 7, False, None)
    rouge = Rouge()
    if len(reference) > 0:
        score_rouge = rouge.get_scores(summary, reference)
        print(score_rouge[0]['rouge-1'])
    print(summary, '\n')

    print('Summary for {0} - bad feedback:'.format(company))
    df_name_bad = 'df_{0}_bad'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name_bad]['re
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    summary = tf_idf_summarisation(feedback_combined, sentences, 7, False, None)
    rouge = Rouge()
    if len(reference) > 0:
        score_rouge = rouge.get_scores(summary, reference)
        print(score_rouge[0]['rouge-1'])
    print(summary, '\n')
```

```
Summary for BPP - good feedback:


Summary for BPP - bad feedback:
{'f': 0.26261682014857635, 'p': 1.0, 'r': 0.1511565357719204}
I then rang customer service who said that I had emailed the wrong email address and
should have emailed the first address which is why it never got booked, even though
I had emailed that address in the first place.There has been another occasion where
```

```
Summary for Deloitte - good feedback:
{'f': 0.999999995, 'p': 1.0, 'r': 1.0}
We were working together on tax matter. Great study on private equity valuations. Th
ey are very reliable and updated. I've had the opportunity to work with them. It was
a great experience. They are reliable and professional.

Summary for Deloitte - bad feedback:
{'f': 0.9893238384164336, 'p': 0.9858156028368794, 'r': 0.9928571428571429}
!They won't Answer any of My E-mails, just Tell Me to Ask Scottish Power, who have N
O Contact details for the Administrators other then "help" at "tonikenergy.com"The A
dministrators need to Transfer Customers Credits to Scottish Power NOW, People need
that Credit to Pay for Winter Energy Usage !!!! Absolutely appalling experience when
trying to obtain any information or indeed obtaining their grievance procedure.They
are not prepared to give names and simply fob you off.I had a very serious matter to
deal with that could have been sorted had they bothered to help me. The most dishone
st unprofessional company I've ever been involved with Very bad company Staff from t
his Company are Handling the Administration of Tonik Energy and after 2 Months plus,
have Still NOT transferred My Credit from Tonik to Scottish Power (forced onto by OF
GEM) !!!

Summary for General Assembly - good feedback:
{'f': 0.999999995, 'p': 1.0, 'r': 1.0}
I taught the full-stack web-development course.I'd always thought that bootcamps wer
e a nice idea for getting a very basic level of competence, but were pretty pricey f
or the privilege for what you could get from an online course.So I was pleasantly su
rprised at how effective this teaching style was for ramping up students from basica
lly knowing nothing about coding to being a hireable junior developer.I would have h
ired my students I wasn't a student, I was a teacher here.

Summary for General Assembly - bad feedback:
```

2.4. TF-IDF from Whole Dataset by Sentiment (with lemmatization)

In [55]:

```
good_feedback_combined = ' '.join(str(body) for body in df_good['lemmatized_reviewBo
reference = ' '.join(str(body).replace('\n','') for body in df_good['reviewBody'].to
sentences = sent_tokenize(reference)
lemmatized_sentences = sent_tokenize(' '.join(str(body).replace('\n','') for body in
summary = tf_idf_summarisation(good_feedback_combined, sentences, 7, True, lemmatize

# recall: how much the words in the reference appeared in the summaries
# precision: how much the words in the summary appeared in the reference
# f1 score: 2*(precision*recall)/(precision+recall)

rouge = Rouge()
score_rouge = rouge.get_scores(summary, reference)

print(score_rouge[0]['rouge-1'])
print(summary)
```

```
{'f': 0.0008119499436721188, 'p': 0.9957264957264957, 'r': 0.0004061405664702201 6}
The Kafka module covers the majority of components/tools used in Kafka ecosystem, th
e Teacher has a very good knowledge and excellent slides complemented by code exampl
es to explain how Kafka works.The project gave me an excellent overview about kafka
and related components of kafka ecosystem. My favorite course was Seth Godin's "Seth
Godin's Freelancer Course." !Leaders inspire other leaders to make more leaders tha
t’s the only way! It was a wonderful education journey…It was a wonderful education
journey that will satisfy my needs and meet all my expectations,In Digital Marketing
Nanodegree they will teach all points and have a great practicing with live monitori
ng for all the sectionsThank you Udacity It was a wonderful education journey…It was
a wonderful education journey that will satisfy my needs and meet all my expectation
s,In Digital Marketing Nanodegree they will teach all points and have a great practi
cing with live monitoring for all the sectionsThank you Udacity I jumped into the na
nodegree course not knowing what to expect. So win win. Kylee and Prajwala work so w
```

