



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

# Temporal Opinion Mining

**Thomas Hoberg Burnett**  
**Eivind Bjørkelund**

Master of Science in Computer Science

Submission date: June 2012

Supervisor: Kjetil Nørvåg, IDI

Norwegian University of Science and Technology  
Department of Computer and Information Science



# Abstract

This project explores the possibilities in detecting changes in opinion over time. For this purpose, different techniques and algorithms in opinion mining have been studied and used as a theoretic foundation when developing strategies towards detecting changes in opinions.

Different approaches to a system that detects and visualises changes in opinions have been proposed. These approaches include using machine learning techniques like the naive Bayes algorithm and opinion mining techniques based on SentiWordNet. Additionally, feature extraction techniques and the impact of burst detection have been studied.

During this project, experiments have been carried out in order to test some of the techniques and algorithms. A data set containing hotel reviews and a prototype have been built for this purpose, allowing easy support for testing and validation. Results found high accuracy in opinion mining with the lexicon SentiWordNet, and the prototype can detect hotel features and possible reasons for changes in opinion. It can also show "good" and "bad" geographical areas based on hotel reviews.

For commercial use, the prototype can help analyse the massive amount of hotel information published each day by customers, and can help hotel managers analyse their products. It can also be used as a more advanced hotel search engine where users can find extra information in a map user interface.

**Keywords:** *Semantical analysis, Opinion mining, Feature extraction, Temporal opinion mining.*



# Sammendrag

Dette prosjektet utforsker mulighetene til å detektere forandringer i meninger over tid. Forskjellige teknikker og algoritmer innen meningsgraving har blitt studert og brukt som et teoretisk grunnlag til å utvikle strategier innen deteksjon av meningsforandringer.

Forskjellige metoder til et system som detekterer og visualiserer meningsforandringer har blitt foreslått. Disse metodene inkluderer maskinlæringsteknikker som "naive Bayes"-algoritmen og meningsgraving basert på SentiWordNet. I tillegg har egenskapsuttrekkingsteknikker og utbruddsdeteksjon blitt studert.

I løpet av dette prosjektet har eksperimenter blitt utført for å teste noen av disse teknikkene og algoritmene. Et datasett som inneholder hotellanmeldelser og en prototype har blitt bygget for å muliggjøre dette. Resultater viser høy korrekthet i meningsgraving med SentiWordNet-biblioteket, og prototypen kan detektere hotellegenskaper og mulige grunner for meningsforandringer. Den kan også vise "gode" og "dårlige" geografiske områder basert på hotellanmeldelser.

For kommersielt bruk vil denne prototypen kunne hjelpe å analysere den massive mengden av hotellinformasjon publisert hver dag av kunder, og den kan hjelpe hotellsjefer til å analysere deres produkt. Den kan også bli brukt som en avansert hotellsøkemotor, hvor brukere kan finne ekstra informasjon i et kart-grensesnitt.

Nøkkelord: *Semantisk analyse, meningsgraving, egenskapsuttrekking, midlertidig meningsgraving.*



# Preface

This report has been written by Thomas Hoberg Burnett and Eivind Bjørkelund as part of the master's thesis at the Norwegian University of Science and Technology (NTNU) during the spring of 2012. The master's thesis concludes our five year integrated study at the Department of Computer and Information Science (IDI). The purpose of the project is to study techniques and algorithms in temporal opinion mining, in an attempt to detect and visualise changes in opinion with a focus on hotel reviews.

We would like to thank our teaching supervisor, Kjetil Nørvåg (Professor) for his continuous support and constructive criticism throughout the project period. We really appreciate the opportunity to work with him on such an interesting subject.

Trondheim, 12th of June 2012

-----  
Thomas Hoberg Burnett

-----  
Eivind Bjørkelund





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Problem Definition and Goals . . . . .	2
1.3	Contributions . . . . .	3
1.4	Project Scope . . . . .	4
1.5	Report Outline . . . . .	5
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	Opinion Mining . . . . .	7
2.1.1	History . . . . .	8
2.1.2	Sentiment classification . . . . .	8
2.1.3	Previous Work . . . . .	9
2.1.4	Feature-based Opinion Mining and Summarisation . . . . .	11
2.1.5	Binary versus Multi-Polarity Analysis . . . . .	12
2.2	Temporal Opinion Mining . . . . .	13
2.2.1	Temporality Process . . . . .	13
2.2.2	Previous Work . . . . .	14
2.2.3	Opinion Spam . . . . .	15
<b>3</b>	<b>Text Analysis</b>	<b>17</b>
3.1	Part of Speech Tagging . . . . .	17
3.1.1	Introduction . . . . .	18
3.1.2	Tagging Approaches . . . . .	18
3.1.3	Part of Speech Tagging in Opinion Mining . . . . .	21
3.1.4	Part of Speech Tagging in our Application . . . . .	21
3.2	WordNet . . . . .	22
3.3	Word Sense Disambiguation . . . . .	22
3.3.1	Introduction . . . . .	24
3.3.2	Lesk Algorithm . . . . .	25
3.3.3	Usage in our Application . . . . .	25
<b>4</b>	<b>Opinion Mining Techniques</b>	<b>27</b>
4.1	Known Challenges . . . . .	27

4.2	Knowledge-based Approaches . . . . .	28
4.2.1	SentiWordNet . . . . .	29
4.2.2	SentiWordNet in our Application . . . . .	30
4.3	Supervised Approaches . . . . .	31
4.3.1	Naive Bayes . . . . .	31
4.4	Unsupervised Approaches . . . . .	33
4.4.1	Supervised versus Unsupervised . . . . .	34
4.5	Language Models . . . . .	34
4.5.1	N-Gram Models . . . . .	35
4.6	Feature Extraction and Summary . . . . .	36
4.7	Temporal Aspects . . . . .	38
4.7.1	Burst Detection . . . . .	38
4.7.2	Opinion Visualisation . . . . .	39
<b>5</b>	<b>Extracting Suitable Data Sets</b>	<b>41</b>
5.1	Web Crawling . . . . .	41
5.1.1	Crawling Process . . . . .	41
5.1.2	Sources . . . . .	43
5.1.3	Politeness Policy . . . . .	48
5.1.4	Performance . . . . .	48
5.1.5	Locations . . . . .	49
5.2	Storing the Data Set . . . . .	49
5.2.1	XML Structure . . . . .	49
5.3	Preparing Data for Future Use . . . . .	50
5.4	Problems, Experiences, Setbacks . . . . .	53
5.4.1	Connection Problems . . . . .	53
5.4.2	Crawling Aggression . . . . .	54
5.4.3	Updating the Data Sets . . . . .	54
5.4.4	File Size . . . . .	55
5.5	Merging Data Sets . . . . .	55
5.5.1	Hotel Matching . . . . .	55
5.5.2	Name and GPS Matching . . . . .	55
5.6	Statistics . . . . .	57
<b>6</b>	<b>Opinion Visualisation Prototype</b>	<b>61</b>
6.1	Opinion Mining Framework . . . . .	61
6.2	Map Prototype . . . . .	62
6.2.1	Data Set Preparation . . . . .	62
6.2.2	Prototype Description . . . . .	62
6.2.3	Google Maps . . . . .	65
6.2.4	Area Sentiment . . . . .	66
6.2.5	Graphing Changes . . . . .	68
6.2.6	Feature Search . . . . .	69
<b>7</b>	<b>Experiments and Results</b>	<b>73</b>

7.1	SentiWordNet Accuracy . . . . .	73
7.1.1	Standard SentiWordNet . . . . .	74
7.1.2	Simplified Lesk . . . . .	74
7.1.3	Five Categories . . . . .	76
7.2	Machine Learning . . . . .	77
7.3	Burst Detection . . . . .	77
7.4	Visualising Temporal Sentiment . . . . .	78
7.4.1	Temporal Sentiment . . . . .	78
7.4.2	Rating and Sentiment Correlation . . . . .	81
7.4.3	Area Sentiment . . . . .	82
7.4.4	Feature Search . . . . .	82
<b>8</b>	<b>Discussion and Conclusion</b>	<b>85</b>
8.1	Result Summary . . . . .	85
8.1.1	Burst Detection and Temporal Sentiment . . . . .	85
8.1.2	Feature Selection . . . . .	86
8.1.3	Sentiment Visualisation and Mapping . . . . .	87
8.1.4	Data Set Comparison . . . . .	87
8.2	Conclusion . . . . .	88
8.3	Future Work . . . . .	89
8.3.1	Prototype Performance . . . . .	89
8.3.2	Visualisation . . . . .	90
8.3.3	Feature Synonyms . . . . .	90
8.3.4	Expand Data Sets . . . . .	91
8.3.5	Other Pre-Processing Steps . . . . .	91
<b>A</b>	<b>List of Data Set Locations</b>	<b>93</b>
<b>B</b>	<b>Class Diagrams</b>	<b>95</b>
B.1	Package Overview . . . . .	95
B.2	Crawler . . . . .	96
B.3	Gatherer . . . . .	97
B.4	Output . . . . .	98
B.5	Processing . . . . .	99
B.6	Storers . . . . .	100
B.7	UI . . . . .	101



# List of Figures

1.1	Screenshot of the prototype . . . . .	4
1.2	The project approach . . . . .	5
2.1	A summary example [23] . . . . .	12
3.1	Raw text before POS tagging . . . . .	18
3.2	Text after POS tagging . . . . .	18
3.3	Example of "that" rule in rule-based tagging . . . . .	20
3.4	Entry of "dog" in WordNet . . . . .	24
4.1	Entry of "adventurous" in SentiWordNet . . . . .	29
4.2	An example of a data set with a cluster structure from [37] . . . . .	34
4.3	A review consisting of pros and cons . . . . .	36
4.4	A review consisting of free text . . . . .	37
4.5	Example of summarisation for the feature breakfast from our test prototype	38
4.6	Pseudo code of our burst detection algorithm . . . . .	39
5.1	High level view of system flow in TripAdvisor and Booking.com crawlers	42
5.2	Positive review from Booking.com . . . . .	43
5.3	Search area used for finding locations . . . . .	44
5.4	Subset of search results . . . . .	44
5.5	Presentation of hotel information . . . . .	45
5.6	Subset of reviews . . . . .	45
5.7	Search area used for finding locations and services . . . . .	46
5.8	Subset of search results . . . . .	47
5.9	Presentation of hotel information . . . . .	47
5.10	Subset of reviews . . . . .	47
5.11	XML format of data sets . . . . .	50
5.12	Example of XML format of data sets . . . . .	50
5.13	Calculated review score output . . . . .	52
5.14	Average scores . . . . .	53
5.15	Distribution of review scores . . . . .	57
5.16	Percentage of hotels for each amount of reviews interval . . . . .	58

