Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Twitter as an Alternative Review Site

Sinéad Dickson

14310749

April 2019

A Dissertation submitted in partial fulfilment
of the requirements for the degree of
MAI (Computer and Electronic Engineering)

Supervised by Professor Séamus Lawless and Anirban Chakraborty

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: _____ Date: _____

Summary

This dissertation proposes a method of extracting review-like tweets from Twitter and then incorporating the sentiment of those tweets into a recommender system. This research aims to evaluate to what extent Twitter can provide a suitable source of online reviews that can be used effectively in the generation of recommendations in a recommender system. The project focused on reviews about hotels in Dublin.

Online reviews have become an important source of consumer information. They provide vast amounts of valuable information on consumer preferences. Twitter is a huge microblogging social media site with 126 million users and 500 million tweets posted daily. Twitter users regularly share their opinions on a wide range of different topics. Often these tweets can take the form of a review. This project aims to utilise these review-like tweets that are posted to Twitter.

Previous literature has shown that supervised machine learning methods can be successfully used to classify tweets. Numerous studies have focused particularly on classifying the sentiment of tweets. However, there are few papers which have looked at classifying tweets as reviews, which is what this research aims to do. One of the objectives of this research is to investigate if it is possible to classify whether a tweet contains content that could be deemed to be a review.

The data was collected through the Twitter Streaming API (application programming interface). It then had to be filtered so that it only contained tweets about hotels posted from Dublin. Once the dataset was filtered it had to be manually annotated so that it could be used to train a classification algorithm. A webpage was built to facilitate this annotation process. The tweets were labelled as 'Review', 'Some Content' or 'Irrelevant'.

Thirteen different classifiers were evaluated. These included: the Decision Tree Classifier, the Random Forest Classifier, the Multi Layer Perceptron Classifier, the Support Vector Machine Classifier, the Logistic Regression Classifier, the K Nearest Neighbours Classifier, the Gaussian Process Classifier, the AdaBoost Classifier, the Gaussian Naive Bayes Classifier, the Bernoulli Naive Bayes Classifier, the Multinomial Naive Bayes Classifier, the Quadratic Discriminant Analysis Classifier and the Linear Discriminant Analysis Classifier. Seven

feature representations were experimented with, unigram bag-of-words, unigram, bigram and trigram TFIDF, unigram TFIDF with stop words removed, Word2Vec and Doc2Vec. The classifiers were implemented using Python's Scikit Learn library.

The results show that the Support Vector Machine Classifier performed best at classifying the tweets that contained reviews. It achieved a precision score of 74%, a recall score of 74%, an f1-score of 73% and an accuracy score of 74.4%. This showed that text classification is a valid method of extracting reviews from Twitter.

The next part of the project involved using the tweets, now classified as reviews, to help generate recommendations in a recommender system. The increasing amount of information available on the internet has meant that recommender systems have grown in importance. They are required to help users quickly find relevant information and to overcome the information overload problem. Hotel recommender systems such as Expedia, TripAdvisor and Booking.com are widely used by consumers.

It was shown in previous research that incorporating reviews from TripAdvisor into a hotel recommender system increased the average user satisfaction. In this project, reviews from Twitter were incorporated into the CoRE recommender system, in the hope that it would perform equally well.

The CoRE recommender system is a Cold Start Resistant and Extensible Recommender system. It makes use of collaborative filtering, content-based recommendation and contextual suggestion. The algorithm is designed to work in cold start situations and to be easily extensible.

Sentiment analysis was carried out on the tweets that were classified as reviews, using the Stanford NLP Sentiment Analyser. The sentiment scores produced were used to re-rank the results of the CoRE Recommender System and produce SentiCoRE.

The results show that incorporating the sentiment score had the desired effect and adjusted the rankings of the hotels. However, in terms of mean percentile rank (MPR) SentiCoRE performed worse that CoRE.

Abstract

Twitter as an Alternative Review Site. By Sinéad Dickson, Trinity College Dublin 2019.

Supervised by Professor Séamus Lawless and Anirban Chakraborty.

The increasing amount of information available on the internet means that recommender systems are growing in importance. Recommender systems help users to overcome information overload. They aim to help users to quickly find information relevant to them.

Online reviews are an important source of consumer opinions. They provide large quantities of data on consumer preferences and opinions. Twitter is a widely used microblogging social media platform, with 126 million daily users and 500 million tweets posted per day. Tweets posted to the site can often take the form of a review. This project will focus on harnessing these reviews and using them to help generate recommendations in the CoRE recommender system.

This dissertation proposes a method of first extracting review-like tweets from Twitter and then incorporating the sentiment of those tweets into a recommender system. This research aims to evaluate to what extent Twitter can provide a suitable source of online reviews that can be used effectively in the generation of recommendations in a recommender system. The project focused on reviews about hotels in Dublin.

This research explores the use of various classification algorithms and feature representations to classify whether tweets contain reviews. The sentiment of these review-like tweets is calculated and used to re-rank the CoRE recommender system.

The classification results were promising, showing that text classification is a valid method of extracting tweets from reviews. The best performing classifier was the Support Vector Machine. It achieved a precision score of 74%, a recall score of 74%, an f1-score of 73% and an accuracy score of 74.4%.

Incorporating the sentiment score into the CoRE had the desired effect and adjusted the rankings of the hotels. However, in terms of mean percentile rank (MPR) SentiCoRE performed worse that CoRE.

Acknowledgements

I would first like to thank my project supervisors, Séamus Lawless and Anirban Chakraborty, for all their help and support over the course of this project.

I would also like to thank Matthew Nicholson and Mostafa Bayomi for there help with the CoRE recommender system.

Finally, I would like to thank my family and friends for their support and encouragement.

Contents

List of Figures

List of Tables

Nomenclature

AB	AdaBoost
API	Application Programming Interface
BNB	Bernoulli Naive Bayes
DT	Decision Tree
GNB	Gaussian Naive Bayes
GP	Gaussian Process
KNN	K Nearest Neighbour
LDA	Linear Discriminant Analysis
LR	Logistic Regression
mDAU	Monetizable Daily Active User
ME	Maximum Entropy
MLP	Multi Layer Perceptron
MNB	Multinomial Naive Bayes
MPR	Mean Percentile Rank
NB	Naive Bayes
NLP	Natural Language Processing
POS	Part of Speech
QDA	Quadratic Discriminant Analysis
RF	Random Forest
RNTN	Recursive Neural Tensor Network
SVM	Support Vector Machine
TF-IDF	Term Frequency Inverse Document Frequency

1 Introduction

1.1 Twitter as a Source of Reviews

The rapid development and expansion of the internet has introduced new ways for individuals to express their opinions and disseminate their views. Online reviews have become a hugely important source of information for consumers. They play an important role in determining whether a person is satisfied with a product or service. Online reviews provide a huge amount of data on consumer preferences. It is now relatively rare that someone will purchase a product, reserve a table at a restaurant or book a room in a hotel without first checking some online reviews. These reviews inform and influence consumer decisions, and have a direct relationship with online sales.

This project will focus on Twitter, a microblogging social media site, where users can post short blocks of text of no more than 280 characters. Twitter currently has 126 million people who use the site daily, which the company terms 'monetizable Daily Active Users' (mDAU), with over 500 million tweets posted per day (4). These tweets can often take the form of a review (Figure 1.1).

Twitter users post tweets which express their ideas and opinions on a wide array of topics. An individual may, for example, tweet about a city they visited, a hotel they stayed in, a restaurant they ate at or a film they watched. These review-like tweets can give insight into consumers' opinions on the entities with which they interact. With millions of tweets being posted every day, Twitter has a huge potential source of underutilised reviews.

Traditional online review sites include websites like TripAdvisor, Foursquare and Yelp. Often these sites have a dedicated area for feedback and encourage users to leave reviews of their hotels, restaurants or products. In our opinion, this method of obtaining reviews can sometimes result in more forced, manufactured, less reliable reviews. It is our contention that users tend to post their genuine feelings more spontaneously and frequently to Twitter. Users often wouldn't consider their posts to be a 'review' in the formal sense of the word and not in the same way they would consider a review left in the dedicated feedback area of another website.

Figure 1.1: Examples of Review-like Tweets.

The language used in tweets is different to longer form text and needs to be treated differently. Tweets are usually very informal, using casual language and slang. They contain things like hashtags, emojis, twitter handles, URLs, images, videos and gifs.

1.2 Research Question

This research project will investigate how useful Twitter can be when used as an alternative or additional source of reviews for use in a recommender system. We will explore methods of classifying review-like tweets, identifying the sentiment of the reviews and the effect of using this data in a recommender system.

The following research question will be addressed:

To what extent can Twitter provide a suitable source of online reviews that can be used effectively in the generation of recommendations in a recommender system?

1.3 Research Objectives

The project will focus on tweets that mention or discuss hotels in the Dublin area. A collection of 2.5 million tweets posted from Dublin, collected between October 2017 and September 2018, will be used as the dataset. The main objective of the project is to

determine whether tweets can be used successfully as a source of reviews in a recommender system.

The objectives of this research project are:

1. **Data Collection**
   The Twitter Streaming API [1] was used between October 2017 and September 2018 to collect a set of 2.5 million tweets. The API allows a bounding box to be specified. A bounding box was set for Dublin so all tweets collected were posted from within Dublin's bounding box.

2. **Data Processing**
   The raw dataset was cleaned and filtered, so that it only contained English language tweets posted from Dublin, and secondly so that it only contained tweets mentioning hotels.

3. **Dataset Annotation**
   This part of the project involved building an annotated dataset of tweets. The tweets from the processed dataset were manually labelled as either: review-like tweets; tweets that contain some contextual information related to hotels; or irrelevant tweets. A webpage was built so that participants could annotate the tweets.

4. **Tweet Classification**
   The annotated set of tweets was used to train multiple classifiers. Various supervised machine learning algorithms and feature representations were explored.

5. **Sentiment Analysis**
   The Stanford NLP Sentiment Analyzer (1) was used to classify the sentiment of the classified tweets. A sentiment score was generated for each hotel.

6. **Recommender System**
   The sentiment score produced was applied to the CoRE recommender system (5). The effect of adding the sentiment score was analysed and evaluated.

1.4 Report Structure

The dissertation will be structured as follows: Chapter 2 presents a review of the literature relating to this project. This includes studies on classification, particularly text classification, sentiment analysis, and recommender systems. Chapter 3 describes the design and methodology of the project. It outlines how the data was collected, filtered, processed, and annotated. It also describes the classification techniques used, the sentiment analysis tool

[1] `https://developer.twitter.com`

used and how the sentiment scores produced were applied to the recommender system CoRE (5). Chapter 4 presents an evaluation and discussion of the results of this research. Finally, Chapter 5 gives our conclusions and future works.

2 Literature Review

This section will discuss the background literature relating to this research project. This will include an introduction to supervised machine learning and classification with a particular focus on text classification. A significant amount of research has been carried out about tweet classification, particularly classifying the sentiment of tweets. The techniques used for tweet classification and sentiment classification will be explored. Finally, some of the current approaches to hotel recommendation will be presented.

2.1 Classification

Classification, in the context of machine learning, is the process of mapping observations into classes, based on some set of training data. There are two main approaches to classification, supervised and unsupervised learning.

Supervised machine learning (6) involves labelled data. There are input variables (x) and output variables (y) and an algorithm is used to train the mapping function from the input to the output (y = f(x)). The mapping function that has been trained is then applied to new unseen data to decide it's class. Supervised machine learning algorithms include Support Vector Machines (7, 8), Naive Bayes Classifiers (9), Random Forest Classifiers (10), Decision Tree Classifiers (11), Logistic Regression Classifiers (12) and Nearest Neighbour Classifiers (13).

Unsupervised machine learning uses unlabelled data. It only has input variables (x) with no output variables. The learning algorithm is used to identify patterns directly from the data. There are no predefined classes. Unsupervised learning is used to discover unknown patterns in data. Unsupervised machine learning algorithms include K-Means Clustering (14) and Gaussian Mixture Models (15).

A supervised machine learning approach will be taken in this research. A collection of tweets will be manually annotated and assigned one of the following three labels: Review, Some Content, or Irrelevant. These annotated tweets will be used to train a text classifier.

2.2 Text Classification

Text classification is an application of both supervised and unsupervised machine learning (16). Classification algorithms can be applied to a wide range of different areas, which include text, speech, images and video. In this research, we are attempting to classify the text found in tweets. Text classification involves automatically assigning a set of classes to raw text documents. Applications include sentiment analysis, spam detection, language detection and topic classification. Certain classification algorithms are more suited to classifying text. This section will discuss the functionality of some classification algorithms that are suited to text classification.

A Support Vector Machine (SVM) is a discriminative classifier, first introduced by Cortes and Vapnik (7, 8). SVMs have been successfully used for text classification (17, 18). The algorithm functions by finding the optimum hyperplane that separates the data into the defined classes. The aim of the SVM is to maximise the distance between the hyperplane and the support vectors (the data points closest to the hyperplane).

Another popular supervised learning method often used for text classification is the Naive Bayes (NB) Classifier (9). NB is a generative classifier, learning a model of the joint probability P(A,B), and making predictions by using Bayes Rule. It calculates the probability that an observation belongs to a particular class. NB is based on applying Bayes Rule along with the 'naive' assumption that features are conditionally independent. Bayes Rule is as follows:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{1}$$

In the case of text classification, this means we assume all words are independent, which is of course not the case. This assumption is a major pitfall of the algorithm.

Logistic Regression (LR) is the discriminative counterpart to Naive Bayes (19). It is a linear classifier. LR uses the logistic function, also known as the sigmoid function or logit function to model the data. This is an S-shaped curve, taking real-valued inputs and mapping them to the range $0 - 1$. The logistic function is as follows:

$$g(z) = \frac{1}{(1 + e^{-z})} \tag{2}$$

Logistic Regression models the probability that an observation belongs to a particular class. The coefficients of the LR algorithm are estimated from the training data, using maximum likelihood estimation or gradient descent.

The Maximum Entropy (ME) classifier is another discriminative classifier. It has been explored as a successful method of text classification (20). The ME classifier is based on

the principle of maximum entropy. The principle of maximum entropy is that the probability distribution that best represents the current state of knowledge is the one with the maximum entropy. Entropy is a measure of the disorder or randomness of a system or a measure of lack of knowledge. The maximum entropy corresponds to the least amount of knowledge, which is when the data is as uniformly distributed as possible. The Maximum Entropy classifier seeks to find the distribution that maximises the entropy. Maximum Entropy is similar to Naive Bayes but has an advantage in that it does not suffer from the independence assumption.

Ensemble Classifiers, combine the effect of multiple learning algorithms to try and achieve better performance than the individual learning algorithms (21). They aggregate various individual base classifiers. There are two major ensemble methods, averaging ensemble methods and boosting ensemble methods. Averaging methods include Bagging and Forests of Randomised Trees. These methods output an average of the base estimators. Boosting methods include AdaBoost and Gradient Tree Boosting. They give an ensemble output which is the sequential effect of the base classifiers. Ensemble classifiers generally perform better than individual base classifiers (22).

A Random Forest Classifier is an ensemble learning method based on the Decision Tree Classifier. The original algorithm proposed by Breinam (10), combines several tree-structured classifiers using bagging (23). A number of Decision Tree Classifiers are fitted, using a random selection of training data, and their results are merged to get a prediction. Random Forest Classifiers improve on Decision Tree over-fitting and give a more accurate and stable prediction. They have been successfully used in many text classification applications, for example, the classification of spam emails (24).

2.3 Tweet Classification

In this section, text classification techniques that have been applied to Twitter data will be discussed, along with the challenges faced in implementing them. All off-the-shelf classification algorithms must be adapted to the domain on which they will be applied in order to achieve optimum performance. A classifier applied to Twitter data is no exception. Each classification algorithm must be fine-tuned in order to achieve the best possible performance.