```
ell together and Kylee misses Prajwala days they don't work together. The only criti
cism I have is that on the Dashboard, when you click on a particular Playlist, inste
ad of taking you to that particular Playlist, it takes you to a list of all your Pla
ylists and you have to find the one you are looking for from that list.
```

In [57]:
```
bad_feedback_combined = ' '.join(str(body) for body in df_bad['lemmatized_reviewBody
reference = ' '.join(str(body).replace('\n','') for body in df_bad['reviewBody'].to_
sentences = sent_tokenize(reference)
lemmatized_sentences = sent_tokenize(' '.join(str(body).replace('\n','') for body in
summary = tf_idf_summarisation(bad_feedback_combined, sentences, 7, True, lemmatized

# recall: how much the words in the reference appeared in the summaries
# precision: how much the words in the summary appeared in the reference
# f1 score: 2*(precision*recall)/(precision+recall)

rouge = Rouge()
score_rouge = rouge.get_scores(summary, reference)

print(score_rouge[0]['rouge-1'])
print(summary)
```

```
{'f': 0.0017555641160730056, 'p': 1.0, 'r': 0.000878553245089893}
if instead of writing < i glyphicon glypohicon-world > < / i > you write  < i  glyph
icon glypohicon-world / > In my instance I only got 5 distinct instances of the icon
instead of the expected 1.
```

2.5. TF-IDF from Each Company (with lemmatization)

In [58]:
```
companies = list(set(df['itemReviewed_name']))

for company in companies:
    print('Summary for {0}:'.format(company))
    df_name = 'df_{0}'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name]['lemmat
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    lemmatized_sentences = sent_tokenize(' '.join(str(body).replace('\n','') for bod
    summary = tf_idf_summarisation(feedback_combined, sentences, 7, True, lemmatized
    # recall: how much the words in the reference appeared in the summaries
    # precision: how much the words in the summary appeared in the reference
    # f1 score: 2*(precision*recall)/(precision+recall)

    rouge = Rouge()
    score_rouge = rouge.get_scores(summary, reference)

    print(score_rouge[0]['rouge-1'])
    print(summary, '\n')
```

```
Summary for BPP:
{'f': 0.26261682014857635, 'p': 1.0, 'r': 0.1511565357719204}
I then rang customer service who said that I had emailed the wrong email address and
should have emailed the first address which is why it never got booked, even though
I had emailed that address in the first place.There has been another occasion where
i emailed in and didn't get a response after 2 weeks.I have had a number of calls to
customer service where they have been rude and unhelpful, and in one case I was told
an issue would be rectified, I rang again a few days later (a lovely lady answered)
and she informed me that the previous call handler didn't even log an issue in the f
irst place.I have had issues with access to my hub account, which have taken 2 weeks
to sort, which meant i was 2 weeks behind on my course as i had no access to learnin
g materials; however they refused to move me to a course which had a later start dat
e, so I have been unable to qualify for pass assurance.On a number of times I have t
elephoned due to an urgent matter i have been told to 'email in with urgent as the r
eference and it will get sorted today' and then when I finally have a response two d
ays later, I am told I should have rang in instead of emailed because they cannot re
spond to urgent emails quick enough.I have been emailed the incorrect dates for my c
urrent revision course, which I am now unable to attend one of the days because of t
his.They are extremely slow in marking the 'Step 4 and 6' which are meant to aid you
```

2.6. TF-IDF from Each Company and Sentiment (with lemmatization)

In [59]:
```
companies = list(set(df['itemReviewed_name']))

for company in companies:
    print('Summary for {0} - good feedback:'.format(company))
    df_name_good = 'df_{0}_good'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name_good]['l
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    lemmatized_sentences = sent_tokenize(' '.join(str(body).replace('\n','') for bod
    summary = tf_idf_summarisation(feedback_combined, sentences, 7, True, lemmatized
    rouge = Rouge()
    if len(reference) > 0:
        score_rouge = rouge.get_scores(summary, reference)
        print(score_rouge[0]['rouge-1'])
    print(summary, '\n')

    print('Summary for {0} - bad feedback:'.format(company))
    df_name_bad = 'df_{0}_bad'.format(company)
    feedback_combined = ' '.join(str(body) for body in df_companies[df_name_bad]['le
    reference = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    sentences = sent_tokenize(reference)
    lemmatized_sentences = sent_tokenize(' '.join(str(body).replace('\n','') for bod
    summary = tf_idf_summarisation(feedback_combined, sentences, 7, True, lemmatized
    rouge = Rouge()
    if len(reference) > 0:
        score_rouge = rouge.get_scores(summary, reference)
        print(score_rouge[0]['rouge-1'])
    print(summary, '\n')
```