5.17	Difference in number of reviews per location . . . . .	58
5.18	Score distribution for TripAdvisor and Booking.com hotels . . . . .	59
6.1	Command line menu for our opinion mining framework . . . . .	62
6.2	Map prototype layout . . . . .	63
6.3	Hotels in London from booking.com marked on map without area calculations. See figure 6.5 for marker descriptions. . . . .	64
6.4	Map marker info . . . . .	65
6.5	Map legend . . . . .	65
6.6	Control panel for the map prototype . . . . .	66
6.7	Hotels in London from booking.com marked on map with area calculations . . . . .	67
6.8	Area marked as relatively positive . . . . .	68
6.9	Month by month score changes . . . . .	69
6.10	XML format for feature scores . . . . .	70
6.11	Selected feature "internet" with negative sentiment . . . . .	71
6.12	Selected feature "internet" with positive sentiment . . . . .	71
7.1	Accuracy of the Lesk algorithm vs Standard SentiWordNet for TripAdvisor . . . . .	75
7.2	Accuracy of the Lesk algorithm vs Standard SentiWordNet for Booking.com . . . . .	76
7.3	Graph from 1891 reviews about Park Plaza Westminster Bridge in London, UK . . . . .	80
7.4	Graph from 4250 reviews about St Giles Hotel & Leisure Club in London, UK . . . . .	80
7.5	Graph from 40 reviews about San Domenico House in London, UK . . . . .	81
7.6	Area with a mostly positive sentiment . . . . .	82
7.7	Area with a mostly negative sentiment . . . . .	83
7.8	Area with no clear one-sided sentiment . . . . .	83
7.9	Overview of different sentiments in neighbouring areas . . . . .	84
8.1	Selected feature 'internet' with positive score, but clearly negative content . . . . .	87
8.2	XML score structure in prototype . . . . .	89
8.3	A possible improvement of XML score structure . . . . .	89
B.1	Package map with dependencies . . . . .	95
B.2	Classes in crawler package . . . . .	96
B.3	Classes in gatherer package . . . . .	97
B.4	Classes in output package . . . . .	98
B.5	Classes in processing package . . . . .	99
B.6	Classes in storers package . . . . .	100
B.7	Class in ui package . . . . .	101

# List of Tables

3.1	The Penn Treebank POS tag set . . . . .	19
3.2	Semantic relations in WordNet . . . . .	23
4.1	Sentiment classification with different keyword lists [48] . . . . .	30
5.1	Average crawler run time per hotel at TripAdvisor . . . . .	49
5.2	Average crawler run time per hotel at Booking.com . . . . .	49
5.3	XML tag explanation . . . . .	51
5.4	Number of services in each data set . . . . .	57
6.1	Circle colouring for area sentiment . . . . .	67
6.2	Percentage of reviews year by year . . . . .	69
6.3	List of features . . . . .	70
7.1	SentiWordNet accuracy results . . . . .	74
7.2	SentiWordNet accuracy results with a simplified Lesk Algorithm . . . . .	75
7.3	SentiWordNet accuracy results with five categories . . . . .	76
7.4	Result table for machine learning algorithms . . . . .	77
7.5	Result table for machine learning algorithms with five categories . . . . .	77
7.6	Burst detection results with different thresholds . . . . .	79
7.7	Features from Grand Midwest Express Hotel Apartments in Dubai from a positive burst month . . . . .	79
7.8	Features from Grand Midwest Express Hotel Apartments in Dubai month before burst . . . . .	79
7.9	Manually feature results based on whole sentences and sentence clauses . . . . .	82
8.1	List of features . . . . .	90
A.1	List of crawling locations . . . . .	93

# Chapter 1

## Introduction

This chapter provides an introduction to the project and the motivation behind it. Further, the problem definition and goals as well as project scope and contributions are listed. At the end of the chapter, a detailed outline of the report is provided.

### 1.1 Motivation

Text mining, and especially opinion mining, has received much attention in later years due to the many possibilities this technology provides. Most of the information available today is unstructured, especially on the Internet. Being able to find and extract relevant information from such large data sets, can be of great value for businesses. Everything from analysing customer opinion regarding their products, to creating knowledge resources from company documents, can be important for better business decision-making.

The emergence of text mining tools have thereby come as a result of the enormous interest and the quantity of data that is gathered and stored every day. Identifying and finding a use for all of the information and knowledge in this data is not an easy task. These huge amounts of data are too large for humans to comprehend and process manually, and thus text mining tools and techniques for performing automatic analysis and extraction of relevant data are of great interest. In addition, the growth and emergence of the World Wide Web has made data available regardless of geographical location, further increasing accessibility and volume.

One popular area amongst researchers and business analysts are the highly dynamic and changeable opinions of customers. The obvious motivation being to gain an advantage in knowledge about possible weaknesses about a product or service. Analysing opinions using various statistical tools or text mining techniques is still an ongoing field, and has seen an increasing interest and research in recent years. Still, no methods have yet been



discovered which can consistently accomplish such a task. Previous attempts have had limited success in some of these areas, but few, if any, have succeeded in creating an analysing tool together with a graphical user interface.

This project aims to further contribute to this research. This project will look at ways to use opinion mining techniques to analyse changes in opinions about hotels. Hotels as a customer-based service is an area where multiple factors may impact customer sentiment. For instance noise, nearby construction, weather, even customer expectations. Such events may influence the overall sentiment at any given time, creating a dynamically changing sentiment. Managing to identify why changes occur in such a setting, may provide both customers and hotel owners valuable information regarding the interpretation of large amounts of opinionated data.

Typical scenarios include:

- A user wants to detect up and coming areas
- A user wants to look at sudden changes in hotel sentiment
- A user wants to find hotels with the best breakfast

Opinion mining tools are used to identify and extract subjective information from user reviews, and then to determine the sentiment of the text. Three different techniques are looked at: feature extraction, burst detection and visualisation. Feature extraction is a technique to identify and extract product features, burst detection is used to detect and analyse abnormal changes and visualisation means to present the information graphically. Evaluation is performed by comparing the actual review scores with our sentiment scores.

## 1.2 Problem Definition and Goals

The assignment text is given below.

*The aim of this project is to investigate various issues of temporal opinion mining in relation to hotel reviews, and to create a graphical prototype of the temporal information. Also an important aspect is the creation of a relevant data set containing user reviews and hotel information. In short, the project consists of five parts:*

1. *Perform a literature study of the field*
2. *Find and create a data set containing user reviews and hotel information*
3. *Identify new challenges to be solved and develop appropriate techniques*
4. *Create a prototype using temporal information*
5. *Evaluate the proposed techniques*

As can be seen here, our main goal for this project is to try out different opinion mining techniques to automate extraction of important information about hotel reviews. Based on

that we will attempt to determine the effect on hotels over time. To that end we need to collect a large data set of relevant hotel information and hotel reviews. The results shall be presented visually to easier detect possible "hidden" information.

To achieve this goal, much of the project will be related to studying techniques and algorithms for extracting important textual information from documents and determine its affection. These documents need to be processed with respect to identifying relevant features and removing noise. Studying techniques for pre-processing of textual data is therefore also of some relevance. Additionally, some visualisation theories will be looked at. However, our main focus will be on creating a modular framework for data set extraction using different opinion mining techniques and a prototype to analyse hotel reviews. The purpose of creating this framework and prototype is to make it easier and more efficient to test and validate these algorithms and techniques, as well as being a tool for future research in this area.

Minor goals are listed below.

- Identify relevant text analysis techniques for data pre-processing
- Identify relevant opinion mining techniques
- Identify relevant feature extraction techniques
- Develop and implement a prototype, utilising one or more of any researched algorithms
- Identify components and find a recommended approach for analysing hotels based on user reviews
- Identify interesting areas for further research

## 1.3 Contributions

Our main tool to detect changes in opinion mining was a web prototype we created to visualise possible changes. The prototype can be seen in figure 1.1. This provided a great way to detect "good" and "bad" areas based on hotel reviews in a user-friendly map view. Additionally, it includes a feature search, where users can find hotels based on features from user reviews. These scores are calculated based on user opinions, and is an effective way for users to filter sentiment data.

The prototype was tested on five different locations with a total of almost 800 000 reviews from 2004 to 2012. We had great success in deciding whether a hotel review is "good" or "bad". Usually we had around 70%-80% correctness depending on review source. However, detecting changes in opinion over time based on external factors, proved to be difficult, as external factors are unknown and one needs a huge amount of data to be able to detect them. It was therefore often unclear why such changes had occurred. Nevertheless,

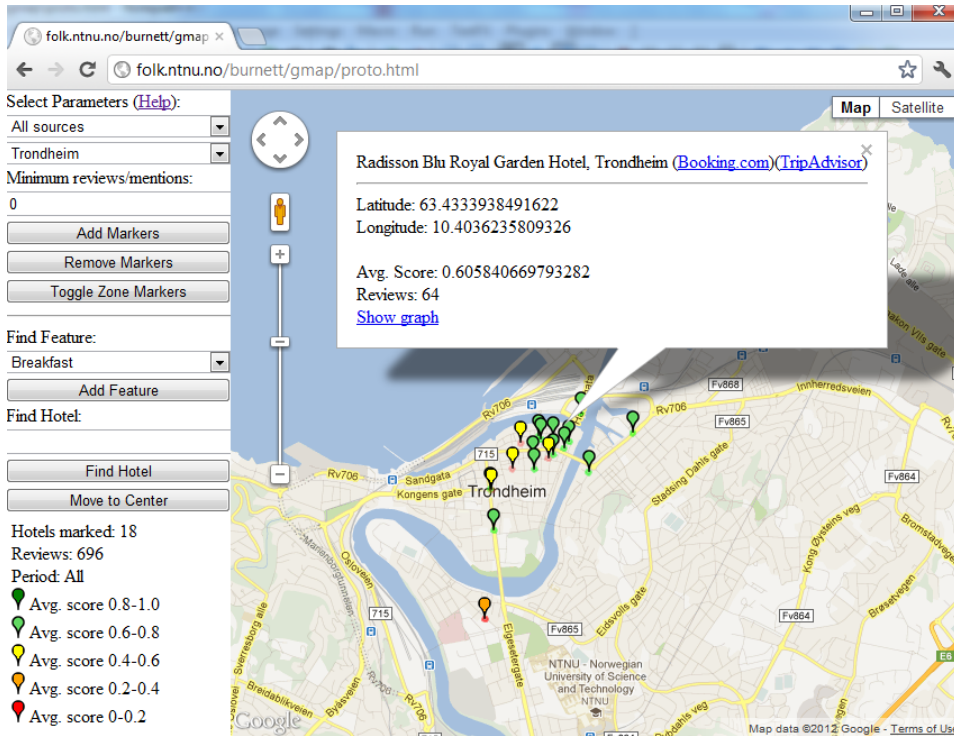


Figure 1.1: Screenshot of the prototype

detecting more known, trivial features like decline in service or improvement in cleaning standard, were easier to detect.