Twitter data has a different format to standard long-form text and needs to be treated differently. Tweets are short with a maximum of 280 characters. This has led to the use of particular characteristic features. Tweets are generally very informal, using casual language and slang. They contain features like hashtags, emojis, Twitter handles, URLs, images, videos and gifs, which don't occur in standard text. The short length and non-standard

features can create a challenge for standard text classification algorithms and standard machine learning document representations.

In 2017, Twitter upped the character limit from 140 characters to 280 characters. This increase has the potential to change the characteristics of tweets. The longer length could perhaps mean users have less of a need to be so brief in their tweets. This could lead to the use of fewer abbreviations and less slang. Many of the papers referenced in this section were written before this character limit increase came in. This needs to be taken into consideration when comparing the results of these studies to our results and to studies completed after Twitter's change in character count.

A. Bermingham and A. F. Smeaton (25) investigated the performance of both Support Vector Machines (SVM) and Multinomial Naïve Bayes (MNB) in classifying the sentiment of short versus long form text documents. The short form documents analysed were tweets from Twitter and micro-reviews from Blippr. The long form documents were TREC Blog06 Corpus and Pang and Lees Movie Review Corpus (26). Maximum accuracy of 74.85% was achieved with MNB versus 73.45% with SVM, for the Twitter data. Overall MNB achieves better accuracy than SVM on the short form documents, from both Twitter and Blippr, suggesting that MNB may be useful for the classification of tweets as reviews in this research. Another point of interest from Bermingham and Smeaton's research is the kind of feature extraction that performed best for the long versus short form text documents. Extending the unigram feature representation improved classification accuracy for the long form documents, but not for the short form documents. POS (part-of-speech) features and punctuation aided classification of the short form documents.

An Ensemble Classifier was proposed by Ankit and N. Saleena (27) to classify the sentiment of tweets. The base classifiers include Naive Bayes, Random Forest, Support Vector Machine and Logistic Regression. Ankit and N. Saleena's weighted ensemble classifier outperforms each of the individual base classifiers, as well as the majority voting ensemble classifier. M. Kanakaraj and R. M. R. Guddeti (28) also found that ensemble methods performed better in classifying the sentiment of tweets than base classifiers. Several base classifiers and ensemble methods were compared on how well they performed in classifying the sentiment of tweets. The base classifiers included Support Vector Machine, Baseline, Maximum Entropy and Naive Bayes, and the ensemble methods included Extremely Randomised Trees, Random Forest, Adaboost, and Decision Tree. The ensemble methods again outperformed the individual base classifiers. The ensemble method that performed best was Extremely Randomised Trees.

A. Go, R. Bhayani, and L. Huang (29) address the problem of obtaining a large annotated dataset to train classifiers. They proposed the idea of creating a training set of tweets, labelled as positive or negative based on the emojis that the tweets contain. The dataset

produced, called Stanford Sentiment 140, was used to train Naive Bayes, Maximum Entropy, and Support Vector Machine classifiers. They report the highest accuracy (83%) with the Maximum Entropy Classifier. They showed how useful emojis can be in automatically annotating large quantities of tweets. Datasets of annotated tweets can be created quickly and easily compared to the time taken to manually annotate datasets, which can be a very time-consuming, costly and labour-intensive process. The Stanford Sentiment 140 dataset produced has been used in many other studies, including the previously mentioned Ensemble Classifier study (27).

B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas (30) classified tweets into a set of generic classes: News, Opinions, Events, Deals and Private Messages. They proposed an eight-feature technique. Eight features were extracted from the tweets, one nominal and seven binary. The nominal feature was the author. The seven binary features were, whether the tweet contained shortened words or slang, whether the tweet contained time-event phrases, whether the tweet contained opinion words, whether the tweet contained an emphasis on words, whether the tweet contained currency or percentage signs, whether the tweet username is at the start of the tweet and whether the username is mid-tweet. A Naive Bayes Classifier was used with 5-fold cross-validation. They compared the 8-feature technique to bag-of-words and found it performed significantly better. It seems BOW cannot be directly applied to short texts because BOW ignores the order and semantic relations between words.

M. Rathi, A. Malik, D. Varshney, R. Sharma, and S. Mendiratta (31) tested SVM, Adaboosted Decision Tree and Decision Tree Classifiers, for classifying the sentiment of tweets. Term Frequency - Inverse Document Frequency (TF-IDF) Vectorization is applied during pre-processing. Using TF-IDF gives a measure of how important a word is within the dataset. A word that is frequent in an individual document but infrequent in the dataset is considered important. The weights from TF-IDF are applied to the dataset emphasising the contribution of some words and reducing the contribution of others. They found the Decision Tree Classifier (84%) achieved the highest accuracy followed by the SVM (82%) and then the Adaboosted Classifier (67%).

A. Rane and A. Kumar (3) compare seven different classifiers for the sentiment analysis of Twitter reviews about US Airline Services. The classifiers were Decision Tree, Random Forest, SVM, K-Nearest Neighbours, Logistic Regression, Gaussian Naive Bayes and AdaBoost. Doc2Vec feature representation was used, which involves mapping each document to a vector in space. In this study, each paragraph was mapped to a vector. The Random Forest Classifier performed the best, with reported precision of 85.6%, recall of 86.5% and accuracy of 86.5% (Table 2.1).

Table 2.1: Precision, Accuracy and Recall of the classifiers from the study by A. Rane and A. Kumar (3).

Classifier	Precision	Recall	Accuracy
Decision Tree	63 %	64.6%	64.5%
Random Forest	**85.6%**	**86.5%**	**86.5%**
Support Vector Machine	81.2%	84.4%	84.8%
Gaussian Naive Bayes	64.2%	64.7%	64.6%
AdaBoost	84.5%	83.5%	83.5%
Logistic Regression	81%	81.6%	81.9%
K Nearest Neighbour	59%	59.2%	59.3%

2.4 Sentiment Analysis

Sentiment analysis is the process of identifying the opinion expressed about a particular subject in some text. The aim is to determine whether the opinion is positive, negative or neutral, and to what extent. Sentiment analysis makes use of natural language processing (NLP) and machine learning. This section will discuss the two main approaches to sentiment analysis, lexicon based approaches and supervised machine learning based approaches.

A lexicon-based approach works by classifying a sentence based on the number of opinion words (positive or negative words) in the sentence. A sentiment score is calculated based on the ratio of positive to negative words. Unlike supervised machine learning methods, no training data is required. However, a lexicon is required and these are not available for every language. The lexicon approach has limitations, it does not allow for a term to be positive in one context but negative in another. It also cannot deal with an opinion being expressed towards multiple entities in a single sentence.

Supervised machine learning approaches require labelled training data. Their performance is very dependent on the size and quality of the set of training data. The accuracy of a classifier depends on the features selected for training and the domain the classifier is applied to. A method that performs well on a set of reviews from TripAdvisor will not necessarily perform well on a set of Tweets. Each method needs to be adapted to the specific domain it will be used on.

S. Bhuta, A. Doshi, U. Doshi, and M. Narvekar (32) reviewed different methods for the sentiment analysis of text, with a focus on Twitter. These included a lexicon approach and

three supervised learning methods, Naive Bayes, Maximum Entropy and Support Vector Machines. The three supervised learning methods generally outperform the lexicon-based approach. There are two first-order probabilistic models for Naive Bayes, Bernoulli and Multinomial. Bernoulli performs better on smaller vocabularies and Multinomial performs better on larger vocabularies. The bigram Naive Bayes outperformed the unigram and X^2 feature selection improved its accuracy. Maximum Entropy, a probability distribution estimation technique, has an advantage over Naive Bayes as it does not suffer from the independence assumption. Maximum Entropy does suffer from over-fitting, which can be improved using maximum-a-posteriori estimation. Support Vector Machines can handle large feature spaces which is useful for Twitter data. A disadvantage of Support Vector Machines is that they are a black box method. It can be hard to know exactly what is having an effect on the algorithm and how to improve it.

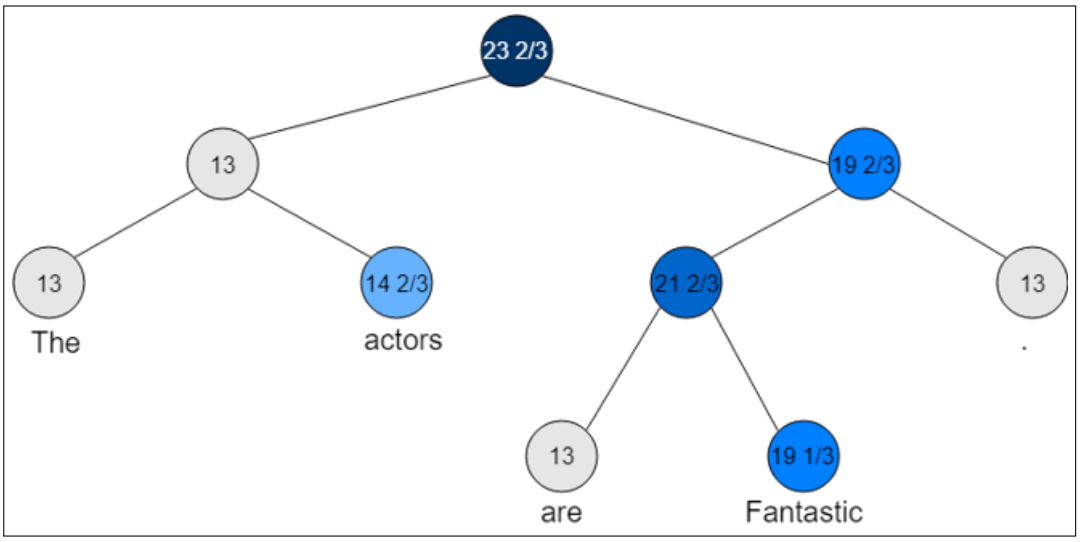

Figure 2.1: A labelled movie review from the Stanford NLP Sentiment Treebank (1).

The Stanford NLP (Natural Language Processing) Group's Sentiment Analyser (1) introduced a Recursive Neural Tensor Network (RNTN) along with a Sentiment Treebank. The Sentiment Treebank extended the corpus of movie reviews originally collected by Pang and Lee (26). The sentences in the movie review corpus were relabelled at a phrase level, producing a sentiment labelled parse-tree for each review (Figure 2.1). The Sentiment Treebank produced has more finely grained sentiment labels than the original corpus. It improved how the compositional effects of sentiment in language were captured. For example, a word may be positive in one context but negative in another. In this sentence, *'The phone has a really long battery life.'*, long is positive. However in this sentence, *'The website took so long to load.'*, long is negative. All classification models trained with the Sentiment Treebank saw a significant increase in accuracy. These included Naive Bayes, Support Vector Machines, and other recursive neural networks. The RNTN achieved the highest accuracy of 85.4% in single sentence positive/negative classification.

2.5 Recommender Systems

Recommender systems provide suggestions for items that are most likely to be of interest to a particular user (33). They recommend based on user behaviours and preferences, such as explicit user ratings and reviews of products, and implicit user clicks and view times. Some popular recommender systems include Netflix which recommends films and television programmes, Facebook which recommends new friends and content via your news feed, and TripAdvisor which recommends hotels. Recommender systems try to help consumers overcome information overload. There are two main methods that recommender systems use to generate recommendations, Collaborative Filtering methods and Content-Based recommender methods.

Collaborative Filtering methods analyse the behaviour of a collection of users and use this information to make recommendations based upon what users similar to the current user have liked. The basic idea is that if we know two users have the same opinion on one item, then they are more likely to have the same opinion on another item. The correlation between users is used to make predictions.

Content-Based (CB) recommender methods use the descriptive features of items and the preferences of individual users. Information about each user and the content information about each item is combined to make recommendations. CB methods recommend items similar to items the user liked or purchased in the past.

Context-aware recommender systems use contextual information to improve their recommendations. This contextual information could be time, location, company etc. The concept is that the same user could prefer different items under different conditions.

A number of recommender systems that focus on hotels have been proposed in the literature. A. Levi, O. Mokryn, C. Diot, and N. Taft proposed a recommender system that specifically focuses on hotel recommendations. This is a cold start, context-based hotel recommender system, which uses the text of online reviews from Tripadvisor and Venere as its main data (34). The system asks the user to identify their trip intent (business, family, etc), nationality and preferences for certain hotel features (location, service, food, etc). Hotels are recommended based on the sentiment of reviews of users who have similar context information. The sentiment score is calculated with a lexicon based approach. Adjectives that reviewers use to express opinions are considered 'opinion words'. Opinion words are extracted for each feature from every sentence of a review. A sentiment score is calculated for each feature using the lexicon. An overall feature sentiment score is then calculated based on information about the user. The feature sentiment score is combined with the scores based on the user's trip intent and nationality to produce a hotel score. A. Levi, O. Mokryn, C. Diot, and N. Taft reported that users were 20% more satisfied with their

recommendations. This is a promising result for this research. The hope is that incorporating review-like tweets into the CoRE recommender will also increase user satisfaction.

Another recommender system that targets hotels was proposed by K. Lin, C. Lai, P. Chen, and S. Hwang (35). It also uses hotel reviews collected from TripAdvisor. The system tracks the user's gestures on a mobile device to identify what part of the review the user has focused on or 'seriously read'. Feature extraction is used to extract the aspects of hotels (e.g room, food, price etc) the user considers important, and build a user interest profile. Hotels are recommended based on the user profile. The score is calculated based on the sentiment of the reviews about the aspects of the hotels the user prefers.

J. Chang, C. Tsai, and J. Chiang (36) proposed a hotel recommendation system that uses a combination of Twitter and Yelp data. They use a collaborative filtering method. The idea is that Twitter provides limited information on its own, but when combined with Yelp, which has explicit user ratings, can achieve better performance in a recommender system. User posting behaviour vectors are generated for both Twitter and Yelp and are combined to make recommendations.

T. Takehara, S. Miki, N. Nitta and N. Babaguchi propose a rule-based method of extracting context information from Twitter for use alongside a restaurant recommender system (37). Relevant keywords are extracted from reviews from Tabelog [1] (a Japanese review site) and are used to search Twitter for the contextual information. Nouns are extracted from Tabelog using part-of-speech tagging. Those characteristically used for assessing restaurants in each area are selected as keywords and categorised as area related keywords or restaurant related keywords. Finally, tweets containing more than two nouns from each set of keywords (area and restaurant) are selected as the contextual information and are displayed alongside the restaurant recommendations.

CoRE (5) is a Cold Start Resistant and Extensible Recommender system proposed by M. Bayomi, A. Caputo, M. Nicholson, A. Chakraborty and S. Lawless in collaboration with Ryanair. It is a hotel recommender system that makes use of collaborative filtering, content-based recommendation and contextual suggestion. The algorithm is designed to work in cold start situations and to be easily extensible. A cold start situation is when you need to make recommendations for a new user that you have no previous information on. CoRE is made up of three models, a user model, a segment model and a context model. Each model is generated as a weighted feature vector. A centroid vector is then calculated, and weighting is applied to determine the effect of each model. The cosine similarity between the vector of each hotel and the centroid vector of the user is then calculated, and the hotels are ranked based on this score. CoRE was evaluated against the recommendation approach currently used by Ryanair. It significantly outperformed this baseline system.