```
Summary for BPP - good feedback:

Summary for BPP - bad feedback:
{'f': 0.26261682014857635, 'p': 1.0, 'r': 0.1511565357719204}
I then rang customer service who said that I had emailed the wrong email address and
should have emailed the first address which is why it never got booked, even though
I had emailed that address in the first place.There has been another occasion where
i emailed in and didn't get a response after 2 weeks.I have had a number of calls to
customer service where they have been rude and unhelpful, and in one case I was told
an issue would be rectified, I rang again a few days later (a lovely lady answered)
and she informed me that the previous call handler didn't even log an issue in the f
irst place.I have had issues with access to my hub account, which have taken 2 weeks
to sort, which meant i was 2 weeks behind on my course as i had no access to learnin
g materials; however they refused to move me to a course which had a later start dat
e, so I have been unable to qualify for pass assurance.On a number of times I have t
elephoned due to an urgent matter i have been told to 'email in with urgent as the r
eference and it will get sorted today' and then when I finally have a response two d
ays later, I am told I should have rang in instead of emailed because they cannot re
spond to urgent emails quick enough.I have been emailed the incorrect dates for my c
urrent revision course, which I am now unable to attend one of the days because of t
his.They are extremely slow in marking the 'Step 4 and 6' which are meant to aid you
r learning and give feedback to help you improve your written answers. The idea is t
hat you have feedback from Step 4 before you do Step 6, but I have even waited until
after the Step 6 deadline and my Step 4 still hadn't been marked.Overall, I wish it
would be simple enough to change apprenticeship providers because trying to deal wit
h customer service at BPP is extremely stressful and makes my whole experience compl
etely unenjoyable. Avoid, avoid, avoid!They are grossly incompetent and their Custom
er Service is something I have not come across before. I had emailed in to book to t
he correct email address, where they informed me that I should email a different add
ress. Good for masochists who like to be belittled and treated like scum by peopole
who think they are special, but not so good for people wanting a professional qualif
ication delivered by professionals. Happy to take your money but not happy to fulfil
l their end of the bargain I thankfully only bought a book not an exam or an apprent
```

```
I taught the full-stack web-development course.I'd always thought that bootcamps wer
e a nice idea for getting a very basic level of competence, but were pretty pricey f
or the privilege for what you could get from an online course.So I was pleasantly su
rprised at how effective this teaching style was for ramping up students from basica
lly knowing nothing about coding to being a hireable junior developer.I would have h
ired my students I wasn't a student, I was a teacher here.

Summary for General Assembly - bad feedback:
```

3. BERT Extractive Summarisation Library

Can only handle text less than 1000000 length (number of characters).

In [31]:
```
def bert_summarisation(sentences):
    if len(sentences) > 1000000:
        return 'Feedback length exceeds 1000000 number of characters. It is going to
    model = Summarizer()
    result = model(sentences, min_length=30, num_sentences=7)
    summary = ''.join(result)
    return summary
```

3.1. BERT from Whole Dataset by Sentiment

In [22]:
```
sentences = ' '.join(str(body).replace('\n','') for body in df_good['reviewBody'].to
bert_summarisation(sentences)
```

Out[22]:
```
'Feedback length exceeds 1000000 number of characters. It is going to cause memory a
llocation error.'
```

In [23]:
```
sentences = ' '.join(str(body).replace('\n','') for body in df_bad['reviewBody'].to_
bert_summarisation(sentences)
```

Out[23]:
```
'Feedback length exceeds 1000000 number of characters. It is going to cause memory a
llocation error.'
```

3.2. BERT from Each Company

In [24]:
```
companies = list(set(df['itemReviewed_name']))

for company in companies:
    print('Summary for {0}:'.format(company))
    df_name = 'df_{0}'.format(company)
    sentences = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    summary = bert_summarisation(sentences)
    print(summary, '\n')
```

```
Summary for General Assembly:
I wasn't a student, I was a teacher here. I taught the full-stack web-development co
urse. I'd always thought that bootcamps were a nice idea for getting a very basic le
vel of competence, but were pretty pricey for the privilege for what you could get f
rom an online course. So I was pleasantly surprised at how effective this teaching s
tyle was for ramping up students from basically knowing nothing about coding to bein
g a hireable junior developer.