Also, we created a data set with user reviews and hotel information. We combined information from two sources: Booking.com and TripAdvisor in a structured format. Since one needs a lot of data to substantially test opinion mining techniques and algorithms, this data set can be used for further research in this area, saving researchers time to gather relevant test data.

## 1.4 Project Scope

This project will be limited to studying techniques within opinion mining with temporal aspects. Only topics relevant to this project will be discussed and evaluated. The system to be implemented, will have limited functionality. The purpose of the prototype is to test and evaluate different algorithms and techniques to detect possible changes in time, as well as being a framework for future work.

The focus of this project will be on automatic techniques and algorithms that might be used to extract temporal information about hotels. That is, to identify and extract meaningful document features and to determine its effect on hotels. Optimisation of techniques and algorithms, as well as identification of other and potentially more promising techniques, are left for further studying.

Note that it is not the goal of this project to necessarily find the best approach to analysing hotels, but to focus on an approach that might provide meaningful results. Additionally, the project aims to identify special challenges and aspects relevant for further study.

The focus will be on a literature study, system development and experimentation, as presented in figure 1.2.

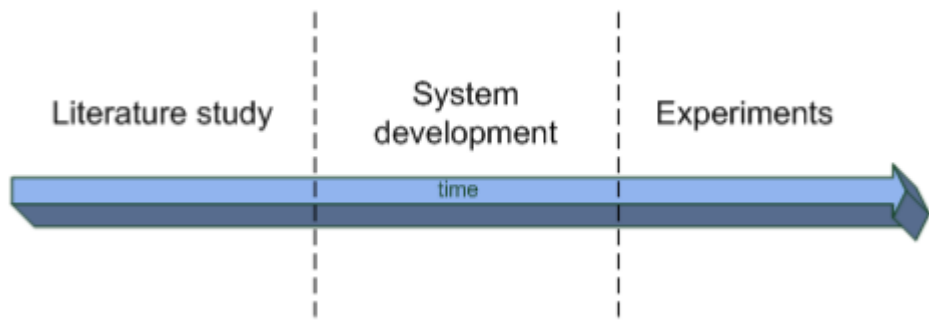


Figure 1.2: The project approach

## 1.5 Report Outline

### Chapter 1 - Introduction

This chapter provides an introduction to the project, the motivation behind it, the problem definition and the goals and the scope of the project.

### Chapter 2 - Background

The chapter gives background information in relation with opinion mining and temporal opinion mining.

### Chapter 3 - Text Analysis

This chapter presents different theories used in this project from the fields part of speech tagging, WordNet and word sense disambiguation.

### Chapter 4 - Opinion Mining Techniques

In this chapter the opinion mining techniques used in this project are presented. This includes challenges, knowledge-based approaches, supervised and unsupervised approaches, language models and time-aware aspects.

**Chapter 5 - Extracting Suitable Data Sets**

The data sets created are presented in this chapter. Everything from data crawling to how the data sets are structured to problems, experiences and setbacks are given.

**Chapter 6 - Opinion Visualisation Prototype**

An explanation of our prototype is given here, with focus on the graphical map prototype. A brief overview of our opinion mining framework is also presented.

**Chapter 7 - Experiments and Results**

The chapter will present the experiments performed in this project, and give an overview of the results.

**Chapter 8 - Discussion and Conclusion**

Discussion of the results together with the conclusion and suggestions for further work is presented in this chapter.

## Chapter 2

# Background

The first part of the project is to perform a literature study, identifying existing studies and potentially functioning systems which utilise opinion mining techniques. This chapter details this research. First, we take a look at the background of opinion mining, with an introduction about sentiment classification, along with a review of previous studies. These are included to better understand what opinion mining is, and what has been done in this field in the past. Following that is background information about advanced opinion mining techniques like feature-based opinion mining and multi-polarity classification.

The next section in this chapter contains information about temporal opinion mining. It describes previous studies about opinion mining with time-awareness, along with a short introduction to opinion spam.

## 2.1 Opinion Mining

Opinion mining or sentiment analysis involves classifying opinions in text into different categories, typically "positive", "negative" and "neutral" [36]. In other words, opinion mining aims to determine the attitude of a writer by identifying and extracting subjective information in a textual document. The attitude is the judgement, evaluation or emotional state of the writer toward something or someone, usually a product, service or a person. One example is an application that would be tracking what bloggers, discussion forums or tweets (post on Twitter) are saying about a brand like Toyota.

This section will first take a short look at the history of opinion mining and a quick presentation of what sentiment classification is. Following that, we will go more in-depth about previous work done in this field. In the end of this section, feature-based opinion mining and the difference between binary and multi-polarity opinion mining are explained.

### 2.1.1 History

Textual information can be broadly classified into two categories, "facts" and "opinions". Facts are objective statements about entities and events in the world, while opinions are subjective statements that represent peoples attitude. Historically text mining procedures have focused on extracting factual information, examples include web search engines, information retrieval and many other natural language processing tasks. It is only recently opinion mining has enjoyed a burst of research activity, since there was little opinion related text to analyse before the World Wide Web. Most opinion data had to be gathered through manual surveys [36]. This created an extra barrier because of the extra resources needed to perform opinion analysis.

Commercially, opinion mining techniques have become increasingly popular in the business community through so called business intelligence. Business intelligence is a term describing computer-based techniques used for identifying, extracting and analysing business data.<sup>1</sup> One of the objectives include understanding and analysing the companies strengths and weaknesses, and their relationship with customers. One thing is to analyse objective measures like a products weight, size or cost, but it is more difficult to measure a products subjective aspect like the design of the product or its usability. Opinion mining can help answer these kinds of subjective preferences from the customers.

A 2008 study by Horrigan [22] showed that 60% of Americans had at one time done an Internet search for a product they were considering to buy. A quarter of these did so on an average day. With the ever increasing growth of the Internet, it is natural to assume this number has not gone down since then. This highlights the value of having generally positive opinions on the web, and how much reach a negative review can have if published on frequently visited websites. This is not only relevant to online shopping, as another study [18] showed that consumers also used online web searches on products before offline purchases.

### 2.1.2 Sentiment classification

Given a set of documents  $D$ , a sentiment classifier categorises each document  $d$  into different classes, typically positive, neutral and negative [35]. Negative means that  $d$  expressed a negative opinion, neutral a neutral opinion and positive means that  $d$  is regarded as a positive opinion. The main task of sentiment classification is to give a quick determination of the prevailing opinion about a subject in a textual expression. The task is similar to classic text classification based on topics, which classifies documents into different topic classes, for example sports, economy and culture and so on. But there is one main difference. In topic-based classification topic related words are very important, but in sentiment classification those words do not have to be of any importance. The most important words in sentiment classification are words that indicate negative or positive opinions, like "good", "bad", "excellent", "horrible" et cetera. However, some topic-related words can be of some

---

<sup>1</sup><http://www.businessdictionary.com/definition/business-intelligence-BI.html>

importance when it comes to finding out what the expressed sentiment is geared towards. Also, one can extend classification down to the sentence-level. In other words, instead of classifying the whole document, each sentence is classified.

Mainly there are two different approaches to sentiment classification. The first one is to determine its classification based on sentiment phrases. This method uses positive and negative words and phrases contained in text to determine its classification. One good source is the approach of Turney [62]. We mostly use sentiment phrases to determine why there is a change in hotels over time. This is described more thoroughly in section 4.2.

The second approach is by using machine-learning techniques with classic text classification algorithms like naive Bayes, support vector machine, k-nearest neighbor et cetera [49]. A more detailed description of these techniques can be found in section 4.3. Additionally, there have been classification approaches by using score functions [14]. These types of algorithms usually consist of counting terms and measuring probabilities, but have not been a big focus in this project.

### 2.1.3 Previous Work

One of the early projects, which can be counted as a forerunner of the opinion mining area as it deals with beliefs, was done by Carbonell [7]. He developed a computer model of subjective understanding called POLITICS. POLITICS could model the political beliefs and inferred expectations of either an American liberal or a conservative based on certain political stories related to United States and Russia and their international policies. Each ideology produces a different interpretation of the events, and POLITICS demonstrates its understanding by answering questions about what the American administration will decide to do in a situation where some of their goals are threatened by actions from the Russians.

Other than some very early work from artificial intelligence studies, previous work within opinion mining can mainly be categorised into two categories: knowledge-based and statistical [15]. The knowledge-based approach uses linguistic models or some other form of knowledge to get insight into the sentiment of a sentence. Later approaches apply statistical or machine learning techniques for achieving sentiment classification. A brief look at previous work done in both areas follows.

#### **Knowledge-Based Sentiment Classification**

One of the earliest techniques within knowledge-based sentiment classification was developed by Hearts. Models inspired by cognitive linguistics were used to decide the directionality, that is whether the agent is in favor of, neutral or opposed to a specific event [21]. A similar technique using a system called SpinDoctor, which identifies a news story's point of view, was made by Sack [55].

Later, researchers such as Huettner and Subasic created a prototype system through manual or semi-manual construction of a discriminate word lexicon and fuzzy logic to categorise



sentiments [25]. Also, Das and Chen used an English dictionary, a hand-picked finance word lexicon and a grammar corpus to extract investor sentiment from stock message boards [13]. This together with different classification algorithms coupled together by a voting scheme, resulted in accuracy levels similar to that of widely used Bayes classifiers.

Tong presents a system for generating sentiment timelines [59]. This system tracks online discussions about movies, and shows a plot of the number of negative sentiment and positive sentiment messages over time. Messages are then classified by looking for specific phrases that can indicate a sentiment towards the movie. Each phrase must be manually added to a special lexicon with a corresponding tag indicating positive or negative sentiment. Hatzivassiloglou and McKeown focused more on semantic orientation of individual adjectives, rather than phrases containing adjectives or adverbs [19]. Their method achieves high precision, but it also relies on a large corpus that needs a big amount of manually tagged training data.

There has also been some research on negative/positive term counting methods provided by Turney and Littman [61]. They determined the similarity between two words by counting the number of results returned by web searches joining the words with a NEAR query operator, and based on that automatically determined if a term was negative or positive. The algorithm got an accuracy of around 80% with 3596 words tested.

Nasukawa and Yi took a different approach to sentiment analysis [42]. They illustrate an approach to extract sentiments from sentences that contain opinions for specific subjects from a document, instead of classifying the whole document into positive or negative. They first identify sentiment expressions in different texts, the polarity and strength of the expressions, and whether the expressions indicate a positive or negative opinion towards a subject. Also, they chose a particular subject of interest and manually defined a sentiment lexicon for identification. The prototype system achieved high precision of 75% to 95% correctness depending on the data. The test was performed on different web pages and news articles.