[1] `https://tabelog.com`

2.6 Summary

In this chapter, the background literature relating to this project has been discussed. A series of text classification techniques for classifying tweets were outlined but there was no obvious winner in terms of performance. A. Bermingham and A. F. Smeaton (25) found that the Multinomial Naive Bayes Classifier outperformed the Support Vector Machine Classifier in classifying tweets and short-form reviews. Ankit and N. Saleena, and M. Kanakaraj and R. M. R. Guddeti (27, 28) both found their Ensemble Classification Methods outperformed the base classifiers in classifying tweets. The study most similar to ours was by A. Rane and A. Kumar (3) and dealt with airline reviews. A. Rane and A. Kumar (3) found that the Random Forest Classifier followed closely by the Support Vector Machine achieved the best performance. It will be interesting to see which if any of these results are confirmed by our research. The majority of the tweet classification studies focus on classifying the sentiment of tweets. It was difficult to find any studies about classifying tweets as reviews. That is the area that this project will focus on.

This chapter also presented the different methods that can be used to analyse the sentiment of text. This included lexicon based and machine learning based approaches. It was through this research that the Stanford NLP Sentiment Analyser (1) was discovered. It was used to classify the sentiment of the tweets in this research.

Finally, this chapter outlined some techniques used in hotel recommender systems. The most promising result came from A. Levi, O. Mokryn, C. Diot, and N. Taft (34) who demonstrated that incorporating TripAdvisor reviews into their hotel recommender system increased the average user satisfaction. Our hope is that incorporating Twitter reviews into the CoRE recommender will have a similar effect.

3 Design and Methodology

3.1 Introduction

In this chapter, the design and methodology choices involved in this project will be presented. This project consists of five key stages (Figure 3.1). Each of these will be discussed in detail in this chapter. The five stages are:

- **Data Collection and Filtering**
  This stage involved collecting the Twitter data. The data was then filtered so it only contained tweets about hotels posted from Dublin.
- **Dataset Annotation**
  This stage involved annotating the filtered dataset of tweets about hotels in Dublin.
- **Classification**
  In this stage of the project, multiple supervised machine learning classifiers were trained with different feature representations.
- **Sentiment Analysis**
  This part of the project involved analysing the sentiment of those tweets that were classified as reviews.
- **Application within a Recommender System**
  In this stage, the sentiment scores produced were used to re-rank the results of the CoRE (5) Recommender System.

Figure 3.1: An overview of the key stages of the project.

3.2 Data Collection

The data was collected using the Twitter Streaming API [1] (Application Programming Interface) between October 2017 and September 2018. Twitter has a Search API and a Streaming API. The Search API allows you to find historical tweets and the Streaming API allows you to stream real-time tweets.

The tweets used in this project were collected using the Streaming API. A location filter was specified so that only tweets posted from within the bounding box of Dublin were returned (Figure 3.2). A total of 2.5 million tweets were collected.

After inspecting the collection of tweets it was found that although a bounding box had been specified, not all tweets in the collection had been posted from Dublin. A significant amount of tweets had slipped through Twitter's location filter. This meant that the first step required was to filter the dataset to ensure, as much as was possible, that all tweets related to Dublin.

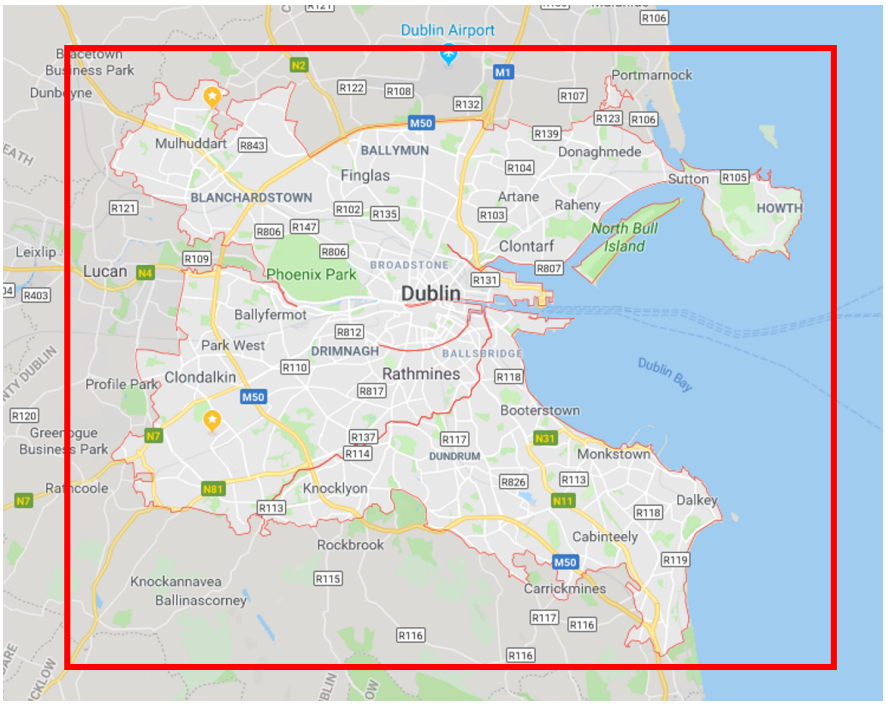

Figure 3.2: Bounding Box of Dublin.

[1] https://developer.twitter.com

3.3 Data Filtering

3.3.1 Geo-tagged Tweets

All of the tweets in the dataset were Geo-tagged because they were collected with a location filter. This meant they all had location data: a specified location from which the tweet was posted. There are two types of Geo-tagged tweets:

- **Point Coordinate**
  Tweets with a specific latitude/longitude or point coordinate. These tweets come from GPS enabled devices (Listing 3.1).
- **Twitter Place**
  Tweets with a specified bounding box and Twitter Place. A bounding box is a four-sided geographic area, defined by four points of the form [longitude, latitude] (Figure 3.2). This defines the general area the tweet was posted from (Listing 3.2).

```
"geo": {
    "type": "Point",
    "coordinates":
        53.28581863,
        -6.11439315
    ]
}
```

Listing 3.1: Geo-tagged Tweet with Point Coordinate

```
"place": {
  "full_name": "Dun Laoghaire-Rathdown, Ireland",
  "url": "https://api.twitter.com/1.1/geo/id/723427e351a01e72.json",
  "country": "Ireland",
  "place_type": "city",
  "bounding_box": {
    "type": "Polygon",
    "coordinates": [
      [
        [
          -6.282038,
          53.199283
        ],
        [
          -6.282038,
          53.315283
        ],
```

```
        [
          -6.066759,
          53.315283
        ],
        [
          -6.066759,
          53.199283
        ]
      ]
    ]
  },
  "country_code": "IE",
  "attributes": {},
  "id": "723427e351a01e72",
  "name": "Dun Laoghaire-Rathdown"
}
```

Listing 3.2: Geo-tagged Tweet with Twitter Place

3.3.2 Filtering Out Non-Dublin Tweets

The first attempt to filter out any non-Dublin tweets used the point coordinates of the tweets. The point coordinate of each tweet was compared to the bounding box of Dublin. If it lay inside the box it was kept, otherwise, it was filtered out. However, only 7.23% of the tweets in our collection actually had a specific point coordinate. Filtering based on this excluded the majority of the dataset. To address this we instead filtered tweets based on a combination of their point coordinates and their Twitter places.

All of the tweets in the collection have a Twitter place defining the general area that the tweet was posted from. A combination of the Twitter place and the point coordinates was used to filter out any non-Dublin tweets that had ended up in the dataset.

Table 3.1: The coordinates of Dublin's Bounding Box.

North Latitude	53.425210
South Latitude	53.223430
East Longitude	-6.043924
West Longitude	-6.447485

A bounding box for the Dublin area was defined (Table 3.1). Tweets with a point coordinate and tweets with only a Twitter place were filtered differently:

- **Point Coordinate**
  Each tweet with a point coordinate was checked to see if the point coordinate fell

within the defined bounding box for Dublin. If it lay inside the box it was kept, otherwise, it was filtered out.

- **Twitter Place**
  Each tweet with a Twitter place has a bounding box specifying the location of the place. The centroid of this bounding box was calculated for each tweet. Then the centroid was checked to see if it fell within the defined bounding box for Dublin. If it lay inside the box it was kept, otherwise, it was filtered out.

The filtered dataset consisted of 1.6 million tweets posted from Dublin between October 2017 and September 2018.

3.3.3 Filtering Tweets About Hotels

The dataset of tweets posted from Dublin now had to be further filtered. The next step was to extract all tweets that mentioned hotels.

A list of the hotels in Dublin was compiled. This included the hotel's name and the hotel's Twitter handles (@hotelname). This list consisted of 159 hotels and Twitter handles (Table A1.1).

The tweets were stored in a Lucene index. A fuzzy search query was used to match the tweets against each of the hotel names and hotel Twitter handles. The fuzzy search query uses a similarity measure that is based on the Damerau-Levenshtein algorithm. The maximum edits option was set to two, meaning that strings with a maximum difference of two characters would still match. This accounted for misspellings and broadened our search slightly. We experimented with higher numbers of maximum edits but found that too many irrelevant tweets were returned.

No NLP (natural language processing) technique that attempts to identify all mentions will be 100% accurate. Users will refer to hotels in all manner of ways, anaphora like 'the hotel'. We have tried to be reasonably conservative and ensure we have a pretty clean dataset, but there are a range of ways to identify hotels that could be experimented with. This further filtered dataset consisted of 3115 tweets that mention hotels posted from Dublin between October 2017 and September 2018. The distribution of tweets collected per hotel can be seen in the appendix (Table A1.2).

3.4 Dataset Annotation

One major goal of this research was to categorise tweets as review-like tweets, as tweets that contain some content and as irrelevant tweets. In order to train a classifier to do this a set of tweets had to first be manually annotated. The set of 3115 tweets about hotels in Dublin was annotated. This involved building an annotation webpage where users could view tweets and assign them labels.

3.4.1 Annotation Webpage

A simple webpage was created in order to annotate the tweets (Figure 3.3). The webpage can be found at:

`http://reviewtweets.epizy.com`

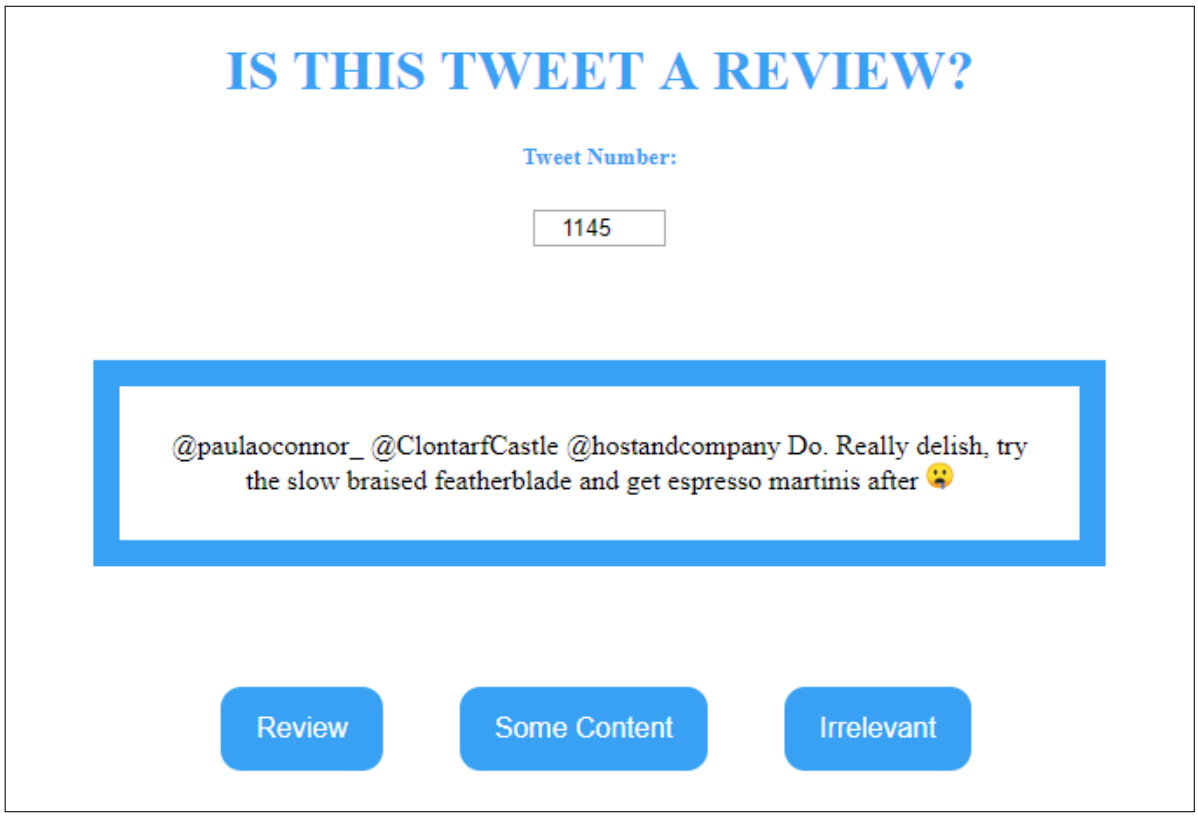

Figure 3.3: Tweet Annotation Webpage.

The text of each tweet was displayed alongside three buttons: 'Review', 'Some Content' and 'Irrelevant'. The participant could click on the option that they thought best described the tweet shown. Once they chose a label, the next tweet would be displayed.

Each time the webpage is loaded a random tweet from the dataset is displayed. This means each participant will label a different section of the tweets, ensuring all tweets in the collection get annotated evenly.

The following instructions, describing what a 'Review', 'Some Content' and 'Irrelevant' tweet should look like accompanied the webpage:

- **Review**
  The tweet could be considered as a review (of any aspects related to a hotel such as the venue, food, view, swimming pool, etc.) for any hotel. Examples would include: *"Amazing view of the Aviva Stadium from my hotel balcony at hotel X"* (positive review), "Room service was awful at hotel Y" (negative review), *"Thank you hotel X for a lovely stay"* (positive review) or *"Had an awful night at hotel Y"* (negative review).

- **Some Content**
  The tweet doesn't look like a review, but it does provide some information related to a hotel, such as the hotel hosts events, information on the menu, information related to accommodation, etc. Examples would include: *"Hotel Z serves Tuna salad on Wednesday"* or *"A packed room for the 2018 fashion conference at Hotel X"*.

- **Irrelevant**
  This tweet is completely irrelevant. While perhaps mentioning the name of a hotel, the tweet doesn't give any additional information about that hotel or offer any opinions related to the hotel.

In this research, we decided to focus solely on the text of the tweets. For this reason, all images, videos and URLs were stripped from the tweets before being displayed on the annotation webpage. Images, videos and URLs could all carry valuable information and could be considered in future research. Image and video processing techniques could be experimented with to gather additional information from the tweets.

The tweets were stored in a SQL table (Table 3.2) linked to the webpage, along with a count of the number of times the tweet had been labelled as a review, some content or irrelevant. Each time a participant chooses a label the corresponding value in the database is incremented. The final label of a tweet is determined by calculating which label had the maximum number of votes.

The annotation webpage was circulated to friends, family and members of the Adapt Research Centre to gather annotations.

Table 3.2: A sample of the tweets from the SQL Table.

Row No.	Tweet ID	Tweet	Review	Content	Irrelevant
1	00000	Creating A Rewarding Experience or CARE, essence of any DoubleTree by Hilton hotels, is in the heart of everything in the Morrison Hotel! #CARE #DoubleTree #dublincity #brandculture	0	1	0
2	00000	@bfitzsimons @doubletree @DTHydePark Oohh. Impressed!	2	0	0
3	00000	SlowMo Training #EliteFest2018 @ The Morrison, a DoubleTree by Hilton Hotel	0	1	0
4	00000	Drinking a Heineken by @heineken at @doubletree —	0	2	0

3.5 Tweet Classification

The annotated set of tweets was used to train a series of different classifiers to determine which classifier was most suited to the task. These classifiers were implemented using Python's Scikit Learn library (2). The data was split into a training set and a testing set. The data was split 80:20 where 80% was used for training and 20% was used for testing.