Summary for Udacity:
it's excellent and better than other sites I experienced.specially great support fro
m tutors.but for being better:1. I will nominate Udasti to anyone who need to learns
marketing it going pretty good, decent information loved how it was presented and th
e support from the team Udacity Android Nanodegree is GoodThere are some things that
are confusing when doing the project, like things not specified or vaguely defined t
hat make you think the project needs things that weren't taught. Altogether I am ver
y happy with my decision to take the course. By the end of the course I'll have paid
a lot more, because everything I pay in AWS is USD and my currency is extremely unde
```

```
ging you. Nowadays, the business model seems to have changed and payment/subscriptio
n is required (what a pit), but that didn't seem to hurt the quality of the courses.
But their platform didn't get any better, in fact the site gets only worse. Informat
ion must be free!I'll pay if I want to pass tests Nothing new, same issues, terrible
customer service, no phone number, no email, hard to navigate the site to be even ab
le to contact them. This company is solely out for money. The feedback is always hel
pful, of course, but I expected the instructor to give at least a portion of the fin
al grade.

Summary for Deloitte:
Staff from this Company are Handling the Administration of Tonik Energy and after 2
Months plus, have Still NOT transferred My Credit from Tonik to Scottish Power (forc
ed onto by OFGEM) !!!! They won't Answer any of My E-mails, just Tell Me to Ask Scot
tish Power, who have NO Contact details for the Administrators other then "help" at
"tonikenergy.com"The Administrators need to Transfer Customers Credits to Scottish P
ower NOW, People need that Credit to Pay for Winter Energy Usage !!!! They are very
reliable and updated. I've had the opportunity to work with them. Great study on pri
vate equity valuations. Absolutely appalling experience when trying to obtain any in
formation or indeed obtaining their grievance procedure. I had a very serious matter
to deal with that could have been sorted had they bothered to help me.
```

3.3. BERT from Each Company and Sentiment

In [32]:

```
companies = list(set(df['itemReviewed_name']))

for company in companies:
    print('Summary for {0} - good feedback:'.format(company))
    df_name_good = 'df_{0}_good'.format(company)
    sentences = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    summary = bert_summarisation(sentences)
    print(summary, '\n')

    print('Summary for {0} - bad feedback:'.format(company))
    df_name_bad = 'df_{0}_bad'.format(company)
    sentences = ' '.join(str(body).replace('\n','') for body in df_companies[df_name
    summary = bert_summarisation(sentences)
    print(summary, '\n')
```

```
Summary for DPG plc - good feedback:
I waited 4 months to review DPG so that it can be more insightful. Hi there, I will
leave a short review due the fact that I just enrolled on the CIPD course, but so fa
r the staff from DPG were most helpful and prompt in my requests by phone or by emai
l. I was very close to signing up with a different provider before discovered DPG, t
he reason why I changed my opinion in the last minutes is because customer service i
s a big thing for me. I have really enjoyed my onboarding experience with DPG.The si
gnup process was quick and easy. All my questions have been answered quickly in a pr
ofessional way I am due to commence my Level 5 HRM Diploma on Monday 3rd February so
have yet to comment on the learning provider itself as of yet, but I must say that t
he Customer Service has been absolutely exceptional. Excellent customer service and
support. thank you so much i love dealing with you Friendly staff and advice, an eas
y process to join the course. Quick to respond to questions and very informative.

Summary for DPG plc - bad feedback:
I purchased the level 5 diploma and after I completed the course, I was issued with
the certificate. Not once was it marked as a priority despite them saying it would b
e. The lack of customer service and response is appalling! I found it difficult to g
et a call bac and speak with an advisor I booked my course with DPG as I heard they
were the best, although be careful when they say they are fully flexible.... I enqui
red about workshop and exam days prior to booking the course and paying my huge cost
s as I knew that I would be getting married. The layout is not what you would expect
from an online training provider. At no point in time did any of the two consults ad
vise that I could loose my place if the paper work was not handed in. This is a prog
ramme that cost well over £4000 and that's how you get treated for the money you are
paying. I would still like to attend the March group and feel that I should not have
to suffer due to the poor service of the consultants representing your company.
```