### **Statistical Sentiment Classification**

One of the earlier works in statistical sentiment classification or so called modern opinion mining was done by Lee, Pang and Vaithyanathan [49]. They examined several supervised machine learning techniques for sentiment classification to a database of movie reviews. Three machine learning methods were used, namely naive Bayes, maximum entropy classification and support vector machines, with eight different sets of words and n-grams as features. The best result was achieved using support vector machine with a unigram feature set. This produced an accuracy of 82.9%. Their naive Bayes classifier with a unigram feature set achieved an accuracy of 81.0%. They concluded that machine learning techniques outperform the method that is based on human-tagged features, although the performance was lower compared to those reported for standard topic-based categorisation.

Pang and Lee continued their research by deciding whether a given text has a factual or opinion nature by implementing techniques for finding minimum cuts in graphs with the

assumption that sentences close to each other tend to have the same class [46]. Since a movie review consists of both objective and subjective sentences, their beliefs were that they only need to use the subjective portions of the review to decide overall sentiment. They achieved some success in this approach with accuracy better than or at least as accurate as the accuracy of the full text review, while only containing 60% the size of the original review.

Further research by Pang and Lee include classifying movie reviews as either positive or negative as before, and then decide its strength as ratings on a three or four star scale [47]. Also there has been performed an in-depth analysis of restaurant reviews by Snyder and Barzilay [58], which predicted ratings for various aspects of the given restaurant, such as the food and the atmosphere on a five-star scale.

### 2.1.4 Feature-based Opinion Mining and Summarisation

One of the problems studying texts at the document level, is that a document can contain positive loaded text on a particular object, but that does not necessarily mean that the author has positive opinions on every aspect of the object. Likewise, a negative opinionated text does not mean that the author dislikes everything about the object. For instance, in a movie review, the author usually writes about both the positive and negative aspect of the movie, although the overall sentiment might be gearing towards negative or positive. To separate the different opinions in a document, one can use feature-based opinion mining on the sentence level.

Three tasks have to be performed [23] [24] [50]:

1. Identifying and extracting features of the object that the authors have expressed their opinion on
2. Determining whether the opinions on the features are negative, positive or neutral
3. Summarising the results

For example in the sentence "the picture quality of this camera is terrible", the feature is "picture quality" and the opinion expressed on this feature is negative, since "terrible" is typically a negative word. After that each review is summarised. One example of how summarisation can be done can be seen in figure 2.1. In this example each review is summarised first by the product name, in this case "Digital\_camera\_1", and then each feature follows with number of positive and negative review sentences about that particular feature.

Blair-Goldensohn et alii have published a study called "Building a Sentiment Summariser for Local Service Reviews" [5]. They claim web-based opinions are becoming the main platform for measuring quality of most products and services. Therefore they present a system that summarises the sentiment of reviews regarding localised businesses. Summaries are determined by splitting different aspects of the service into smaller, focused parts. Text related to given aspects are then analysed for any form of sentiment. Each

```

Digital_camera_1:
  Feature: picture_quality
    Positive: 253
              <individual review sentences>
    Negative: 6
              <individual review sentences>
  Feature: size
    Positive: 134
              <individual review sentences>
    Negative: 10
              <individual review sentences>
  ...

```

Figure 2.1: A summary example [23]

aspect then receives a summarised sentiment based on all the relevant data from the data set.

## 2.1.5 Binary versus Multi-Polarity Analysis

Binary, or standard polarity analysis, means purely positive or negative sentiment. This means that there is nothing that is slightly positive or slightly negative. Each sentiment is either positive or negative. Studies on opinion mining using binary analysis include [49] [62]. An alternative to binary analysis is multi-polarity analysis, which results in degrees of sentiment, for instance slightly happy, very happy, a little sad and so on. Numeric values are often used to separate such polarities, for instance using anywhere between 0 and 1, where 0 is no relation to given sentiment and 1 is the complete opposite. Values in between describe the grade of sentiment. Wilson et alii [65] describe difficulties with this approach.

Algorithmically a binary analysis is simpler than multi-polarity approaches. It is basically a check to see whether or not a given sentiment can be linked to the data set. Often basic classification algorithms can be successfully adapted to such sentiment mining tasks. With multi-polarity however, one needs to determine the degree to which each sentiment matches a data set, resulting in somewhat more complex calculations.

As part of a study by Pang and Lee [48], they looked at the differences between binary classification and multi-polarity classification. One thing they found was that neutral texts were often perceived to be more negative than neutral by readers. Also, classifying a data set as "neutral" does not necessarily mean the text is objective. A neutral opinion can be as strong as a negative or positive one [48].

## 2.2 Temporal Opinion Mining

Temporal opinion mining is a relative new field and has many different names including time-aware opinion mining, temporal sentiment analysis, opinion change mining and opinion tracking, to mention a few of them. Temporal opinion mining is the process of monitoring and detecting possible changes to specific opinions over given periods of time, and can be seen as a continuation of opinion mining.

This section will first present the temporal process added to regular opinion mining. Then previous work is detailed, and lastly an interesting area closely related to temporal opinion mining called opinion spam is looked at.

### 2.2.1 Temporality Process

The basic process involves determining the average opinion on a given topic at two or more unique moments in time. Ideally this results in a complete time series, which is then mapped out with changes over time. The timeline can be presented by the predominant polarity [31] or as a graph based on sentiment value [17]. Any changes in opinion can then be identified, and may be used to find patterns [9]. This is one of main advantages compared to basic opinion mining. However, changes in opinion are by themselves not necessarily useful without something to compare them to. Therefore the usefulness of change detection becomes more apparent when combined with understanding why said change occurred.

To determine why an opinion has changed, one needs to find correlations between opinions and the textual content of a data set. This can be done by determining a set of association rules from each collected data set. These rules match textual tokens and map them to a certain opinion. This gives a set of tokens which are determined to belong to a given opinion. With enough such rules, a system can be taught to determine subjectivity of textual data sets. When utilising these rules and opinions when comparing data sets, it is theoretically possible to determine patterns in any changes of opinion. Different change patterns can be defined in several ways, but they mostly relate to whether or not changes match the rules for each data set.

Change is defined as either gradual or sudden, and based on any determined rules, expected or unexpected [9]. Gradual or sudden describe the degree in which an opinion has changed. An expected change is a change which corresponds to the rules for each data set. An unexpected change does not correspond to the specified rules. For instance, if two data sets have very similar rules, but completely different opinions attached to them, the change might be unexpected.

### 2.2.2 Previous Work

Being a new field of study, there are relatively few studies done in this area. Temporal opinion mining continues the work done with opinion mining and adds the factor of time awareness into the process. Most studies utilise the opinion mining techniques described in section 2.1, but with added time awareness.

In a study by Fukuhara et alii [17], they looked at news and blog articles and produced two sets of graphs: a topic graph and an emotion graph. The topic version graphed out topics associated with a certain sentiment. With a specific sentiment, it was possible to see when certain events were highly associated with that sentiment. As such the graph illustrated which events had the greatest impact on the given sentiment at specific moments in time. The emotional graphs showed a range of sentiments over time regarding a given event. This approach is the more common way to illustrate temporal opinion mining results. Having a specific topic or event, and seeing how sentiment toward it changes and fluctuates as time goes by. Combining the analysis of these two types of graphs, the result is a solid visual representation of the association between sentiment and event. However, this approach requires both sentiment and event to be known prior to analysis. It does not attempt to discover and identify previously unknown events which might have had an effect on sentiment.

In another study, by Dipankar Das et alii, they created a prototype system called TempEval-2007 with the TimeML framework.<sup>2</sup> It is designed to create visualisations of opinions over time and track changes, but focuses on temporal relations between events associated with sentiments [12]. They employ a machine learning approach based on Conditional Random Field [32] for solving the identification problem of event-event relations, with the TimeML-based TempEval-2007 system and considering sentiment as a feature of an event. This is done by classifying the event pairs into their respective temporal classes. Like the study by Fukuhara et alii, this differs somewhat from our approach since it does not attempt to find unknown events based on changes in sentiment.

Prior to TempEval-2007, a system called MoodViews<sup>3</sup> was created by Mishne et alii [40]. The system tracks the mood of blogs hosted by LiveJournal.<sup>4</sup> Conceptually similar to TempEval-2007, the system continuously downloads updates from thousands of blogs. It tracks moods, predicts what they might become and analyses them in an attempt to understand why specific changes in mood occur. The mood tracker follows moods in close to real time and creates graphs based on these time series. The mood predictor combines natural language processing with machine learning to estimate moods based purely on textual content. This is then compared to the actual mood data gathered from tagging and an accuracy statistic can be determined. Using language statistics it finally identifies terms which occur more often or less often than usual during certain peaks in mood.

---

<sup>2</sup><http://timeml.org>

<sup>3</sup><http://www.moodviews.com/>

<sup>4</sup><http://www.livejournal.com/>

### 2.2.3 Opinion Spam

With the continued growth of the Internet, opinions on more or less any topic are easily available. This is especially true when it comes to consumer products [34], but is relevant for basically anything one may have an opinion on. Consumer opinion is therefore a valuable commodity for many companies. Swaying opinion regarding a certain product to be more positive, and negative for a competing product, can prove to be highly valuable. Due to this, creating fake opinions and spreading them online may be considered useful for several parties. Companies may use it to both improve opinion regarding their own products, as well as lowering opinion on competitor products. This, unfortunately, gives good incentives for opinion spam, which is human activity in the form of spam reviews that try to give undeserving positive opinions to some targets in order to promote their products or services, or undeserving negative opinions to other targets in order to damage their reputation [35] [45]. Opinion spam is an increasing problem both when performing opinion mining and in the more general case of a consumer assessing what the public opinion is about a certain product. The influx and impact of such spam has resulted in it being compared to web search spam in terms of disruption and annoyance [27].

Viewing opinions over time can make it easier to detect opinion spam [35]. It makes it easier to find rating spikes and rating and content outliers. Examples of more traditional spam detection techniques include duplicate detection, compare average ratings from multiple sites and to compare review ratings of the same reviewer on products from different brands.



## Chapter 3

# Text Analysis

Techniques and tools for analysing and processing input text are an important aspect of any type of text mining. Any data set containing human-generated written text in its basic format only contains raw text. This is basically a long list of different characters, only contextually and semantically understandable by humans. The potentially large amounts of text however make it impossible to manually process it all. Therefore several techniques for automatically and correctly marking the data are required.

Put simply, this process can be broken down into three main steps. The first step is to find techniques for breaking down sentences into specific terms, marking the correct grammatical context of each. What type of word it is makes it possible to extract the correct dictionary definition, which leads to the second step. A single word may have multiple meanings depending on context. Determining the correct one is therefore highly important. With the correct grammatical tagging in place, a dictionary look-up for selecting the most likely definition may be achieved. From there, it is possible to begin step three, which is to properly analyse and determine the sentiment of the textual subset. This chapter describes the first two steps in this process. Step three is described in chapter 4.