3.5.1 Data Pre-processing

The set of annotated tweets was pre-processed before it was used to train the classifiers. The pre-processing stage tokenizes the tweets and deals with things like punctuation and case. The data pre-processing techniques used were inspired by the techniques found in previous literature (3, 27, 29, 31).

The following steps were taken to pre-process the tweets before they were used to train the classifiers:

- **Emojis**
  Emojis were removed and replaced with text using Python's Emoji library (38). For example, a thumbs up emoji would be converted to the text *':thumbs_up:'* and then *'thumbs up'* once punctuation is removed. In this way, some meaning was extracted from each emoji. Emojis are another area of potential valuable sentiment information for future work. A. Go, R. Bhayani and L. Huang (29) used emojis to automatically label a dataset of tweets as positive or negative.

- **Punctuation**
  The majority of special characters and punctuation was removed from the tweets. This included everything except the digits 0 - 9, the letters a - z (in upper and lower case), and the following set of symbols: . , ? ! ( ) & ' - . Tweets are generally quite noisy and contain a lot of unnecessary punctuation. Symbols such as @ and # were removed so that the words contained in hashtags and Twitter handles could be recognised as words. However, symbols like apostrophes were not removed so that a word like 'won't' wouldn't be split into 'won' and 't'.

- **Single Characters**
  All single characters were removed. The majority of the single characters in the tweets were typos or added no extra meaning to the text. The only single character words are 'I' and 'a'. These are both stop words which are later removed anyway.

- **Case Change**
  Words were split on case changes. For example *'MerrionHotel' –> 'Merrion Hotel'*. This was because we noticed a lot of Twitter handles and hashtags contain words joined together without space.

- **Word-digit Boundaries**
  Words were split on word-digit boundaries. For example *'HouseDublin2' –> 'House Dublin 2'*. Again, a lot of Twitter handles and hashtags contain words and digits joined together without space.

- **Lowercase**
  All text was converted to lower case. This is so that a word with a capital letter and a word without a capital letter are recognised as the same word.

- **Stemming**
  Stemming was performed using the Word Net Lemmatizer from the Natural Language Toolkit [2] (NLTK). Stemming is the process of reducing words down to their base or root. For example *'organised'* and *'organisation'* would both be reduced to *'organise'*. Stemming conflates similar terms, reducing the number of words fed into the classifier.

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

- **Stopwords**
  Stop words were removed. All text contains words that are irrelevant and do not add any additional meaning. These are called stop words. Some examples of stop words are *'and'*, *'I'*, *'of'* and *'the'*. Stop words were not removed in all cases. It will be clearly stated in all cases where stop words were removed.

3.5.2 Feature Extraction

The classifiers require numerical feature vectors of fixed length rather than variable length raw text as their input. To comply with this our processed tweets had to be converted into numerical feature vectors. This process of converting raw text to a numerical feature vector is called vectorization.

We experimented with seven different feature extraction methods to see which performed best at classifying the Twitter data. Twitter data is quite different to standard text. Tweets are short with a maximum of 280 characters, which has led to the use of particular characteristic features. They contain features like hashtags, emojis, Twitter handles, URLs, images, videos and gifs, which don't occur in standard text. This means that standard feature extraction methods that perform well on regular, long-form text will not necessarily perform well on Twitter data.

The seven feature extraction methods implemented were:

1. **Unigram Bag-of-Words (BOW)**
   A unigram implementation of BOW using Scikit Learn's Count Vectorizer.
2. **Unigram Term Frequency - Inverse Document Frequency (TF-IDF)**
   A unigram implementation of TF-IDF using Scikit Learn's Count Vectorizer and TF-IDF transformer.
3. **Bigram TF-IDF**
   A bigram implementation of TF-IDF using Scikit Learn's Count Vectorizer and TF-IDF transformer.
4. **Trigram TF-IDF**
   A trigram implementation of TF-IDF using Scikit Learn's Count Vectorizer and TF-IDF transformer.
5. **Unigram TF-IDF (stop words removed)**
   A unigram implementation of TF-IDF with stop words removed using Scikit Learn's Count Vectorizer, TF-IDF transformer and English stop word list.

6. **Word2Vec**
   Gensim's implementation of word2vec with Google's pretrained model.
7. **Doc2Vec**
   Gensim's implementation of doc2vec, each tweet being considered a document.

Bag-of-Words

In Bag-of-Words (BOW) documents are described by word occurrences. A vocabulary is created of all unique terms in the dataset. The vocabulary is ranked by frequency of occurrence. The maximum size of the vocabulary can be specified, so only the top n terms are kept. Each tweet is then represented as a feature vector consisting of ones and zeroes. One representing a term's occurrence and zero representing a term's absence. A major drawback of BOW is that it does not take word order into account. This is why it is called 'bag' of words. You can imagine that all the words have been thrown into a bag together and have lost any order.

The Count Vectorizer from Python's Scikit Learn was used to implement BOW.

TF-IDF

BOW can be extended with TF-IDF (Term Frequency - Inverse Document Frequency). Term frequency is the frequency of the word in the current document. Inverse Document Frequency takes into account how often the word occurs in the whole dataset. The idea is to balance how important a term is in a document versus how important it is in the entire collection. We are less interested in a very frequently occurring term like for example 'the' or 'and' than some less frequently occurring words which occur frequently in a small number of documents. The TF-IDF score is used to re-weight the count features produced by the Count Vectorizer.

TF-IDF

$$TF - IDF = Term\ Frequency \times Inverse\ Document\ Frequency \tag{1}$$

$$Term\ Frequency = \frac{Number\ of\ Term\ Occurrences\ in\ a\ Document}{Number\ of\ Terms\ in\ a\ Document} \tag{2}$$

$$Inverse\ Document\ Frequency\ = \log(\frac{1\ +\ n}{1\ +\ df(t)}) + 1 \tag{3}$$

$$n = Total\ number\ of\ documents\ in\ the\ dataset.$$

$$df(t) = Number\ of\ documents\ in\ the\ dataset\ containing\ term\ t.$$

The TF-IDF transformer from Python's Scikit Learn was used to implement TF-IDF.

N-grams

Another extension of BOW is the use of N-grams. N-grams help to address the problem BOW has with discarding word order.

An N-gram is a sequence of N consecutive terms. For example, the bigrams of *'Twitter as an Alternative Review Site'* are:

- *'Twitter as'*
- *'as an'*
- *'an Alternative'*
- *'Alternative Review'*
- *'Review Site'*

Combining bigrams with BOW means that occurrences of pairs of consecutive words are counted instead of individual terms. Unigram, bigram and trigram versions of TF-IDF were implemented.

Word2Vec

Word2Vec is another method for converting text into numerical feature vectors. It takes a dataset as its input and produces a vector space. Each word in the dataset is represented by a corresponding vector. It groups the vector of similar words together in the vector space. Cosine similarity, the cosine of the angle between two vectors, measures the similarity of vectors in the vector space.

Gensim's implementation of Word2Vec (39) and Google's pre-trained model was used. Google's model includes word vectors for three million words and phrases. It was trained with about 100 billion words from a Google News dataset. The word vectors of each word in a document are averaged to produce a feature vector for that document.

Doc2Vec

Doc2Vec takes the same idea as Word2Vec, but instead of words being represented by vectors, full documents are represented as vectors. This captures the relationship between words, which Word2Vec does not.

Gensim's implementation of Doc2Vec was again used. Unlike Word2Vec, we built our own Doc2Vec vocabulary based on the training data.

3.5.3 Classifiers

Thirteen different classifiers were implemented, each with their implementation from Python's Scikit Learn library (2).

The twelve classifiers implemented were:

1. Decision Tree (DT) Classifier.
2. Random Forest (RF) Classifier.
3. Multi Layer Perceptron (MLP) Classifier.
4. Logistic Regression (LR) Classifier.
5. Support Vector Machine (SVM) Classifier.
6. K Nearest Neighbours (KNN) Classifier.
7. Gaussian Process (GP) Classifier.
8. Adaboost (AB) Classifier.
9. Gaussian Naive Bayes (GNB) Classifier.
10. Bernoulli Naive Bayes (BNB) Classifier.
11. Multinomial Naive Bayes (MNB) Classifier.
12. Quadratic Discriminant Analysis (QDA) Classifier.
13. Linear Discriminant Analysis (LDA) Classifier.

Decision Tree

The Decision Tree Classifier is a simple classification algorithm that can be used for binary or multi-class classification. DT classifiers learn simple decision rules based on the attributes of the training data. These decision rules form a tree structure (Figure 3.4) which is used to predict the value of a target variable. The leaf nodes represent the class labels. When an unclassified document is received questions are asked until a leaf node is reached and the document is assigned that class.

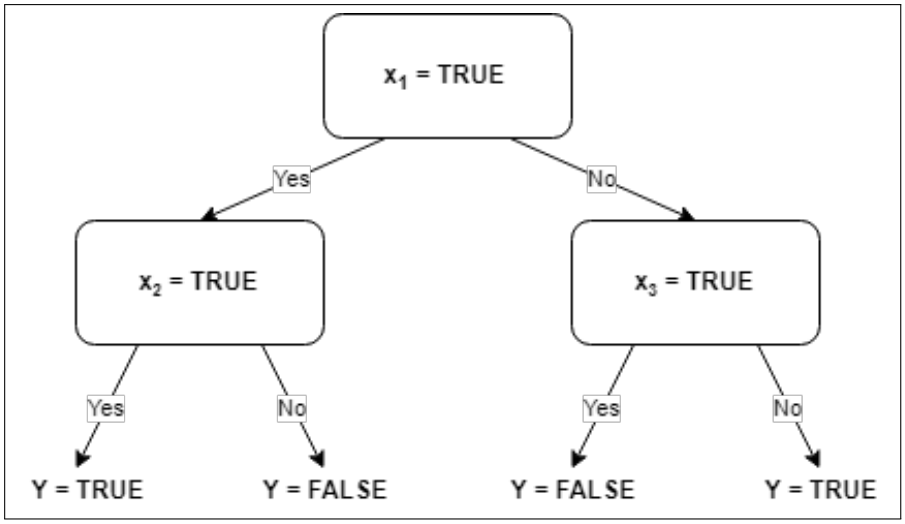

Figure 3.4: An example of a Decision Tree Classifier.

The DT Classifier was implemented using Scikit Learn with the following parameters:

DecisionTreeClassifier (criterion='entropy', max_depth=32, max_features=None, min_samples_leaf=2, min_samples_split=0.1)

Random Forest

The Random Forest Classifier is an ensemble classification algorithm, meaning it combines multiple base classification algorithms. It consists of multiple DT classifiers. The final class prediction is calculated by getting the average of the decisions of the individual decision trees.

The RF Classifier was implemented using Scikit Learn with the following parameters:

RandomForestClassifier (n_estimators=500, max_features='log2', criterion='entropy')

Multi Layer Perceptron

The Multi Layer Perceptron Classifier is a deep, feedforward, artificial neural network. It consists of a minimum of three layers (Figure 3.5): an input layer, a hidden layer and an output layer. The input layer receives the data, the output layer makes a decision and the hidden layers approximate the function.

The MLP classifier learns a function based on the training set. Given a set of features $X = x_1, x_2, x_3, ..., x_n$ and a target y, it learns a non-linear approximation function. Backpropagation

is used to train the MLP Classifier.

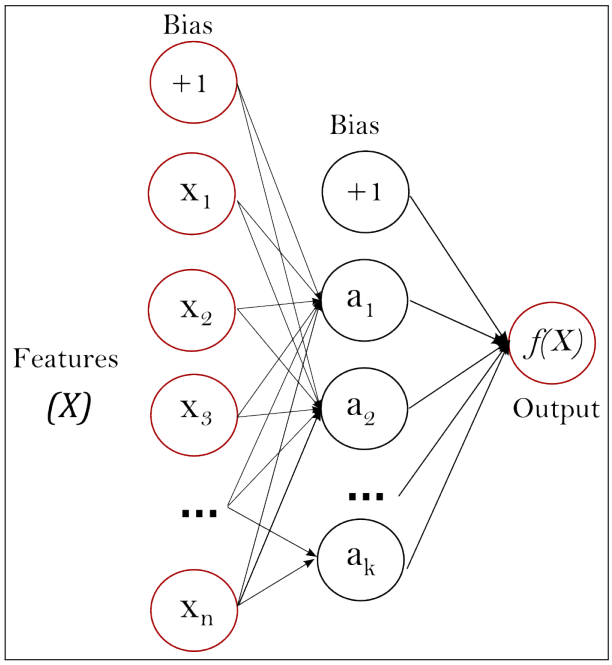

Figure 3.5: A Multi Layer Perceptron Classifier with one hidden layer (2).

The MLP Classifier was implemented using Scikit Learn with the following parameters:

MLPClassifier (alpha=1,activation='identity', hidden_layer_sizes=(100,), learning_rate='constant', solver='adam')

Logistic Regression

The Logistic Regression Classifier is a linear classification algorithm that uses the logistic function to model the training data. The logistic function is as follows:

$$g(z) = \frac{1}{(1 + e^{-z})} \tag{4}$$

The LR Classifier was implemented using Scikit Learn with the following parameters:

LogisticRegression (C=1,multi_class='multinomial',penalty='l2',solver='saga')

Support Vector Machine

The Support Vector Machine Classifier is a discriminative classifier (Figure 3.6). It finds the optimum hyperplane that separates the data into the labelled classes. It aims to maximise

the distance between the hyperplane and the support vectors (the points closest to the hyperplane).

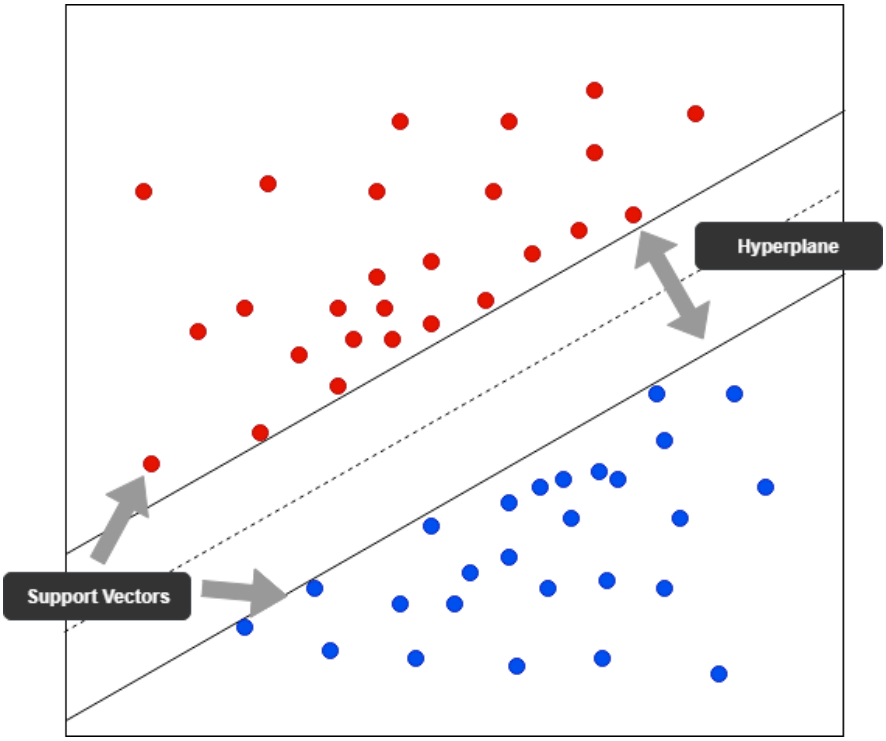

Figure 3.6: An example of a Support Vector Machine Classifier.

The SVM Classifier was implemented using Scikit Learn with the following parameters:

```
SVC (C=10.1,decision_function_shape='ovo',degree=1,gamma=1,kernel='rbf')
```

K Nearest Neighbours

The K Nearest Neighbours Classifier doesn't build a model like other classification algorithms. It uses a majority vote of the "nearest neighbours" to a document. A label is assigned to a document based on the class that has the majority of the nearest neighbours to the document. For example in Figure 3.7, point one would be assigned the label blue. Taking it's four nearest neighbours it has three blue neighbours and one red neighbour. The point is assigned to the class with the majority of the nearest neighbour, blue.

The KNN Classifier was implemented using Scikit Learn with the following parameters:

```
KNeighborsClassifier(n_neighbors=17,algorithm='ball_tree',weights='distance',
leaf_size=10,p=2)
```

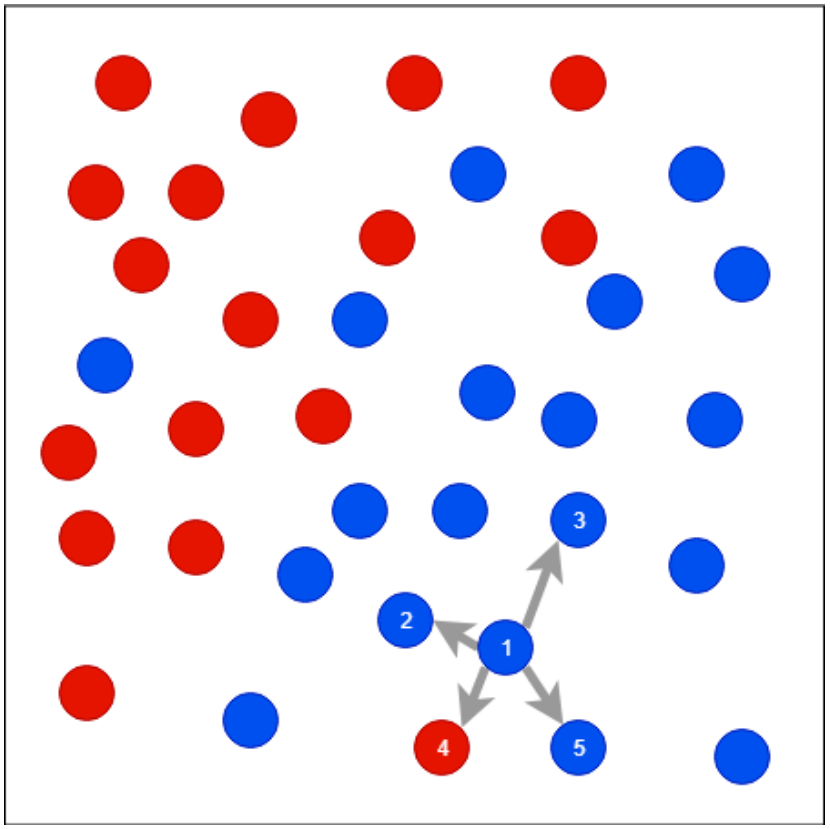

Figure 3.7: An example of a K Nearest Neighbours Classifier.

Gaussian Process

The Gaussian Process Classifier implements Gaussian Processes for classification. Gaussian Processes are probability distributions over possible functions.

The definition of a Gaussian Process is: $P(f)$ is a Gaussian process if for any finite subset $\{x_1, x_2, ..., x_n\} \subset X$, the marginal distribution over that finite subset $P(f)$ has a multivariate Gaussian distribution.

The GP Classifier puts a Gaussian Process on a latent function, which is then squashed through a link function to get the probabilistic classification. For classification, the posterior of the latent function is not Gaussian, instead, the logit function is used. A Gaussian likelihood function is inappropriate for discrete class labels (40).

The GP Classifier was implemented using Scikit Learn with the following parameters:

```
GaussianProcessClassifier(kernel=1.0 * RBF(1.0), optimizer='fmin_l_bfgs_b')
```

AdaBoost

The AdaBoost Classifier is another ensemble classification algorithm, introduced by Freund and Schapire (41).

The AB Classifier works by fitting a series of weak base classifiers on incrementally re-weighted versions of the training data. The predictions from the base classifiers are combined with a majority vote or sum, to get an overall classification. The 'boosting' component of the algorithm involves re-weighting the data on each iteration. Samples that were predicted correctly have their weight increased and samples that were predicted incorrectly have their weight decreased.

The AB Classifier was implemented using Scikit Learn with the following parameters:

```
AdaBoostClassifier(base_estimator=None, n_estimators=50,learning_rate=1,
algorithm='SAMME.R')
```

Scikit Learn's version of AdaBoost implements the SAMME (Stagewise Additive Modeling using a Multi-class Exponential loss function) algorithm (42).

Gaussian Naive Bayes

The Gaussian Naive Bayes Classifier is one of a set of Naive Bayes Classification algorithms. All of the Naive Bayes classifiers are based applying Bayes Rule along with the 'naive' assumption that features are conditionally independent. Bayes Rule:

$$P(A \mid B) = \frac{P(B|A)\,P(A)}{P(B)}.$$

In the GNB Classifier the likelihood of the features is assumed to be Gaussian:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2})$$

The GNB Classifier was implemented using Scikit Learn with the following parameters:

```
GaussianNB (priors=None, var_smoothing=1e-09)
```

Bernoulli Naive Bayes

The Bernoulli Naive Bayes Classifier is another of the Naive Bayes Classification algorithms. The BNB Classifier implements the Naive Bayes algorithm for data that is distributed according to multivariate Bernoulli distributions. It has been seen to work better than the GNB Classifier on shorter texts.

The decision rule for the BNB Classifier is:

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i)$$

The BNB Classifier was implemented using Scikit Learn with the following parameters:

```
BernoulliNB (alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

Multinomial Naive Bayes

The Multinomial Naive Bayes Classifier is another of the Naive Bayes Classification algorithms. MNB implements the Naive Bayes algorithm for multinomial distributed data.

The MNB Classifier was implemented using Scikit Learn with the following parameters:

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Quadratic Discriminant Analysis and Linear Discriminant Analysis

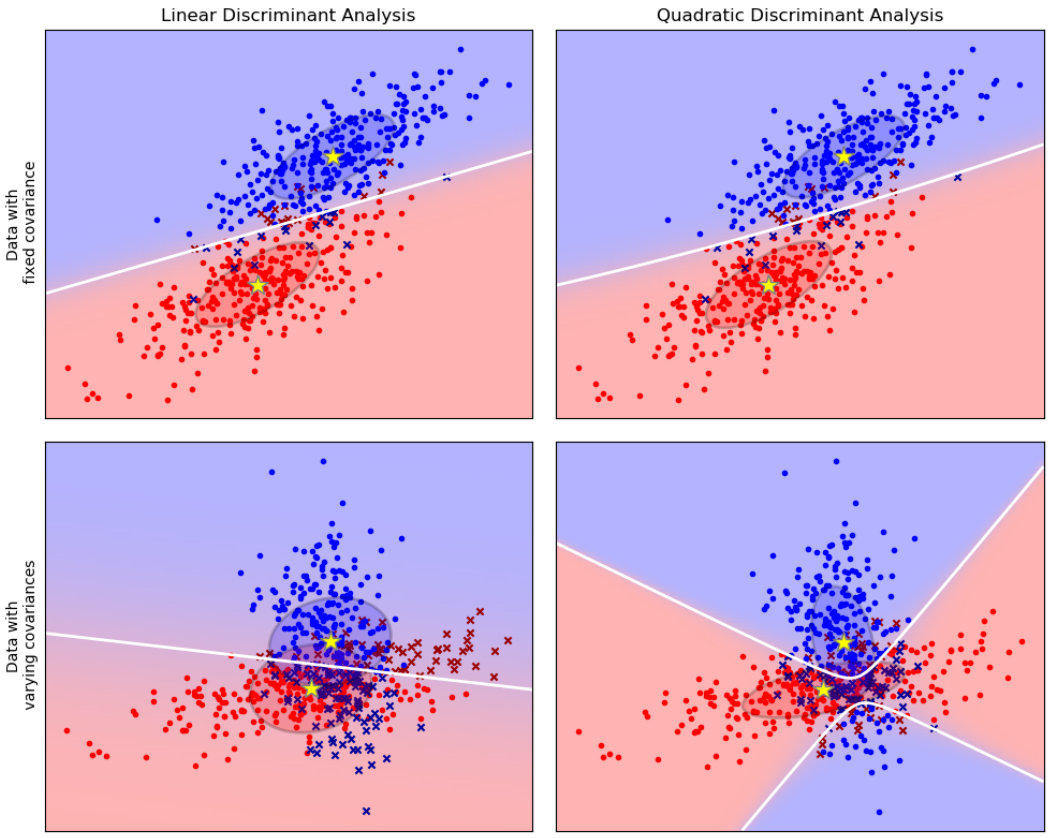

Figure 3.8: Boundaries of the Quadratic Discriminant Analysis and Linear Discriminant Analysis Classifiers (2).

The Quadratic Discriminant Analysis Classifier and the Linear Discriminant Analysis Classifier, have a quadratic and linear decision surface, respectively. QDA is more flexible as it can learn quadratic boundaries, LDA can only learn linear boundaries (Figure 3.8).

Both the QDA and the LDA Classifier were implemented using Scikit Learn with the following parameters:

QuadraticDiscriminantAnalysis(priors=None, reg_param=0.0, store_covariance=False, store_covariances=None, tol=0.0001)

LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None, solver='svd', store_covariance=False, tol=0.0001)

3.6 Sentiment Analysis

Once the tweets had been classified as reviews, sentiment analysis was performed on them. Sentiment analysis is the process of identifying the opinion expressed about a particular subject in some text. The goal is to determine whether the opinion is positive, negative or neutral, and to what extent.

The Stanford NLP Sentiment Analyser (1) was used to classify the sentiment of the tweets. The Stanford NLP Sentiment Analyser was chosen as it is the dominant NLP library used in research in this area. It has achieved 85.4% accuracy in single sentence positive/negative classification.

3.6.1 The Stanford NLP Sentiment Analyser

The Stanford Sentiment Analyser is based on a Recursive Neural Tensor Network. It was trained on a sentiment treebank of movie reviews from the movie review site RottenTomatoes. The treebank consists of sentences which are annotated on a phrase level.

The Stanford NLP Sentiment Analyser has its limitations. Due to the fact that it was trained on movie reviews and is being applied to tweets, optimum performance is not achieved. The best performance would be achieved if the Sentiment Analyser was re-trained on a dataset as similar to ours as possible. Ideally, the Sentiment Analyser would be re-trained with a set of hotel-related tweets. This would involve manually annotating a collection of tweets at a phrase level to produce a treebank. This is an area for future research.

The Stanford NLP Sentiment Analyser classifies the text into one of the following five sentiment judgements:

1. Very Negative
2. Negative
3. Neutral
4. Positive
5. Very Positive

The output from the Sentiment Analyser for a tweet looks like this:

```
"sentimentValue": "3",
"sentiment": "Positive",
"sentimentDistribution": [
    0.03910365552553,
    0.18375415309071,
    0.26346977003905,
    0.42548434014862,
    0.0881880811961
]
```

Listing 3.3: Output from Stanford NLP Sentiment Analyser

The sentiment value (3) and sentiment (Positive) define the class the tweet has been assigned (Listing 3.3). In this case, the tweet is positive. The sentiment distribution shows how strongly the tweet aligns with each class, from very negative to very positive. This tweet has a maximum value of 0.42548434014862 so it is assigned the positive label.

3.6.2 Normalisation

The sentiment scores produced by the Stanford NLP Sentiment Analyser were produced per tweet. They needed to be normalised so that they were per hotel and lay between zero and one.

A majority voting technique was used to normalise the scores. For each hotel, we had a selection of tweets. These tweets are divided into five clusters; very negative, negative, neutral (ignored), positive and very positive. A mean value was calculated for each cluster. This was calculated by summing the corresponding score in the sentiment distribution for each tweet and dividing by the number of tweets in the cluster. The class with the highest mean value was assigned to the hotel.

There were some cases where tweets related to a hotel chain, for example, Hilton Hotels, rather than an individual Hilton Hotel, for example, Hilton Dublin Kilmainham. In these

cases, sentiment scores were generated for the chain as a whole. The score for the chain was assigned to each hotel in the chain. This only happened in the cases where we had no information about the individual hotel but had information about the chain as a whole, in an attempt to boost the numbers of hotels we had sentiment scores for. The problem is that one Hilton Hotel could be amazing and another could be terrible. Ideally more data would be collected so that every hotel had its own individual sentiment score.

This mean score was then normalised between zero and one, with the following weighting.

- Very Negative: 0 –> 0.25
- Negative: 0.25 –> 0.5
- Positive: 0.5 –> 0.75
- Very Positive: 0.75 –> 1

The following max-min normalisation formulae were used:

Normalisation

$$\textit{Normalised Score } (-) = \textit{max score} - (\textit{actual score} \times (\textit{max score} - \textit{min score}))$$
$$\textit{Normalised Score } (+) = (\textit{actual score} \times (\textit{max score} - \textit{min score})) + \textit{min score}$$

Taking the tweet above with a positive score of 0.42548434014862 as an example:

Example

$$\textit{Normalised score } (\textit{Positive}) = (0.42548434014862 \times (0.75 - 0.5)) + 0.5.$$
$$\textit{Normalised score } (\textit{Positive}) = 0.606$$

3.7 CoRE Recommender System

The sentiment scores produced by the Stanford NLP Sentiment Analyser (1) were used to re-rank the list of hotel recommendations produced by the CoRE Recommender System (5). CoRE is a Cold Start Resistant and Extensible Recommender system, developed in collaboration with Ryanair.

3.7.1 CoRE

The CoRE recommender is an algorithmic approach to hotel recommendation. It can function in extreme cold start conditions. CoRE is a hybrid recommender that makes use of

collaborative filtering, content-based recommendation and contextual suggestion. It is made up of three models, a user model, a segment model and a context model. Each model is generated as a weighted feature vector.

The user model takes hotel features from Expedia, such as Free Wi-Fi, Free Breakfast, Restaurant, Hair dryer, etc. and uses them to build a vector of features describing a hotel. A vector of features is stored for each hotel. Then, for each user who has a previous hotel booking, the user model is constructed by weighting the features of the hotels that the user has booked before.

The segment model introduces collaborative filtering to the recommender. Collaborative filtering methods recommend to users based on the preferences of similar users. The segment model is constructed as a weighted vector of features of hotels that have been booked by all users from a specific segment. A segment is a group of similar users. Users are assigned to a segment based on their flight booking history.

The context model uses contextual information to recommend hotels to the user. The contextual information used is the trip type specified by the user on their flight booking. If trip type information is not available it can be inferred by the number of people travelling with the user, for example, two adults and two children indicate a family trip. The context model is constructed as a weighted vector of features of hotels that have been booked by all users who had the same trip type.

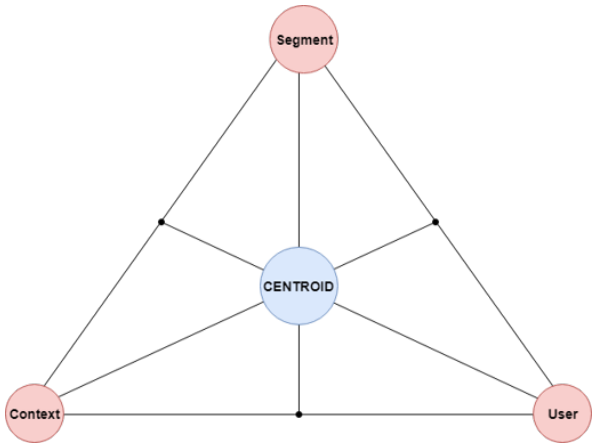

Figure 3.9: Centroid of the three CoRE Models.

The three models are combined by calculating the centroid vector (Figure 3.9). A weighting is applied to determine the effect of each model in the final vector. The cosine similarity between the vector of each hotel and the centroid vector of the user is then calculated, and a list of hotels is produced. The hotels are ranked based on the similarity score. CoRE is easily extensible. A fourth model could easily be added and incorporated into the centroid calculation.

CoRE was evaluated against the ranking approach currently used by Ryanair. Hotels from the target city are sorted based on their sequence number which is given to them by Expedia. The sequence number is based on the hotel's transactional data from the last 30 days. CoRE significantly outperforms this baseline, which is labelled 'Sequence' in Figure 3.10. CoRE achieves a mean percentile rank (MPR) of 22.04% without feature weighting and 18.27% with feature weighting.

Used Models	# Users	Approach	MPR (%)
User Segment Trip Type	22	Sequence	63.50
		CoRE	22.04
		$CoRE_{FW}$	18.27

Figure 3.10: Performance of CoRE with multiple models.

3.7.2 Re-Ranking

The scores produced by the Stanford NLP Sentiment Analyser were used to re-rank the list of hotels produced by the CoRE. This was done by simply multiplying the score of the hotel in the CoRE ranked list by the normalised sentiment score. The sentiment score normalised between zero and one boosts the score of hotels with a positive sentiment score and drags down the score of the hotels with a negative sentiment score.

$$SentiCoRE\ Score = Sentiment\ Score \times CoRE\ Score \tag{5}$$

4 Evaluation

4.1 Introduction

This chapter will present the results of this research. First, the evaluation metrics that were used to evaluate the classifiers will be introduced. The results of the evaluation of the thirteen classifiers with the seven different feature representations will be outlined. The methods used to evaluate the recommender system will be defined. Finally, the results of this evaluation will be presented.

4.2 Evaluation Metrics for Classification

4.2.1 Confusion Matrix

A confusion matrix is a technique commonly used for evaluating the performance of a classification model. It presents a summary of the prediction results of the classifier. A confusion matrix consists of four different combinations of the actual values versus the predicted values (Table 4.1).

Table 4.1: Confusion Matrix.

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

The values in a confusion matrix are:

- **True Positive (TP)**
  You have a True Positive value if your algorithm classified an item as positive and this was correct.

- **False Positive (FP)**
  You have a False Positive value if your algorithm classified an item as positive but the value was actually negative.
- **False Negative (FN)**
  You have a False Negative value if your algorithm classified an item as negative but the value was actually positive.
- **True Negative (TN)**
  You have a True Negative value if your algorithm classified an item as negative and this was correct.

The confusion matrix is useful for calculating the precision, recall, accuracy and f1-score of a classification algorithm. These four metrics were used to evaluate the performance of the classifiers.

4.2.2 Precision

Precision is the percentage of positive identifications that were classified correctly.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{1}$$

The higher the precision score the better. A low precision score can indicate that there is a large number of false positives. A high precision score indicates that the majority of what was classified was classified correctly. You could, however, have a very high precision score in the case where only a small portion of samples were actually classified but of these the majority were classified correctly. In this case, the precision score is high but the recall would be very low.

4.2.3 Recall

Recall is the percentage of all actual positives that were classified correctly.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2}$$

The higher the recall score the better. A low recall value can indicate that there are a large number of false negatives. A high recall value indicates that the majority of what should have been classified was classified correctly. You could, however, have very high recall and low precision if everything was classified. All the samples that should have been classified

are classified correctly, giving the high recall. But a lot of extra samples are classified that shouldn't have been classified, giving a low precision.

4.2.4 Accuracy

Accuracy is the total number of predictions the classifier got right. It is the percentage of the tweets that were classified correctly.

$$Accuracy \ = \ \frac{Number \ of \ Correct \ Predictions}{Total \ Number \ of \ Predictions} \tag{3}$$

The higher the accuracy score the better, but a high accuracy does not always give an accurate representation of the situation if there is a major class imbalance.

4.2.5 F1-Score

F1-Score is the harmonic mean of precision and recall. It is also called the F-Score or F-Measure.

$$F1 - Score \ = \ 2 \times \ \frac{Precision \ \times \ Recall}{Precision \ + \ Recall} \tag{4}$$

The higher the F1-Score the better. A high F1-Score indicates a good balance between precision and recall.

4.2.6 Multi-Class Classification

Multi-Class Classification is any classification task that involves two or more classes. This research tackles a multi-class classification problem. We aim to classify tweets as one of the following three classes: 'Review', 'Some Content', or 'Irrelevant'. For a binary classification problem, the precision, recall, f1-score and accuracy scores can be calculated simply with the above formulas. For multi-class classification, there needs to be a way of getting an average metric over the three classes, rather than an individual class score. There are three ways that precision, recall, f1-score and accuracy can be calculated for this kind of multi-class classification problem:

1. **Micro Average:**
   The micro average calculates the metrics globally. It aggregates the contributions of all classes, counting the total true positives, false negatives and false positives, to calculate the overall average scores.

2. **Macro Average:**
   The macro average calculates the metrics independently for each class. It then calculates the unweighted mean of the scores of the three classes.

3. **Weighted Average:**
   The weighted average calculates the metrics independently for each class like the macro average does. It then calculates the weighted average of the scores of the three classes. This differs from the macro average in that it takes class imbalance into account. Class imbalance means that there is a significant amount more or less samples in one class. In our case, there are significantly more tweets labelled as irrelevant than there are labelled as reviews (Table 4.2).

The metrics listed in the tables below (Tables 4.3, 4.4, 4.6, 4.5) were calculated with the weighted average.

4.3 Classification Results and Analysis

In this section, the results and evaluation of the classifiers will be presented. The classifiers evaluated include:

1. Random Forest (RF) Classifier.
2. Decision Tree (DT) Classifier.
3. Multi Layer Perceptron (MLP) Classifier.
4. Logistic Regression (LR) Classifier.
5. Support Vector Machine (SVM) Classifier.
6. K Nearest Neighbours (KNN) Classifier.
7. Gaussian Process (GP) Classifier.
8. AdaBoost (AB) Classifier.
9. Gaussian Naïve Bayes (GNB) Classifier.
10. Multinomial Naïve Bayes (MNB) Classifier.
11. Bernouilli Naïve Bayes (BNB) Classifier.
12. Quadratic Discriminant Analysis (QDA) Classifier.
13. Linear Discriminant Analysis (LDA) Classifier.

These classifiers were evaluated with seven different feature representations defined below:

1. Unigram BOW.
2. Unigram TF-IDF.
3. Bigram TF-IDF.
4. Trigram TF-IDF.
5. Unigram TF-IDF with stop words removed.
6. Word2Vec.
7. Doc2Vec.

Taking the thirteen classifiers and seven feature representations there are 91 different combinations. Each was evaluated in terms of precision, recall, accuracy and f1-score.

The set of 3115 tweets was divided into a training set and a testing set with an 80:20 split. The label distribution of the data is shown in the table below (Table 4.2).

Table 4.2: Distribution of Tweet Labels.

Label	Tweet Count
Review	566
Some Content	910
Irrelevant	1639

4.3.1 Precision

In terms of precision, the QDA Classifier with unigram TF-IDF and no stop words feature representation was the best performing classifier achieving a precision score of 0.79 (Table 4.3). This indicates that the QDA Classifier had a high percentage of positive identifications that were classified correctly. The QDA Classifier with unigram TF-IDF feature representation, this time with stop words included achieved the next highest precision score of 0.76. This further confirmed that unigram feature representation seems to work best with the QDA Classifier. The RF Classifier and the SVM Classifier, both with unigram TF-IDF feature representations were the two next best performing classifiers both achieving precision values of 0.74.

It is interesting to compare our precision scores to those in the study by A.Rane and A.Kumar (3), in which they investigated the performance of various classifiers in classifying the sentiment of Twitter reviews about airlines. They found that the RF Classifier achieved

Table 4.3: Precision of Classifiers for different Feature Representations.

	BOW	TF-IDF (Unigram)	TF-IDF (Bigram)	TF-IDF (Trigram)	TF-IDF (Unigram, No Stop Words)	Word2Vec	Doc2Vec
RF	0.72	0.74	0.7	0.69	0.73	0.71	0.59
DT	0.57	0.58	0.58	0.59	0.63	0.54	0.56
MLP	0.71	0.71	0.7	0.7	0.7	0.71	0.59
SVM	0.56	0.74	0.73	0.71	0.71	0.71	0.62
LR	0.72	0.72	0.7	0.68	0.7	0.71	0.61
KNN	0.53	0.62	0.62	0.62	0.61	0.67	0.54
GP	0.72	0.72	0.71	0.69	0.7	0.72	0.61
AB	0.63	0.62	0.63	0.62	0.64	0.63	0.57
GNB	0.6	0.6	0.63	0.63	0.59	0.64	0.59
MNB	0.71	0.67	0.68	0.66	0.67	0.32	0.35
BNB	0.73	0.73	0.71	0.71	0.72	0.65	0.58
QDA	0.72	0.76	0.73	0.67	0.79	0.59	0.57
LDA	0.63	0.64	0.65	0.63	0.62	0.69	0.66

the highest precision score, followed first by the AdaBoost Classifier and then the SVM Classifier. They did not include the QDA Classifier in their evaluation. Our results do not fully confirm or contradict the results of A.Rane and A.Kumar (3). Two out of three of our top performing classifiers agree, the RF classifier and the SVM classifier. Their precision scores are higher than ours for all the classifiers that they implemented, at approximately 0.8 compared to ours at approximately 0.7. A likely reason for this is that they trained their classifiers on a much larger dataset, a total of 14640 tweets, while our classifiers were trained on a smaller set of 3116 tweets. Supervised machine learning methods tend to perform the best on large datasets. Our results could possibly be improved by collecting and annotating a larger dataset.

4.3.2 Recall

The SVM Classifier was the best performing classifier with regard to recall, achieving a recall score of 0.74 (Table 4.4). The unigram TF-IDF and bigram TF-IDF feature representations performed equally well. This again partly agrees with A.Rane and A.Kumar (3), who found that the SVM Classifier was their second best performing classifier in terms of recall, with the RF classifier again coming out on top. Our scores are again slightly lower than those of A.Rane and A.Kumar (3).

Notice that the QDA Classifier which performed the best in precision performs pretty poorly

Table 4.4: Recall of Classifiers for different Feature Representations.

	BOW	TF-IDF (Unigram)	TF-IDF (Bigram)	TF-IDF (Trigram)	TF-IDF (Unigram, No Stop Words)	Word2Vec	Doc2Vec
RF	0.7	0.71	0.7	0.69	0.72	0.69	0.64
DT	0.61	0.6	0.61	0.62	0.65	0.56	0.59
MLP	0.72	0.72	0.71	0.71	0.71	0.72	0.62
SVM	0.59	0.74	0.74	0.72	0.72	0.72	0.64
LR	0.72	0.72	0.71	0.7	0.71	0.72	0.64
KNN	0.61	0.66	0.66	0.65	0.65	0.68	0.61
GP	0.73	0.73	0.72	0.71	0.71	0.73	0.64
AB	0.65	0.64	0.65	0.65	0.66	0.64	0.61
GNB	0.48	0.48	0.47	0.45	0.48	0.56	0.49
MNB	0.66	0.68	0.69	0.67	0.68	0.57	0.59
BNB	0.71	0.71	0.66	0.62	0.71	0.56	0.54
QDA	0.26	0.23	0.24	0.56	0.26	0.61	0.68