### 3.1 Part of Speech Tagging

Part of speech (from here on referred to as POS) tagging, also called grammatical tagging, is the process of marking up each word in a text as corresponding to a particular part of speech, based on its definition and relationship with related and close words in a phrase, sentence or paragraph [35].

This section will start with an introduction to what part of speech tagging is, and then go into more detail of different tagging approaches. In the end more practical uses of part of speech tagging is explained, both in opinion mining in general and in our application.



Stayed at this hotel for one night due to business in Trondheim.

Figure 3.1: Raw text before POS tagging

Stayed\_VBN at\_IN this\_DT hotel\_NN for\_IN one\_CD night\_NN due\_JJ to\_TO business\_NN  
in\_IN Trondheim\_NNP .\_.

Figure 3.2: Text after POS tagging

### 3.1.1 Introduction

A simplified form of part of speech tagging is typically the identification of words as nouns, verbs, adjectives et cetera. The eight common English part of speech categories are: noun, verb, adjective, adverb, preposition, pronoun, interjection and conjunction [28]. There are also more subcategories within those categories. For example an adjective can be an adjective in its base form, superlative form and so on.

POS tagging can be seen as a way to disambiguate part of speech, where the tagger algorithm returns the most likely tag for a word based on the surrounding context. If one consider the word "break", it can be used both as a noun, for instance "The break starts now", or as a verb, for example "I am going to break this chair". In other words POS tagging helps to figure out the meaning of a word being used.

Additionally, there are some differences between the tags returned for each word. The tags depend on which tag set is being used. However, most tag sets usually use at least some variant of the eight common English categories mentioned above, but many tag sets use more. Also, there are language differences. Japanese, for instance, has three different classes of adjectives, while English and Norwegian have one.

One of the most well known tag sets is the Penn Treebank tag set, which has over 40 different categories. Figure 3.1 shows a regular sentence before POS tagging, and figure 3.2 shows the same sentence after tagging with the Penn Treebank tag set. All the tags are explained in table 3.1.

### 3.1.2 Tagging Approaches

POS tagging algorithms are usually classified into two different groups: rule-based taggers and stochastic taggers. Rule-based taggers generally involve a manually created database where disambiguation rules apply, while stochastic taggers pick the most likely tag for a word based on probability. There are also taggers that try to have a combination of those two architectures, typically called transformation-based taggers. This section gives a brief description of each of these three approaches. For a more detailed presentation please see [28].

Table 3.1: The Penn Treebank POS tag set

<b>Tag</b>	<b>Part of speech</b>
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP\$	Possesive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP\$	Possesive wh-pronoun
WRB	Wh-adverb
\$	Dollar sign
(	Opening parenthesis
)	Closing parenthesis
,	Comma
.	Sentence-final punctuation
:	Colon, semi-colon

**Input word:** "that"

**if** next word is adjective, adverb, or quantifier, and following which is a sentence boundary. And the previous word is not a verb like 'consider' which allows adjectives as object complements

**then** eliminate non-ADV tags

**else** eliminate ADV tag

Figure 3.3: Example of "that" rule in rule-based tagging

### Rule-Based Taggers

Rule-based tagging is the earliest approach to assigning part of speech. It is based on a two-stage approach [28]. First, a dictionary is used to assign all possible tags for each word. The second stage consists of narrowing down the list to a single part of speech tag for each word. This is done through a large set of disambiguation rules. An example of such a rule is shown in figure 3.3.

One popular rule-based tagger is the ENGTWOL tagger [64].

### Stochastic Taggers

Stochastic taggers usually resolve ambiguities by using a training corpus to compute the probability of a word having a given tag in a given context [28]. Stochastic taggers often produce a reasonable output without seeming to know anything about the rules of a particular language. A stochastic approach includes most frequent tag, n-gram and Hidden Markov Model (HMM). HMM is the stochastic model which is mostly used in part of speech tagging. Since we can not observe the sequence of tags, only the sequence of words can directly be observed, an HMM will enable us to estimate the sequence of tags, which is "hidden" from the observer, and then choose the most likely sequence with the highest probability. More formally HMM taggers choose the tag sequence which maximises the following formula:

$$P(\text{word}|\text{tag}) * P(\text{tag}|\text{previous } n \text{ tags}) \quad (3.1)$$

where  $P(\text{word}|\text{tag})$  is the most frequent tag (likelihood) and  $P(\text{tag}|\text{previous } n \text{ tags})$  is the N-gram (a prior).

The probabilities used in the formula above are collected from a tagged corpora. One example for the likelihood for a noun and a verb to appear after the word "race" in the Brown Corpus <sup>1</sup> is shown below.

$$P(NN|\text{race}) = .98$$

$$P(VB|\text{race}) = .02$$

---

<sup>1</sup>General corpus in the field of corpus linguistics for the English language

### Transformation-Based Taggers

Transformation-based taggers is a combination of rule-based taggers and stochastic taggers. This kind of tagging is also sometimes called Brill tagging after Eric Brill who first came up with the method. It consists of three major stages [6]. First, it applies rules which specify what tag a word should be assigned, just like in rule-based taggers. Then it examines every possible transformation of the rules and selects the one that results in the most improved tagging, similar to stochastic tagging. Lastly, it re-tags the data according to the rules, and this continues until a threshold is met.

### 3.1.3 Part of Speech Tagging in Opinion Mining

In opinion mining part of speech tagging is often used to extract adjectives. Research on subjectivity detection revealed a high correlation between the presence of adjectives and sentence subjectivity [20]. This finding has often been taken as evidence that adjectives are a good indicator of sentiment. However, this does not imply that other part of speech do not contribute to the sentiment. There have been studies that used only adjectives as features in machine learning algorithms, and they performed worse than using the same number of most frequent unigrams [49]. The researchers point out that also verbs like "love" and "like" can be strong signal for sentiment.

### 3.1.4 Part of Speech Tagging in our Application

In our case we use adverbs, nouns, verbs and adjectives to determine its sentiment, and disregard prepositions, pronouns and conjunctions, since they rarely contribute to the sentiment. Interjections, however, can be more interesting to look at. For example the interjection "ooh" in the phrase "Oooh. Hotel Rica Bakklundet lies by the river Nidelva". Clearly for the human user this is a positive phrase because of the interjection, but in our application this phrase will be deemed objective, since the rest of the sentence states an objective fact. The reason for exclusion of interjections is two fold. SentiWordNet, which we use to calculate the sentiment value, and described in section 4.2.1, does not have support for interjections, so if we are going to support it, we ourselves have to add scores for each interjections manually. Secondly, interjections are more used in conversions than in reviews, for example we checked interjections for all the reviews for London from TripAdvisor. That data set includes 167 655 reviews and 26 068 271 words. There we found in total 11 437 interjections. In other words 0.044% of all the words are interjections. This is, however, something that could easily be added in the future.

We use Stanford Log-linear Part-Of-Speech Tagger<sup>2</sup> to perform the part of speech tagging. This tagger uses a Maximum Entropy model, which is similar to stochastic tagging, with the Penn Treebank tag set [60].

---

<sup>2</sup><http://nlp.stanford.edu/software/tagger.shtml>

## 3.2 WordNet

WordNet<sup>3</sup> is a large lexical database for the English language, and was created and is being maintained at the Cognitive Science Laboratory at Princeton University [39]. It groups nouns, verbs, adjectives and adverbs into sets of synonyms called synsets. Each synset corresponds to one underlying lexical concept and a gloss describing the concept it represents.

For example, one of the synsets for the word dog is {dog, domestic dog, *Canis familiaris*} with the following gloss: "a member of the genus *Canis* (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds". In addition, many of the synsets contains examples of usage, for instance "the dog barked all night".

Most synsets are linked to other synsets through semantic relations. These relations vary based on whether the synset represents a noun, verb, adjective or adverb because they follow different grammatical rules. A short description of the different relations for the different word types are given in table 3.2.

WordNet is freely and publicly available, and can be accessed online through a web browser<sup>4</sup> or through download.<sup>5</sup> Figure 3.4 shows the result of performing a search for the word "dog" in the web interface of WordNet. As seen from the figure, there are 7 different synsets for "dog" as a noun and 1 synset as a verb. All these synsets show the different senses the word can have. To have a more precise meaning of a text, one has to determine which sense of a word is used. This is called word sense disambiguation, and is described in section 3.3.

We use WordNet as a lemmatisation tool and to get the specific word ID, so we can quickly find the sentiment value of the word through SentiWordNet, since SentiWordNet uses the same words and word ID as WordNet. It is also used in the word sense disambiguation process described in section 3.3. SentiWordNet is explained in section 4.2.1.

## 3.3 Word Sense Disambiguation

Word sense disambiguation (WSD) is the problem of identifying appropriate meaning or sense of a word used in a sentence when the word has multiple meanings [35]. This section will start with an introduction to word sense disambiguation, then go more in-depth with an algorithm called Lesk. Lastly, practical use in our prototype is detailed.

---

<sup>3</sup><http://wordnet.princeton.edu/>

<sup>4</sup><http://wordnetweb.princeton.edu/perl/webwn>

<sup>5</sup><http://wordnet.princeton.edu/wordnet/download/>

Table 3.2: Semantic relations in WordNet

Type	Relation	Explanation	Example
Nouns	Hypernyms	A is a hypernym of B if B is a kind of A	Canine is a hypernym of dog
Nouns	Hyponyms	A is a hyponym of B if A is a kind of B	Dog is a hyponym of canine
Nouns	Holonym	A is a holonym of B if B is a part of A	Bicycle is a holonym of pedal
Nouns	Meronym	A is a meronym of B if A is a part of B	Air bag is a meronym of car
Verbs	Hypernym	The verb A is a hypernym of the verb B if the activity B is a kind of A	To perceive is an hypernym of to listen
Verbs	Troponym	The verb A is a troponym of the verb B if the activity A is doing Y in some manner	To march is a troponym of to walk
Verbs	Entailment	The verb A is entailed by B if by doing B you must be doing A	To sleep is entailed by to snore
Adjectives	Similar-to	The adjective A is similar to the adjective B if A and B have similar meaning	Beautiful is similar-to pretty
Adjectives	Pertainym	The adjective A is a pertainym of the adjective B if A is of or pertaining to B	Academic is a pertainym of academia
Adjectives	Antonym	The adjective A is an antonym of the adjective B if A means the opposite of B	Pretty is an antonym of ugly

## Noun

- **S: (n) dog, domestic dog, Canis familiaris** (a member of the genus *Canis* (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*
- **S: (n) frump, dog** (a dull unattractive unpleasant girl or woman) *"she got a reputation as a frump"; "she's a real dog"*
- **S: (n) dog** (informal term for a man) *"you lucky dog"*
- **S: (n) cad, bounder, blackguard, dog, hound, heel** (someone who is morally reprehensible) *"you dirty dog"*
- **S: (n) frank, frankfurter, hotdog, hot dog, dog, wiener, wienerwurst, weenie** (a smooth-textured sausage of minced beef or pork usually smoked; often served on a bread roll)
- **S: (n) pawl, detent, click, dog** (a hinged catch that fits into a notch of a ratchet to move a wheel forward or prevent it from moving backward)
- **S: (n) andiron, firelog, dog, dog-iron** (metal supports for logs in a fireplace) *"the andirons were too hot to touch"*

## Verb

- **S: (v) chase, chase after, trail, tail, tag, give chase, dog, go after, track** (go after with the intent to catch) *"The policeman chased the mugger down the alley"; "the dog chased the rabbit"*

Figure 3.4: Entry of "dog" in WordNet

### 3.3.1 Introduction

Part of speech tagging can be seen as a crude form of word sense disambiguation, which is explained in section 3.1, but this section deals with more fine grained methods and algorithms. As an example consider the word "bridge" and two of its meanings:

- Bridge - a structure that allows people or vehicles to cross an obstacle such as a river or canal or railway etc.
- Bridge - any of various card games based on whist for four players

Then consider these two sentences:

- The vehicle has to cross a bridge to reach its destination.
- Bridge is played with a standard deck of 52 playing cards.