LDA	0.61	0.61	0.61	0.59	0.59	0.69	0.67

in terms of recall. This is a common issue and is why we have also evaluated the classifiers with the f1-score. The f1-score conveys the balance between precision and recall giving us a better idea of the actual performance of the classifier.

The GP Classifier was the next highest performing classifier, achieving a recall score of 0.73, with unigram TF-IDF feature representation. This is interesting as I have not come across any studies where a GP Classifier performs well in classifying tweets. The GP Classifier also performed well in precision achieving a precision score of 0.72.

4.3.3 F1-Score

Again, the SVM Classifier achieved the highest f1-score of 0.73 (Table 4.5), with the unigram and bigram TF-IDF feature representations again performing equally well. This high f1-score indicates that the SVM Classifier has a good balance of precision and recall. This confirms the results of both Rathi et al. (31), and A.Rane and A.Kumar (3) who both found in their studies that the SVM Classifier performed well in classifying tweets. However, it contradicts the results of Bermingham and Smeaton (25) who found that the MNB Classifier performed better than the SVM Classifier on tweets and short reviews. In our experiments, the SVM Classifier outperformed the MNB Classifier in all metrics. This difference in results could be due to the change in character count introduced by Twitter in 2017. The character count was increased from 140 characters to 280 characters. Bermingham and Smeaton

Table 4.5: F1-Score of Classifiers for different Feature Representations.

	BOW	TF-IDF (Unigram)	TF-IDF (Bigram)	TF-IDF (Trigram)	TF-IDF (Unigram, No Stop Words)	Word2Vec	Doc2Vec
RF	0.65	0.65	0.64	0.64	0.69	0.63	0.59
DT	0.57	0.59	0.58	0.6	0.63	0.54	0.56
MLP	0.71	0.7	0.69	0.69	0.69	0.71	0.57
SVM	0.45	0.73	0.73	0.71	0.71	0.71	0.63
LR	0.71	0.7	0.69	0.68	0.69	0.71	0.6
KNN	0.49	0.62	0.62	0.62	0.61	0.67	0.55
GP	0.71	0.71	0.72	0.7	0.7	0.72	0.61
AB	0.63	0.62	0.63	0.62	0.63	0.63	0.59
GNB	0.5	0.51	0.49	0.47	0.5	0.58	0.51
MNB	0.67	0.63	0.66	0.65	0.64	0.41	0.44
BNB	0.72	0.72	0.67	0.64	0.71	0.58	0.55
QDA	0.23	0.19	0.19	0.48	0.23	0.55	0.61
LDA	0.62	0.62	0.62	0.6	0.6	0.69	0.66

(25) found that the SVM Classifier outperformed the MNB Classifier on longer form text so perhaps the character count increase means that tweets are now more similar to longer form text. This warrants further investigation in future work.

The next best performing classifiers were the GP Classifier and the BNB Classifier, with f1-scores of 0.72. This somewhat agrees with Bermingham and Smeaton, who found that the MNB Classifier achieved the best classification results on their set of tweets. I have not come across any examples of the GP Classifier performing well for tweet classification.

4.3.4 Accuracy

The classifier with the highest accuracy score was also the SVM classifier with an accuracy score of 0.744 (Table 4.6). This indicates it had the highest percentage of correct predictions. The next best performing classifier was the GP classifier.

Overall considering the four classification metrics evaluated, the best performing classifier was the SVM Classifier, achieving the highest score in three out of four of the metrics, all but precision. The SVM classifier has a good balance of precision and recall, achieving the highest f1-score, and also had the highest accuracy score of all the classifiers.

A possible reason that SVM achieves the top performance is that the SVM classifier tends to be effective in high dimensional feature spaces. Twitter data produces a large feature set making this property of the SVM classifier useful. Another property of the SVM classifier

Table 4.6: Accuracy of Classifiers for different Feature Representations.

	BOW	TF-IDF (Unigram)	TF-IDF (Bigram)	TF-IDF (Trigram)	TF-IDF (Unigram, No Stop Words)	Word2Vec	Doc2Vec
RF	0.703	0.706	0.696	0.691	0.721	0.677	0.635
DT	0.614	0.596	0.614	0.621	0.652	0.563	0.594
MLP	0.719	0.719	0.711	0.711	0.709	0.718	0.621
SVM	0.594	0.744	0.737	0.718	0.724	0.716	0.637
LR	0.724	0.721	0.709	0.696	0.713	0.722	0.639
KNN	0.608	0.655	0.657	0.652	0.649	0.681	0.609
GP	0.726	0.726	0.724	0.706	0.708	0.729	0.644
AB	0.654	0.644	0.650	0.650	0.655	0.637	0.609
GNB	0.478	0.483	0.473	0.448	0.479	0.565	0.486
MNB	0.660	0.683	0.691	0.675	0.678	0.569	0.588
BNB	0.713	0.713	0.662	0.621	0.709	0.565	0.537
QDA	0.263	0.235	0.238	0.558	0.261	0.612	0.678
LDA	0.609	0.614	0.609	0.588	0.591	0.693	0.665

is that they are still effective in cases where the number of dimensions is greater than the number of samples. The number of dimensions in this case, is not higher than the number of samples. However, our number of samples is relatively low.

4.3.5 Feature Representation

Different types of feature representation were explored to try and improve the performance of the classifiers. This section will outline the changes in performance produced by these feature representation methods.

TF-IDF

TF-IDF did not have a massive impact on the performance of the majority of the classifiers in comparison to just BOW. It improved performance in some cases and worsened performance in others. TF-IDF and BOW achieved roughly the same scores. TF-IDF did however significantly improve the classification precision, recall, accuracy and f1-scores of the SVM classifier. These scores saw an increase from about 0.5 to about 0.7.

TF-IDF does not conclusively improve tweet classification performance. This may be due to the short length of the tweets. The short text is likely to have noisy TF-IDF values whereas the binary occurrence information obtained through plain BOW is more stable.

N-Grams

Three different N-gram feature sets were evaluated: unigrams, unigrams and bigrams, unigrams and bigrams and trigrams. We found that increasing the number of N-grams did not improve the classification precision, accuracy, recall or f1-score. In most cases, it actually worsened the performance of the classifiers or had little or no effect. This agrees with the findings of Bermingham and Smeaton (25) who found that expanding N-grams did not help the classification performance of either the tweets or the short reviews they experimented with them on.

Word2Vec and Doc2Vec

Word2Vec achieved better results in classifying the tweets than Doc2Vec. This could be because Word2Vec used Google's pretrained model. This model was trained on about 100 billion words from a Google News dataset. In comparison, the Doc2Vec model was trained on our significantly smaller dataset of 3116 tweets.

Neither Doc2Vec or Word2Vec achieved any better performance than BOW or TF-IDF. Like BOW, Word2Vec also loses the relationships between words when documents are converted to numerical vectors. This may be a reason why Word2Vec performs no better than BOW.

Stop Words

The unigram TF-IDF feature representation was implemented with and without stop words. The implementation with stop words included, performed better in nine out of the thirteen classifiers. This indicates that stop word removal did not improve classification performance. TF-IDF itself reflects how important a term is within a collection. It may be that TF-IDF is already accounting for the stop words high frequency, so no stop word removal is required.

The best performing approach was a combination of the SVM Classifier and unigram TF-IDF feature representation, closely followed by bigram TF-IDF. TF-IDF balances out how important a term is to a document compared with how important it is to the entire dataset. It reduces the weighting of less important terms that occur in many documents in the dataset. TF-IDF seems to perform particularly well in combination with the SVM Classifier. TF-IDF significantly improved the performance of the SVM Classifier.

4.4 Sentiment Scores

The Stanford NLP Sentiment Analyser was used to generate sentiment scores for the tweets that were classified as reviews. They were aggregated and normalised, as described in chapter 3, to produce an overall sentiment score for each hotel. Each score lies between zero and one, with one being the most positive score and zero being the most negative score. The sentiment scores of the top ten highest scoring hotels (Figure 4.1) and the top ten lowest scoring hotels (Figure 4.2) are presented below.

Figure 4.1: Hotels with Highest Sentiment Scores.

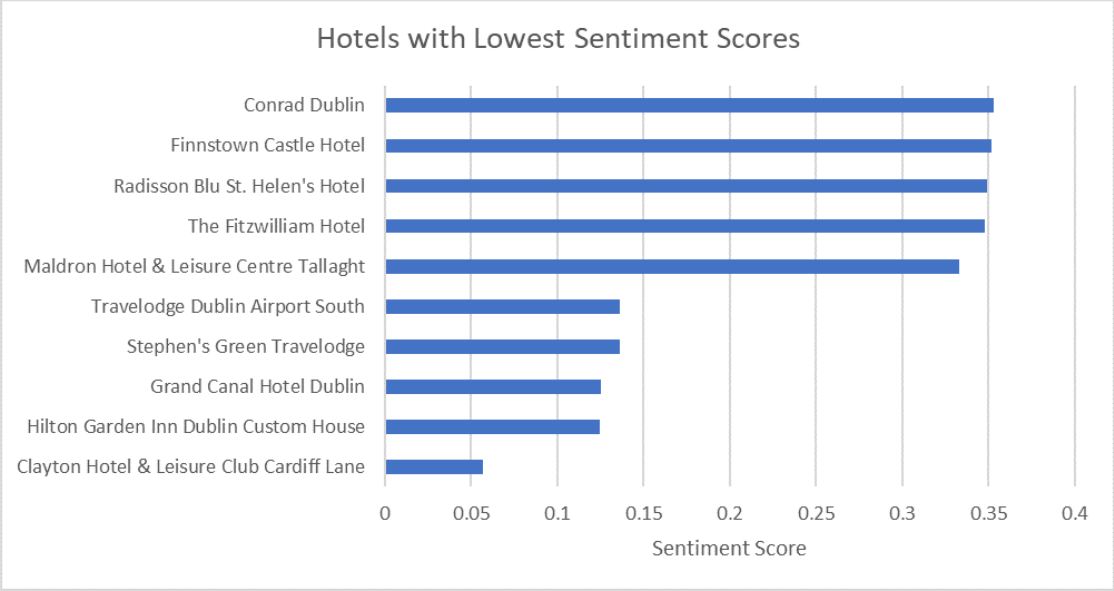

Figure 4.2: Hotels with Lowest Sentiment Scores.

4.5 Evaluation Metrics for Recommender Systems

4.5.1 Leave-One-Out Cross Validation

The 'leave-one-out' cross validation approach involves removing one of a users n hotel bookings and using their remaining hotel bookings to generate the hotel recommendations. A list of hotels ranked in descending order by their predicted value is produced. The booking that was removed is compared to this list to see where it ranked, the higher the better. This is repeated with each of the users n bookings.

The 'leave-one-out' approach was used because of the data that we have. We do not have any explicit user reactions to hotel recommendations but we do have implicit feedback in the form of user hotel bookings.

4.5.2 Mean Percentile Rank

The Mean Percentile Rank (MPR) is calculated based on the rankings recorded through leave-one-out cross validation. MPR is a recall-based measure. It measures the user satisfaction of items in an ordered list. The MPR formula is as follows:

$$MPR = \frac{\sum_{u,i} r_{ui} \times rank_{ui}}{\sum_{u,i} r_{ui}} \tag{5}$$

$rank_{ui}$ is the percentile-ranking of hotel i within the ordered list of all hotels ranked for user u. r_{ui} is a binary variable indicating whether user u booked hotel i. $rank_{ui} = 100\%$ indicates that the hotel i is predicted to be less desirable for user u. $rank_{ui} = 0\%$ indicates that the hotel i is predicted to be the most desirable hotel for user u. $rank_{ui} = 50\%$ would be expected for a list of randomly ranked hotels.

4.6 Evaluation of Added Sentiment Scores

4.6.1 User Data

The same Ryanair data that was used to evaluate the CoRE recommender was used for our evaluation. The Ryanair dataset consists of 29,704 hotel bookings, 11,638 unique hotels that have a least 1 booking and 20,223 users who made a booking at one or more hotels.

This project focused on hotels in Dublin so the dataset had to be filtered. The dataset was first filtered so that it only contained users with multiple hotel bookings. Multiple hotel bookings are required for the 'leave-one-out' approach so that a booking can be removed while still leaving at least one booking to use for generating the recommendations. Then the dataset was filtered so that it only contained users who had a booking in one of the Dublin hotels that we had produced a sentiment score for. The 'leave-one-out' approach had to be altered slightly so that the user's Dublin hotel booking was always the booking that was removed. The recommendations could be generated based on the other bookings but the removed booking had to be from Dublin so that we could evaluate where it ranked in the returned list.

There were 27 users who had multiple hotel bookings with at least one hotel booking in a Dublin hotel that we had a sentiment score for. The Dublin hotel bookings were for 21 different hotels. The evaluation was performed on these 27 users and 21 hotels. This is quite a small number of users which needs to be taken into consideration in the evaluation. Ideally, we would like to evaluate the system on a much larger set of users and hotels, to verify our results. This is an area for future work.

4.6.2 Baselines

In order to assess the performance of SentiCoRE (CoRE with the added sentiment information), it was compared against multiple baselines. These included:

1. **Random**
   A randomly ranked list of hotels.
2. **Expedia Baseline**
   The approach currently used by Ryanair in their rooms booking site. Hotels from the target city are sorted based on a sequence number given to them by Expedia. The sequence number is based on their transactional data from the last 30 days.
3. **CoRE**
   The CoRE recommender system without the sentiment score added in.
4. **SentiCoRE**
   The CoRE recommender system re-ranked with the sentiment scores produced by the Stanford NLP Sentiment Analyser.

4.7 Recommender Results and Analysis

The results show that the addition of the sentiment score increases the MPR of CoRE, with and without feature weighting (Table 4.7), which means the hotels are being ranked lower by SentiCoRE than CoRE. The best performing recommender system is CoRE, achieving a MPR of 2.51%. CoRE performs slightly better with feature weighting than without feature weighting as was seen in the evaluation of CoRE by itself (5).

The Random Recommender which produces a random list of hotels performed the worst, which was expected. Typically a MPR of 50% would be expected for a random list of hotels which is not far off the Random Recommender's MPR of 42.72%. The Expedia Recommender only performed slightly better than the Random Recommender with a MPR of 40.87. Both CoRE and SentiCoRE performed significantly better than the Random and Expedia baselines.

Table 4.7: Recommender Systems Performance.

Recommender System	**MPR (%)**
Random	42.72
Expedia Baseline	40.87
CoRE (with feature weighting)	2.51
CoRE (without feature weighting)	3.31
SentiCoRE (with feature weighting)	14.68
SentiCoRE (without feature weighting)	9.92

From the results, we can see that incorporating the sentiment scores into the CoRE recommender definitely has an impact on the hotel rankings. A positive sentiment score will move a hotel up the list while a negative ranking will move a hotel down the list. Taking only the MPR into account you would say that SentiCoRE is not performing as well as CoRE. However, SentiCoRE is having the desired effect and is adjusting the hotel rankings based on the sentiment scores.

4.8 Summary

One of the main aims of this research was to determine if it was possible to identify whether tweets contained reviews. A series of classifiers with different feature representations were implemented and evaluated.

The best performing approach was a combination of the SVM Classifier and unigram TF-IDF feature representation, closely followed by bigram TF-IDF.

The other aim of this research was to evaluate if review-like tweets could be used to appropriately influence the generation of recommendations in a recommender system. Sentiment scores were produced based on the classified tweets and used to re-rank the results of the CoRE recommender system. Incorporating the sentiment scores into the CoRE recommender had the desired effect and adjusted the rankings of the hotels. However, SentiCoRE performed worse than CoRE in terms of MPR.

5 Conclusion

5.1 Summary of Results

In this dissertation, two problems were addressed. The first was to assess if it was possible to classify whether a tweet contains content which could be deemed to be a review. The second was to determine to what extent Twitter can provide a suitable source of online reviews that can be used effectively in the generation of recommendations in a recommender system.

Tweets were collected through the Twitter Streaming API. They were then filtered so that they only contained tweets about hotels posted from Dublin. Once the dataset was filtered it had to be manually annotated so that it could be used to train a series of classification algorithms. A webpage was built to facilitate this annotation process. The tweets were labelled as 'Review', 'Some Content' or 'Irrelevant'. Thirteen different classifiers and seven different feature representations were evaluated. The classifiers were implemented using Python's Scikit Learn library.

We evaluated the performance of these classifiers. The best performing classifier was the Support Vector Machine Classifier. It achieved a precision score of 74%, a recall score of 74%, an f1-score of 73% and an accuracy score of 74.4%. The best performing feature representation was unigram TF-IDF. These results confirm that text classification is a valid method of extracting reviews from Twitter.

Sentiment analysis was performed on the tweets that were classified as reviews, using the Stanford NLP Sentiment Analyser. The sentiment scores produced were used to re-rank the results of the CoRE recommender system. The performance of the CoRE recommender system with the sentiment scores was evaluated against the original CoRE recommender and two further baseline systems.

Incorporating the sentiment scores into the CoRE recommender had the desired effect and adjusted the rankings of the hotels. But in terms of MPR SentiCoRE performed worse than CoRE. The size of the dataset needs to be considered when analysing these results. These

results need to be verified with a larger dataset.

5.2 Future Work

Several promising lines of future work have been identified during the course of the research described in this dissertation.