For a human it is pretty obvious which sense of the word "bridge" is used in these two sentences, but to develop algorithms to replicate this human ability can often be a difficult task. The process of WSD can mainly be divided into two steps:

1. Find all senses for all relevant words that shall be disambiguated
2. Give each word its correct sense

The first step is straightforward, since there exists many computer-readable dictionaries or knowledge sources that may be used. WordNet is one such dictionary, which contains all

possible senses for each word, as described in section 3.2. The second step is accomplished by relying on the context or external knowledge sources [26]. The context can include both information within the text and extra-linguistic information, such as the situation. External knowledge sources like lexical or encyclopedic resources can also provide data useful for associating words with senses.

### 3.3.2 Lesk Algorithm

The Lesk algorithm is based on the hypothesis that words used together in text are related to each other, and that this relationship can be observed in the description of the words and their senses [33]. The algorithm therefore looks for overlaps in the descriptions, and chooses the sense that has most correlation in its description. The most simple version of the Lesk algorithm simply counts the number of same words in all the words definitions of every sense, and picks the sense that has the highest count. Consider an example with the context "bridge player" and with the following dictionary definitions:

1. Bridge - a structure that allows people or vehicles to cross an obstacle such as a river or canal or railway etc.
2. Bridge - any of various card games based on whist for four players
1. Player - a person who participates in or is skilled at some game
2. Player - someone who plays a musical instrument (as a profession)

If one count each of the word in those four definitions, only counting adjectives, subjectives, verbs and adverbs, one will found that the most likely sense is 2. bridge and 1. player with a count of 1. This is because "game" as a subjective is included in both the descriptions. None other words is included in other definitions, and therefore has a count of 0. This was of course a simplified example with very small context, but the same principle applies for more than 2 words, only with a longer and more complex computation.

One of the challenges with this algorithm is to choose how many of the words in the context should be used in the disambiguation. The more words chosen, the longer time the calculations will take. On the other hand, to few context words lead to a higher error rate.

### 3.3.3 Usage in our Application

In our application we use a dictionary-based method. First we use WordNet to find all the senses for each word, and then we use the Lesk algorithm to determine the most likely sense. The results with this approach can be seen in section 7.1.2. We have also used the standard WordNet approach, which basically uses the most common sense for each word. This way it is possible to compare Lesk against the standard to see if one performs better.

We did not find any libraries with the Lesk algorithm in Java, so we have so far only implemented the simple version from scratch, but in the future one can try to make a more advanced version, like the adapted Lesk algorithm [3]. Another reason for not focusing on



improving the Lesk algorithm is that we already had very good results in opinion mining, as shown in section 7.1. A comparison of the two can also be found in that section.

## Chapter 4

# Opinion Mining Techniques

This chapter describes the theories and algorithmic techniques utilised in the project for analysing opinion. The goal is to find efficient methods for extracting the semantic context of documents. The chapter starts with describing some of the challenges with opinion analysis, then presents the two main approaches: knowledge-based and supervised. Other operations and techniques related to sentiment analysis like unsupervised learning, language models and feature extraction will also be presented.

Additionally, temporal aspects will be looked at. This part will describe ways to detect changes in opinion over time. Two aspects are described: burst detection and opinion visualisation.

## 4.1 Known Challenges

Most of the challenges in opinion mining are related to the authenticity of extracted data and the methods used [48]. Often a document contains sentences with mixed views. For example assume a news article about two different companies, say Statoil and SAS, contains positive news about Statoil, but negative news about SAS. Should this text be classified as positive or negative? Or maybe neutral? Another issue is that a word could be considered positive in one situation and negative in another situation. For example the word rise, can be considered both positive and negative depending on the context. If the costs rise, that is definitely negative for the company, but if the value of the company or the revenue rises then that is positive. This example is maybe more of a general text mining problem than pure opinion mining, but the same will apply to subjective opinions.

To only look at opinions as simply negative or positive is one form of evaluation, but it overlooks the comparing factor. Comparisons may be objective or subjective. For example, an objective sentence is "this cellphone is 10 grams heavier than iPhone 4S". This is a

stated fact, and does not necessarily have an impact on which cellphone is the better one. A subjective sentence can for example be the sentence "this cellphone is better than Sony Ericsson Xperia Play, but worse than iPhone 4S". Is this a positive sentence? Or a negative sentence? Clearly, the cellphone is regarded, according to the author, better than Sony Ericsson Xperia Play, so that is a positive evaluation, but worse than iPhone 4S, which is a negative statement.

Additionally, opinions may change over time. If a lot of customers complain about a product, the company will eventually take notice and try to fix it. In other words opinions can be outdated after some time. Other challenges include misleading opinions like sarcasm and irony and that people have different writing styles. A person can, for example, use a word that can be regarded negative for some, but neutral or even positive for others.

Determining subjectivity and sentiment of a document is one thing. Finding the general sentiment in huge collections of data sets is quite another. What one person thinks of a product is often not interesting. What 10,000 people think of the same product almost always is. Due to this, along with the sheer amount of information available at times, the need to summarising opinions [4] regarding given topics arises.

Another problem is domain dependence, that is that the general notion of positive and negative opinions can be different depending on the domain. For example the sentence "go read the book" can be a positive sentence in book reviews, but negative in movie reviews. One way to deal with this was provided by Yang et alii, who saw if the features were good domain-independent indicators by checking that the features were good in two different domains, in this case movie reviews and product reviews [66].

It is also worth mentioning that human classification has around 70% correctness because human raters typically agree about 70% of the time [38]. Thus, a system that has around 70% accuracy is as good as human raters, even though it may not sound too impressive. If a program were "right" 100% of the time, the average human would still disagree with it around 30% of the time.

## 4.2 Knowledge-based Approaches

A number of approaches to opinion mining take the effort of first creating an opinion lexicon. This can be done in many ways. The simplest method is to manually decide the degree of positivity and negativity of words, and then have some way of calculating the sentiment for each sentence or whole text based on the values of the words. This can of course be tricky because it is difficult to set a sentiment value on a simple word, since many words can be both positive and negative depending on the context. Adding word phrases and have some form of word disambiguation can help improve the results, but it will still be extremely time consuming doing this manually.

In the other extreme, one can create an opinion lexicon in an unsupervised manner. An example of that can be found in [62], where adjectives and adverbs are first extracted.

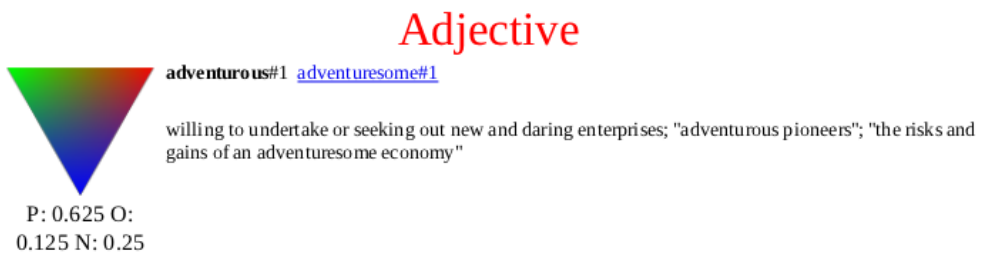


Figure 4.1: Entry of "adventurous" in SentiWordNet

Then the semantic orientation is decided by an algorithm that uses mutual information as a measure of the strength of semantic association between two words [10]. The last step is to calculate the average semantic orientation of phrases in the given text based on the created lexicon, and classify the text as positive or negative.

An approach that combines both manually and automatically labelling is SentiWordNet, which is the main approach we use in our application. SentiWordNet is a good sentiment lexicon with very high accuracy, as shown in section 7.1.

#### 4.2.1 SentiWordNet

SentiWordNet is a publicly available lexical resource for opinion mining and is freely available for research purposes [16]. It can also be accessed through a web-based graphical user interface,<sup>1</sup> as can be seen in figure 4.1. SentiWordNet is based on the lexical database of WordNet, and automatically annotates each synset of WordNet according to three sentiment scores: positivity, negativity, objectivity, describing how positive, negative and objective the terms contained in the synset are. In SentiWordNet 3.0 only the positive and negative scores are included in the database, so the objectivity score is calculated as:

$$ObjScore = 1 - (PosScore + NegScore)$$

Terms may also have different senses, and thus possibly different sentiment properties. For example the synset "estimable" with the sense "may be computed or estimated" has 1.0 in objectivity and 0.0 in negativity and positivity. Another synset "estimable" with the sense "deserving of respect or high regard" has a positivity score of 1.0, and negativity and objectivity of 0.0. Each of the three scores range in the interval from 0.0 to 1.0, and their sum is 1.0 for each synset. Additionally, a synset may have a non zero score for each of the categories, which means that a synset may have some degree of each of the three opinion-related properties. For instance, the synset "adventurous" with the meaning "willing to undertake or seeking out new and daring enterprises" has a 0.625 positive score, 0.25 negative score and 0.125 in objectivity, as shown in figure 4.1.

SentiWordNet is built in a two-step process [1]. The first step consists of semi-supervised learning with weak supervision, where WordNet term relationships such as synonym,

<sup>1</sup><http://sentiwordnet.isti.cnr.it/>

Table 4.1: Sentiment classification with different keyword lists [48]

	<b>Proposed Word Lists</b>	<b>Accuracy</b>	<b>Ties</b>
Human 1	positive: <i>dazzling, brilliant, phenomenal, excellent, fantastic</i> negative: <i>suck, terrible, awful, unwatchable, hideous</i>	58%	75%
Human 2	positive: <i>gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting</i> negative: <i>bad, cliched, sucks, boring, stupid, slow</i>	64%	39%
Statistics-based	positive: <i>love, wonderful, best, great, superb, still, beautiful</i> negative: <i>bad, worst, stupid, waste, boring</i>	69%	16%

antonym and hyponymy are explored and automatically extended from a small set of seed words used in [63]. These seed words are known beforehand to have positive or negative opinion bias. After this is done with a fixed number of iterations, a subset of WordNet terms will have either a positive or negative label. Then the glosses from each of the terms will be used to train machine learning classifiers using different algorithms and different training set sizes, so that bias is minimised. In the second step, the predictions from the classifier are used to determine the sentiment orientation of the terms left in WordNet through a iterative, random walk process until the iterative process has converged.

Interestingly, sentiment keywords are best determined automatically by machine learning, which SentiWordNet for the most part does. Manual selection has proven to be less effective. In a study by Pang [46], they found that keyword lists based on statistical information of the selected data set provided a better result than manual selection. This was found to be true even though the length of the keyword lists were the same. The manually created lists resulted in an average accuracy of roughly 60%, while the statistics based result averaged to around 70%. A list of the keywords and the corresponding results are shown in table 4.1.

A few studies exist which utilise SentiWordNet in relation to opinion mining techniques, among them [44] for sentiment classification of film reviews. It got an overall accuracy between 65.85% with regular term counting and 69.35% with a linear support vector machine classifier with scores used as features. 1000 film reviews was used as data.

## 4.2.2 SentiWordNet in our Application

We use SentiWordNet 3.0 in our application which is based on WordNet 3.0. Each adjective, adverb, subjective and noun in a review are looked up in the SentiWordNet lexicon,

and their scores are subsequently added. If the total positive score is larger than the negative score, the review is marked as positive and vice versa if the negative score is larger than the positive score. If the two scores are equal, it is deemed objective or neutral.

Since SentiWordNet outputs sentiment value, it is easy to classify the degree of sentiment. In other words, it is easy to classify the reviews in more than two or three categories. For example we tested with five categories: strong positive, weak positive, neutral, weak negative and strong negative. The results can be seen in section 7.1.3.

## 4.3 Supervised Approaches

Text and document classification is the task of assigning a document to one or more categories or classes based on its contents [37] [54]. Categories or classes are typically human defined for the needs of an application. For example if one has a document with one sentence "Nidaros Cathedral is a Church and a popular tourist attraction", and the categories "Oslo", "Bergen" and "Trondheim", one would obviously want to attach that document to the category "Trondheim", since the Nidaros cathedral is in Trondheim. This sort of text classification is a supervised approach since the classification involves some external mechanism, for example human feedback or training data, that provides information on the correct classification.

There exists different supervised classification algorithms, but one of the most popular is the naive Bayes algorithm.

### 4.3.1 Naive Bayes

Naive Bayes is a simple probabilistic classifier that is commonly used for classification problems. It is based on applying Bayes' theorem with strong independence assumptions [37]. In simple terms, a naive Bayes classifier assumes that a feature of a class is unrelated to other features. For example, a vegetable may be considered a carrot if it is orange and around 10 centimetres long. Even if these features depend on each other, a naive Bayes classifier considers all of these properties to contribute independently to the probability that this vegetable is a carrot.

The calculation of term probability is not very complex, and training the data is fairly efficient. It is also easy to update with new data, which needs to be done occasionally. In spite of over-simplified assumptions, the naive Bayes classifier has worked quite well in many complex situations [52]. Another advantage is that it only requires a small amount of training data to estimate the parameters necessary for classification. Since independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

The two most commonly used ways to set up a naive Bayes classifier are the multinomial model and the Bernoulli model.

### Multinomial Naive Bayes Model

The multinomial classifier selects the most likely classification  $C_{map}$  given terms  $t_1, t_2, \dots, t_{n_d}$  in document  $d$ . This results to:

$$C_{map} = \arg \max_{c \in C} P(c) \prod_{1 \leq k \leq n_d} P(t_k | c) \quad (4.1)$$

where:

$P(t_k | c)$ : Conditional probability of term  $t_k$  occurring in a document of class  $c$

$P(c)$ : The prior probability of a document occurring in  $c$

$n_d$ : Number of such tokens in  $d$

Then  $P(c)$  and  $P(t|c)$  are estimated using Laplace smoothing. Laplace smoothing takes into account if a term does not occur in the training data. This means that the numerator can not be zero if there is zero terms in the training data. We then have:

$$P(c) = \frac{N_c}{N} \quad (4.2)$$

$$P(t|c) = P(t|c) = \frac{T_{ct} + 1}{(\sum_{t \in V} T_{ct}) + B} \quad (4.3)$$

where:

$V$ : Vocabulary

$B$ : Number of terms in vocabulary

$N_c$ : Number of documents in class  $c$

$N$ : Total number of documents

### Bernoulli Naive Bayes Model

The Bernoulli classifier selects the most likely classification  $C_{map}$  given terms  $e_1, e_2, \dots, e_{m_d}$  where each indicates whether the term occurs in document  $d$  or not. This results to:

$$C_{map} = \arg \max_{c \in C} P(c) \prod_{1 \leq i \leq m} P(e_i | c) \quad (4.4)$$

where:

$P(e_i | c)$ : Conditional probability of term  $e_i$  occurring in a document of class  $c$  or not

$P(c)$ : The prior probability of a document occurring in  $c$

$n_d$ : Number of such tokens in  $d$

We then estimate  $P(c)$  and  $P(e|c)$  with Laplace smoothing to take into account if a term does not occur in the training data:

$$P(c) = \frac{N_c}{N} \quad (4.5)$$

$$P(e|c) = P(e|c) = \frac{E_{ce} + 1}{(\sum_{t \in V} E_{ce}) + 2} \quad (4.6)$$

where:

$V$ : Vocabulary

$N_c$ : Number of documents in class  $c$

$N$ : Total number of documents

### Differences

The differences between the Bernoulli and multinomial models are mainly found with regards to the rules and strategies used to estimate and classify. Training is done the same way with both models. In the classification step, multinomial only considers terms actually found in a document collection, while Bernoulli utilises terms not found as well. Because of this the multinomial model generally only classifies to a select few categories. Bernoulli on the other hand classifies to each category separately, only calculating whether or not it belongs. This often results in multiple classifications.

## 4.4 Unsupervised Approaches

Unsupervised classification is done entirely without external information and deals with the problem of trying to find hidden structures in unlabelled data [56]. In other words there is no error or reward signal to evaluate a potential solution.

The most common approach to unsupervised learning is clustering [37]. Clustering is the process of assigning a set of documents into groups, typically called clusters, of similar documents. That way each cluster will contain similar documents, and different clusters will contain dissimilar documents. This can be done by various algorithms, which have different notion in defining a cluster and how to efficiently find them [37]. One example can be to divide the documents in clusters based on the distance between them, with low distances among the cluster members. An example of a data set with a cluster structure can be seen in figure 4.2.



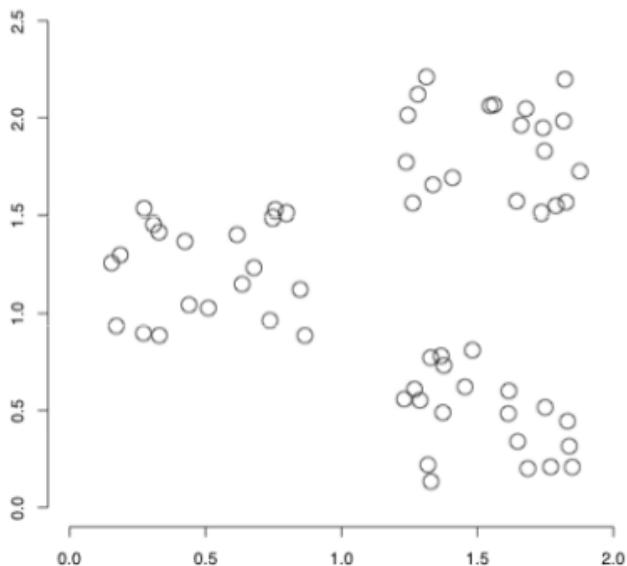


Figure 4.2: An example of a data set with a cluster structure from [37]

#### 4.4.1 Supervised versus Unsupervised

Supervised opinion mining usually utilise machine learning techniques. Such approaches are generally more accurate than unsupervised with 85.54% versus 77% [8]. However, supervised also requires more time to complete, due to the need for training. Supervised machine learning also requires keywords and sentiment calculations to be domain specific. This means training data must be in the same domain as the testing data to be relevant, preferably with a representative selection of sentiments.

Because supervised approaches today is more accurate than unsupervised, we have not implemented any unsupervised approach in our opinion mining.

## 4.5 Language Models

A statistical language model assigns a probability to strings of symbols by means of a probability distribution [37] [53]. Traditionally, these strings are usually sequences of tokens, bytes or characters. In other words, language models estimate the likelihood of a given text belonging to a certain language. First the language models train using training

data consisting of a sequence of texts. After they have been trained, the language models are used to estimate the probability of new text based on the text they saw in training.

The language models we are using are character-based, and are used as a basis of implementation for some of the classifiers. For instance, in our language model based classification, there is a language model for each category, positive, negative and neutral, and texts are classified into the category which assigns them the highest probability. Although a language modelling approach has many similarities with classification techniques we have discussed so far, language models differ in that they assign probabilities to text rather than labels drawn from a finite set. They can, however, convert their output to labels when necessary, and this is something we do in our application, but because of this language models are usually not defined as either supervised or unsupervised classifiers.

Estimating the probability of sequences can become difficult in large texts, in which phrases or sentences can be arbitrarily long and hence some sequences are not observed during training of the language model. For that reason these models are often approximated using n-gram models.

### 4.5.1 N-Gram Models

An n-gram is nothing more than a sequence of n symbols from a text or speech corpus [2]. The symbols in question can for example be phonemes, letters or words. An n-gram of size 1 is referred to as a "unigram", size 2 is called "bigram", size 3 a "trigram", size 4 is a "four-gram" and sizes 5 and up are simply called an "n-gram". For instance, the triple (a, b, c) is a character trigram (3-gram), whereas (I, was, in, Trondheim) is a token four-gram (4-gram).