One line of future work would be to improve the classification performance of the machine learning algorithms. An accuracy of 74.4% was achieved with the SVM Classifier. There are definitely improvements to be made here. This could be achieved by:

- Increasing the size of the dataset used for training the classifiers. In general, supervised machine learning methods perform better on large datasets. This would involve gathering more tweets and manually annotating them.
- Investigating other feature representations. BOW, TFIDF, N-Grams, Word2Vec and Doc2Vec were evaluated. Other feature representations like Part-of-Speech tagging and other NLP techniques could be experimented with.
- More classification algorithms could be implemented. Classification algorithms such as Stochastic Gradient Descent and ensemble classification algorithms such as Gradient Tree Boosting and Extremely Randomised Trees which were not evaluated in this research could be investigated.

A second line of future work would be to expand the scope of the project. This research focused on hotels in Dublin. This scope could be expanded first geographically. It could then be expanded to different fields. Twitter could be used to gain sentiment information about restaurants, movies etc. and applied to recommender systems relating to these fields.

A possible improvement to the methodology outlined in this project would be re-training the Stanford NLP Sentiment Analyser. The analyser was trained on a set of movie reviews from Rotten Tomatoes. Due to this, it will perform best in classifying texts similar to these movie reviews, rather than tweets which have a very different format. It would be interesting to evaluate whether re-training the analyser using review-like tweets would improve how well it classifies the sentiment of the tweets.

Another important line of future work would be to evaluate the performance of the recommender system on a larger dataset. In this project, the system was evaluated for 27 users and 21 hotels. Our results could be confirmed and would hold more weight if the recommender was re-evaluated with data for more users and hotels.

Finally, the proposed methodology could be improved by experimenting with prioritising the sentiment of more recent Twitter reviews or personalising the process by prioritising reviews

from Twitter users similar to the user in question. The sentiment of hotels with a lot of reviews could also be prioritised. For example a hotel with lots of positive reviews being boosted higher than a hotel with just one positive review.

5.3 Final Thoughts

Overall this dissertation has investigated whether it is possible to extract reviews from Twitter and to what extent Twitter can provide a suitable source of online reviews that can be used effectively in the generation of recommendations in a recommender system. The results showed that text classification is a valid method of extracting reviews from Twitter and that incorporating the reviews from Twitter into a recommender system adjusted the hotel's rankings. The methodology proposed in this research requires further verification on a larger dataset but I think that this area of research has a lot of potential.

Over the course of writing this dissertation, I have learnt a lot. I have learnt how to research topics and find relevant information. In the beginning, I found it quite difficult to sift through research papers and extract the information relevant to my research. I have also learnt lots about the topics discussed in this dissertation, data processing and filtering, machine learning algorithms and recommender systems.

Bibliography

[1] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[3] A. Rane and A. Kumar. Sentiment classification system of twitter data for us airline service analysis. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 769–773, 2018. ISBN 0730-3157.

[4] Twitter Investor Relations. Q4 and fiscal year 2018 letter to shareholders, February 2019. URL `https://s22.q4cdn.com/826641620/files/doc_financials/2018/q4/Q4-2018-Shareholder-Letter.pdf`.

[5] Matthew Nicholson Anirban Chakraborty Mostafa Bayomi, Annalina Caputo and Séamus Lawless. Core: A cold-start resistant and extensible recommender system. Trinity College Dublin, 8-12 April 2019.

[6] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[7] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer science & business media, 1995. ISBN 0-387-94559-8.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995. ISSN 0885-6125.

[9] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, volume 752, pages 41–48. Citeseer, 1998.

[10] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125.

[11] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[12] Alexander Genkin, David D Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

[13] Eui-Hong Sam Han, George Karypis, and Vipin Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-asia conference on knowledge discovery and data mining*, pages 53–65. Springer, 2001.

[14] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7): 881–892, 2002.

[15] Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[16] Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. A review of machine learning algorithms for text-documents classification. *Journal of advances in information technology*, 1(1):4–20, 2010.

[17] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, pages 137–142. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-69781-7.

[18] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[19] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[20] Andrew MacCallum Kamal Nigam, John Lafferty. Using maximum entropy for text classification. In *Workshop on Machine Learning for Information Filtering*, pages 61–67. IJCAI, 1999.

[21] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[22] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[23] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[24] Andronicus A Akinyelu and Aderemi O Adewumi. Classification of phishing email using random forest machine learning technique. *Journal of Applied Mathematics*, 2014, 2014.

[25] Alan F. Bermingham, Adam Smeaton. Classifying sentiment in microblogs: is brevity an advantage?, 2010. URL `http://doi.acm.org/10.1145/1871437.1871741`.

[26] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[27] Ankit and Nabizath Saleena. An ensemble classification system for twitter sentiment analysis. *Procedia Computer Science*, 132:937–946, 2018. ISSN 1877-0509.

[28] M. Kanakaraj and R. M. R. Guddeti. Performance analysis of ensemble methods on twitter sentiment analysis using nlp techniques. In *Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015)*, pages 169–170, 2015. doi: 10.1109/ICOSC.2015.7050801.

[29] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[30] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM, 2010. ISBN 1450301533.

[31] M. Rathi, A. Malik, D. Varshney, R. Sharma, and S. Mendiratta. Sentiment analysis of tweets using machine learning approach. In *2018 Eleventh International Conference on Contemporary Computing (IC3)*, pages 1–3, 2018. ISBN 2572-6129. doi: 10.1109/IC3.2018.8530517. URL `https://ieeexplore.ieee.org/document/8530517/`.

[32] S. Bhuta, A. Doshi, U. Doshi, and M. Narvekar. A review of techniques for sentiment analysis of twitter data. In *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pages 583–591, 2014. doi: 10.1109/ICICICT.2014.6781346.

[33] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems: Introduction and Challenges*, pages 1–34. Springer US, Boston, MA, 2015. ISBN 978-1-4899-7637-6. doi: 10.1007/978-1-4899-7637-6_1. URL https://doi.org/10.1007/978-1-4899-7637-6_1.

[34] Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. Finding a needle in a haystack of reviews: Cold start context-based hotel recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 115–122. ACM, 2012.

[35] K. Lin, C. Lai, P. Chen, and S. Hwang. Personalized hotel recommendation using text mining and mobile browsing tracking. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 191–196, 2015. doi: 10.1109/SMC.2015.46.

[36] J. Chang, C. Tsai, and J. Chiang. Using heterogeneous social media as auxiliary information to improve hotel recommendation performance. *IEEE Access*, 6:42647–42660, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2855690.

[37] T. Takehara, S. Miki, N. Nitta, and N. Babaguchi. Extracting context information from microblog based on analysis of online reviews. In *2012 IEEE International Conference on Multimedia & Expo Workshops (ICMEW 2012), 9-13 July 2012*, 2012 IEEE International Conference on Multimedia Expo Workshops (ICMEW 2012), pages 248–53. IEEE Computer Society, 2012. doi: 10.1109/ICMEW.2012.49. URL http://dx.doi.org/10.1109/ICMEW.2012.49.

[38] emoji, 2018. URL https://pypi.org/project/emoji/.

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

[40] CE Rasmussen. Cki williams gaussian processes for machine learning, 2006.

[41] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55 (1):119–139, 1997.

[42] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

A1 Appendix

Table A1.1: Hotel Names and Twitter Handles.

Hotel Name	Twitter Handle
abberley hotel	
abbey hotel	
aberdeen lodge	
academy plaza hotel	@academyplaza
accor hotels	@accorhotels
albany house	@albanyhouse
alex hotel	@thealexdublin
alexander hotel	
ariel house	@arielhouse
arlington hotel	@arlingtondublin
ashling hotel	@ashlinghotel
aspect park west	@aspectparkwest
avoca house	@avocahouse
avondale house	
baggot court townhouse	
ballsbridge hotel	@bbhoteldublin
barrys hotel	
beacon hotel	@thebeaconhotel
belvedere hotel	@belvederparnell
beresford hotel	@beresfordhoteli
blooms hotel	@bloomshotel
bonnington dublin	
brooks hotel	@brookshoteld2
buswells hotel	@buswellshotel
butlers townhouse	@butlersthd4
camden court hotel	@camdencourt
carlton hotels	@carltonhotels
carnegie court	@carncourthotel
cassidys hotel	@cassidys_hotel
castle hotel	@thecastlehotel
castleknock hotel	@cknock_hotel
caulfields hotel	

Hotel Name	Twitter Handle
celtic lodge	@celticlodgedub
central inn	
charleville lodge	@clodgehotel
city hotel	
citywest hotel	@citywesthotel
clarence hotel	@theclarence
clarion hotel	@clarionnorden
clayton dublin airport	@claytoncarpark
clayton hotels	@claytonhotels
cliff townhouse	@clifftownhouse
clifton court hotel	
clontarf castle	@clontarfcastle
cloverleaf suites	@cloverleafrs
clyde court hotel	
conrad dublin	@conraddublin
croke park hotel	@crokeparkhotel
crowne plaza	@crowneplazablan
dawson hotel	
dean dublin	@dean_dublin
dergvale hotel	
devlin dublin	@thedevlindublin
double tree	@doubletreedub
drury court hotel	@drurycourthotel
dublin city inn/dublin central inn	@dublincityinn
dylan hotel	@dylanhotel
ferryman townhouse	
finnstown castle hotel	@finnstowndublin
fitzpatrick castle hotel	@fitzcastle
fitzsimons hotel	
fitzwilliam hotel	@fitzwilliamdub
fleet street hotel	@fleetsthotel
gate hotel	
generator hostel	@gen_dublin
gibson hotel	@thegibsonhotel
glashaus hotel	
gleann na smol	
glen guesthouse	
grafton capital	
grafton guesthouse	@ghousedublin
grand canal hotel	@grandcanalhotel
grand hotel	@grandmalahide
green isle hotel	@greenislehotel
gresham hotel	@greshamhotel

Hotel Name	Twitter Handle
hampton hotel	@hamptondublin
harcourt hotel	
harding hotel	
harrington hall	
herbert park hotel	@herbertpark
hilton dublin	@hiltondub
hilton garden	@hgidublin
holiday inn	@hiexpress
hotel st. george by the key collection	
house dublin 2	@housedublin2
intercontinental	@intercondublin
isaacs hostel	@isaacsdublin
iveagh garden	@iveaghgardenhot
jackson court	@jacksoncourtdub
jacobs inn	@jacobsinndublin
jurys inn	@jurysinnsdublin
key collection	@collectionkey
kildare street hotel	
kingswood hotel citywest	@kingswooddublin
lansdowne hotel	@lansdownehotel1
leeson inn downtown	
louis fitzgerald	@louisfitzhotel
lynams hotel	
maldron hotel parnell square	
maldron hotels	@maldronhotels
maple hotel	@maplehotel
marine hotel	@marinehotel
marker hotel	@themarkerhotel
mercantile	@the_mercantile
merrion hotel	@merrionhotel
mespil hotel	@mespilhotel
metro hotel	@metrohoteldub
metrostays	
morehampton	@morehampton_
morgan hotel	@themorganhotel
morrison hotel	@morrisondublin
my place hotel	@myplacehotels
north star hotel	@northstar_hotel
o'callaghan collection	@ochdublin
o'callaghan stephens green	
o'shea's hotel	@hotelosheas
paramount hotel	
parliament hotel	@paramounthotel

Hotel Name	Twitter Handle
pembroke townhouse	@pembroketownhse
phoenix park hotel	
plaza hotel	@plazatallaght
portobello hotel	
premier suites	@premiersuiteseu
radisson blu	@radissondublin
ranelagh rooms	@ranelaghrooms
red cow moran hotel	@moranhotels
regency hotel	
ripley court hotel	
river house	
roganstown hotel & country club	
roxford lodge	@roganstown
royal hotel and merrill leisure club	@royalhotelbray
royal marine	@royalmarine
russell court hotel	
sandymount hotel	@sandymounthotel
schoolhouse hotel	
shelbourne hotel	@theshelbourne
sheldon park hotel and leisure club	@sheldonpark_
sky backpackers	
skylon hotel	@skylondublin
spencer hotel	@thespencerifsc
stauntons on the green	@stauntons_
staycity	@staycity
sweeney hotel	
talbot hotel	@talbotstill
tara towers hotel	@taratowershotel
temple bar hotel	@templebarhotel
temple bar inn	@templebarinn
times hostels	@timeshostels
travelodge	@travelodge_ire
travelodge dublin airport south	
trinity city hotel	@trinity_city
waterloo house	
west county hotel	@westcountyhotel
westbury hotel	@westburydublin
westin dublin	
white sands hotel	@whitesandshotel
wingate hibernian	
wynn's hotel	@wynnshotel

Table A1.2: Tweets Collected Per Hotel.

Hotel Name	Review	Content	Irrelevant	Total
Abberley Court Hotel	0	0	10	10
Academy Plaza Hotel	0	2	2	4
Ariel House	1	3	11	15
Arlington Hotel O'Connell Bridge	2	4	21	27
Ashling Hotel Dublin	3	3	2	8
Ballsbridge Hotel	13	24	15	52
Beresford Hotel	1	0	1	2
Blooms Hotel	0	0	0	0
Brooks Hotel	4	6	4	14
Buswells Hotel	0	11	29	40
Camden Court Hotel	0	1	0	1
Carlton Hotels	0	0	1	1
Castle Hotel	2	0	1	3
Castleknock Hotel & Country Club	2	5	9	16
Charleville Lodge Hotel	0	1	2	3
Citywest Hotel	15	50	27	92
Clayton Dublin Airport	3	4	0	7
Clayton Hotel & Leisure Club Cardiff Lane	2	0	0	2
Clayton Hotel Ballsbridge	0	1	3	4
Clayton Hotel Burlington Road	3	5	3	11
Clayton Hotel Leopardstown	0	6	1	7
Clayton Hotels	2	6	20	28
Cliff Townhouse	6	10	8	24
Clontarf Castle Hotel	19	16	20	55
Conrad Dublin	6	29	60	95
Crowne Plaza Dublin Airport	1	2	1	4
Crowne Plaza Hotel Dublin Blanchardstown	0	4	1	5
Double Tree	0	3	2	5
Dublin Skylon Hotel	4	4	3	11
Dylan Hotel	11	11	8	30
Finnstown Castle Hotel	5	5	2	12
Fitzpatrick Castle Hotel	7	6	5	18
Generator Hostel Dublin	3	16	53	72
Grand Canal Hotel Dublin	2	0	1	3
Grand Hotel	1	0	5	6
Green Isle Conference & Leisure Hotel	3	15	6	24
Guest House	0	3	7	10

Hotel Name	Review	Content	Irrelevant	Total
Herbert Park Hotel	11	3	24	38
Hilton Dublin Airport Hotel	0	0	14	14
Hilton Dublin Kilmainham	7	4	8	19
Hilton Garden Inn	9	7	181	197
Hilton Garden Inn Dublin Custom House	1	2	7	10
Hilton Hotel	9	12	51	72
Holiday Inn	3	0	28	31
Hotel 7 Dublin	1	0	3	4
Hotel Riu Plaza The Gresham Dublin	1	7	6	14
House Dublin	26	32	57	115
Ibis Dublin	0	0	1	1
InterContinental Dublin	41	76	100	217
Isaacs Hostel	0	3	7	10
Iveagh Garden Hotel	3	7	2	12
Jacobs Inn - Hostel	6	0	2	8
Jurys Inn Hotels	8	6	46	60
Lansdowne Hotel	1	1	4	6
Maldron Hotel	3	7	14	24
Maldron Hotel & Leisure Centre Tallaght	0	4	3	7
Maldron Hotel Kevin Street	0	3	4	7
Maldron Hotel Newlands Cross	0	0	4	4
Maldron Hotel Parnell Square	0	0	1	1
Maldron Hotel Pearse Street	0	0	2	2
Maldron Hotel Smithfield	0	0	2	2
Maple House Hotel	0	0	1	1
Marine Hotel	1	1	1	3
Mespil Hotel	2	3	2	7
Morehampton Townhouse Dublin	0	0	27	27
North Star Hotel	5	1	10	16
O'Callaghan Collection	2	1	7	10
O'Callaghan Davenport Hotel	0	1	1	2
O'Sheas Hotel	1	2	5	8
Pembroke Townhouse	2	0	2	4
Radisson Blu	8	15	24	47
Radisson Blu Dublin Airport	0	0	1	1
Radisson Blu Royal Hotel, Dublin	6	33	19	58
Radisson Blu St. Helen's Hotel	12	5	20	37
Ranelagh Rooms	0	0	3	3
Red Cow Moran Hotel	3	1	5	9
Roganstown Hotel & Country Club	6	4	10	20
Sandymount Hotel	1	2	8	11

Hotel Name	Review	Content	Irrelevant	Total
Schoolhouse Hotel	0	0	6	6
Staycity	2	2	19	23
Talbot Hotel Stillorgan	2	14	9	25
Tara Towers Hotel	2	0	4	6
Temple Bar Hotel	0	1	1	2
Temple Bar Inn	2	0	7	9
The Alex	3	0	12	15
The Beacon	1	1	1	3
The Belvedere Hotel Parnell Square	1	0	0	1
The Clarence	1	1	6	8
The Croke Park Hotel	6	13	19	38
The Dawson Hotel	0	1	0	1
The Dean Dublin	12	15	29	56
The Fitzwilliam Hotel	3	10	14	27
The Gibson Hotel	15	30	17	62
The Louis Fitzgerald Hotel	0	10	1	11
The Marker Hotel	42	39	41	122
The Mercantile Hotel	0	3	7	10
The Merrion	33	44	31	108
The Metro Hotel Dublin Airport	2	10	5	17
The Morgan Hotel	0	0	1	1
The Morrison, a DoubleTree by Hilton Hotel	12	20	19	51
The Plaza Hotel	0	0	2	2
The Royal Hotel and Merrill Leisure Club	1	7	0	8
The Royal Marine Dublin	6	15	19	40
The Shelbourne Dublin, A Renaissance Hotel	44	87	168	299
The Spencer Hotel Dublin City	0	0	1	1
The Westbury Hotel	58	65	78	201
The Westin Dublin	12	19	41	72
Travelodge	1	1	12	14
Travelodge Dublin Airport	0	0	1	1
Trinity City Hotel	5	0	5	10
White Sands Hotel	0	0	1	1
Wynn's Hotel	0	3	0	3