The basis of an n-gram language model is storing counts for n-grams seen in a training corpus of text. These are then used to calculate the probability of a new sequence of symbols. The key feature of n-gram models is that they calculate the probability of a symbol in a sequence based only on the previous n-1 symbols. The probability of observing the sentence,  $w_1, \dots, w_m$ , is:

$$P(w_1, w_m) = \prod_{i=1}^m P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) \approx \prod_{i=1}^m P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) \quad (4.7)$$

where  $w_i$  is the i-th symbol and n is the number of preceding symbols. The conditional probability for each observation can then be calculated:

$$P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \dots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \dots, w_{i-1})} \quad (4.8)$$

As an example a 4-gram character-based model of the word "text" will have the following probability:

$$P(\text{text}) = P(t) * P(e|t) * P(x|te) * P(t|ext) \quad (4.9)$$

To prevent the possibility of underflow from multiplying long sequences of numbers less than one, the log scale is used to calculate the probabilities. In our case that is especially important since we use a character-based n-grams, which means it often will contain longer calculations than for example token-based n-grams. The LingPipe<sup>2</sup> library is used for implementing language models for sequences of characters. It is also worth mentioning that n-gram models are language independent, since it creates models based on the textual input, but in our case it does not matter because we only deal with English reviews.

## 4.6 Feature Extraction and Summary

As mentioned in section 2.1.4, evaluating a text at the document level and evaluate that document as a whole has some disadvantages. For example a negative evaluation as the document as a whole does not necessarily mean that everything that is mentioned in that document is negative. There can be some specific aspect about that particular object that is positive. Likewise, a positive evaluation does not mean that the author dislikes everything about the object. For instance, in hotel reviews an author usually writes both positive and negative aspects about that hotel, even though the overall sentiment of the review can be either positive or negative. To obtain such detailed aspects, we will have to go to the sentence level and extract the interesting features.

There are basically two different formats that we will have to deal with:

1. Format 1 - Pros and cons: The reviewer is asked to describe pros and cons separately. An example of such a review can be seen in figure 4.3. This is the review structure in Booking.com.
2. Format 2 - Free format: The reviewer can write freely. In other words there is no specific separation of pros and cons. An example is given in figure 4.4. This is the review structure in TripAdvisor.

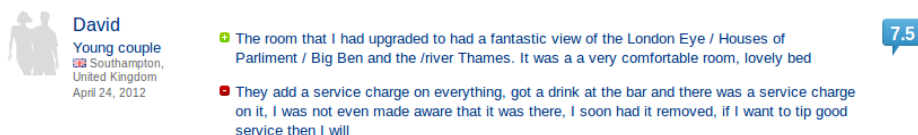


Figure 4.3: A review consisting of pros and cons

For format 1, only the features have to be identified since the semantic orientation is already known, but in format 2 both the features and their opinion orientations have to be found. One of the simplest way to identify features is to rely on the simple notion that features are usually expressed as nouns or noun phrases. This is obviously not always true, for instance, the verb "use" in the sentence "difficult to use". However, in most cases

<sup>2</sup><http://alias-i.com/lingpipe/>



**“Fantastic stay”**  
 ●●●●● Reviewed May 13, 2012 **NEW**  
 2 people found this review helpful

Booked this as an overnight stay because of the fantastic reviews on Trip Adviser, it met and exceeded our expectations. This is definitely THE place to stay in London. Amazing rooms, super staff, nothing too much trouble. Would definitely recommend to anyone and am looking forward to a return visit.

Stayed April 2012, traveled as a couple

●●●●● Value	●●●●● Rooms
●●●●● Location	●●●●● Cleanliness
●●●●● Sleep Quality	●●●●● Service

Angela C  
 Saffron Walden, United Kingdom  
 2 reviews

Figure 4.4: A review consisting of free text

the features are nouns [35]. More advanced methods used to identify features includes association mining and rule mining, but also different heuristic methods [24] [43] [67]. After a feature has been found, it together with its opinion have to be extracted. This can be done by a simple heuristic to extract adjectives that appear in the same sentence as the features [24] or with more advanced procedures like manually or semi-automatically developed language rules [50]. Also sentiment lexicons can be used [41].

In our case we use six predefined features: "breakfast", "staff", "service", "clean", "location" and "internet". These six features are a combination of features from Booking.com and TripAdvisor which a user can grade separately. These features are used together with part of speech tag patterns and a synonym check. We also include cases where the feature is a verb, adverb, adjective and of course a noun. That way most combinations of the features are covered. This is a simplified identification process of features, but it does cover the most important ones, since we also cross-checked them with the two review sites. For extraction we keep the sentence or clause where the feature was included, and then uses SentiWordNet as a sentiment lexicon to determine the sentiment of the feature.

So far, we have talked about analysing and extracting semantic information from documents. The next step will be to present the opinion information in an orderly fashion. For instance, it might be desirable to summarise the main point in a single review or summarise the main points from a lot of reviews of the same product or service. This is called summarisation. There are many different kinds of approaches to summarisation. Everything from recapitulation of a single document to more advanced summarisation where multiple documents about the same topic shall be summarised. This includes techniques for detecting redundancy, identifying important differences among documents and ensuring summary coherence.

In our case, since there is clearly a strong connection between feature extraction and summarisation, the extracted features together with parts of the sentence or whole sentences can serve as a summary [51]. This is what we did in our prototype, and an example of summarisation for the feature "breakfast" can be seen in figure 4.5.

What people think:

- "good breakfast, with wide range of choices"
- "i also enjoyed the breakfast and that there were much helthier than most hotels"
- "and as said the breakfast was lovely though a little too few seats"

Figure 4.5: Example of summarisation for the feature breakfast from our test prototype

## 4.7 Temporal Aspects

Within the domain of temporal opinion mining, opinion lexicon and statistical modelling are the most popular techniques to predict and estimate changes in opinion. These changes can be things such as time or recent events. For instance, as time goes by, consumer interest towards a certain product might naturally diminish. When a competitor releases a clearly superior product, consumer opinion towards older products might plummet. These techniques build on the algorithms presented earlier in this chapter, but includes a new time element.

### 4.7.1 Burst Detection

One of the biggest problems in temporal opinion mining is to find meaningful data from documents that arrive continuously over time. Reviews with different opinions about the same object, growing and fading in intensity for a period of time can be an example of such a problem. More specifically, the main problem will be to find the time periods where certain features have risen in intensity. This provides a framework for analysing why there has been an increase of activity.

The simplest and most used method is to use plain frequencies of the occurrences of words together with some thresholds. One disadvantage with this method is that words that are used very little, but that does change in frequency over time, is hard to detect with these kinds of algorithms. A more advanced approach to burst detection has been provided by Jon Kleinberg [30]. His approach is based on labelling the continuously stream of documents by using a probabilistic automaton, where each states correspond to the frequencies of individual words. State transitions correspond to points in time where the frequency of the word changes significantly. The full algorithm with its description can be found at the home page of Indiana University.<sup>3</sup>

We used burst detection to identify changes in grade scores and opinions over time. However, we used a simplified version with frequencies and thresholds. The first thing our burst detection does is to check if there is a change in the average monthly sentiment score. If that is the case, there could possibly be a big change in sentiment, but this could also just be a one time event. This has to be factored in, so we also check if the reviews in the next, future month is somewhat stable to eliminate possible fluctuation. Of course, this

---

<sup>3</sup><http://iv.slis.indiana.edu/sw/burst.html>

also eliminates the chance of grades moving up or down over the threshold two month in a row. However, this rarely happens. The last thing we do is to make sure that we have substantial data. In this case enough reviews to minimise the importance of just a few reviews. The algorithm can be found in figure 4.6, where `limitChangeScore` is the threshold for changes in average score over a one month period and `limitNumberReviews` is the review threshold.

**Input:** monthly sentiment score for a hotel  
**if:** monthly sentiment score is between the absolute value of a user defined threshold. And next monthly sentiment score is between the absolute value of a user defined threshold. And the total number of reviews for both last and current month is above a user defined threshold.  
**then:** possible burst

Figure 4.6: Pseudo code of our burst detection algorithm

## 4.7.2 Opinion Visualisation

When working with larger data sets, it becomes important to have ways to extract and utilise the data in a useful manner. The amount of data usually results in making manual reading and checking not very feasible. Extracting relevant subsets of data and presenting it therefore needs to be approached structurally. One such way is to visualise data statistics, making it clear and easy to understand what the data tells us. The purpose is to allow the user to interact with and manipulate the data, and it is very useful when it is somewhat unclear what the data set actually shows [29]. There are multiple ways to visualise data. We chose to focus mostly on using charts and graphs for statistics and temporal sentiment visualisation, along with mapping for the data relating to geographical sentiment. Chart and visualisation techniques chosen are based partly on a subjective interpretation of visualisation needs and an online overview of different chart types.<sup>4</sup>

### Charts and Graphs

An important aspect of this project is to visualise sentiment changes over time. The most common way to accomplish this in this specific context is to use a simple two dimensional line chart with sentiment value and time along the axes.

For presenting statistics about a data set, and show comparisons between different subsets, bar charts are a good match. Depending on the type of data, histograms may also be useful.

---

<sup>4</sup>[http://www.sapdesignguild.org/goodies/diagram\\_guidelines/charts\\_bk.html](http://www.sapdesignguild.org/goodies/diagram_guidelines/charts_bk.html)

**Opinion Mapping**

When a data set contains information relating to space, it becomes possible to visualise the data using mapping techniques. One of these techniques are called simple buffering. Simple buffering means identifying regions of a map within a specified distance of one or more features. In our case a feature is a hotel. The distance is then calculated based on the strength of the feature. This makes it easier to see "good" and "bad" areas.

## Chapter 5

# Extracting Suitable Data Sets

Before we were able to start testing our ideas and techniques, we needed an interesting data set to test them on. Two of the largest travel websites, TripAdvisor and Booking.com, with reviews on hotels, were chosen. This chapter describes the steps involved in finding, gathering and storing relevant data.

## 5.1 Web Crawling

Along with the need for a large data set, came the need for an efficient way to gather the large amount of data required and store it in a structured and useful manner. To that end, we created a web crawler. The resulting crawler was a focused and partially automated crawler. It was not a standard web crawler such as those used by most search engines, which continuously harvest URLs and go deeper and deeper into every site. It was a focused crawler, which downloads what it has been told and subsequently terminates when it has completed its task. Figure 5.1 shows a high level view of the intended flow of events for the crawler.

### 5.1.1 Crawling Process

Because the crawler is so focused in its approach, creating a general multi-purpose crawler is difficult. Each site is in most cases unique in how data is presented to the user. Therefore the crawler is separated into modules. The main crawler interface takes in a search term and which site it should crawl, and the actual crawling is then delegated to a separate module. A module exists for each site. Modules are standardised in that they take the search term in string format as input and return a list of hotels and reviews. The inner workings of each module is entirely dependent on the target site.



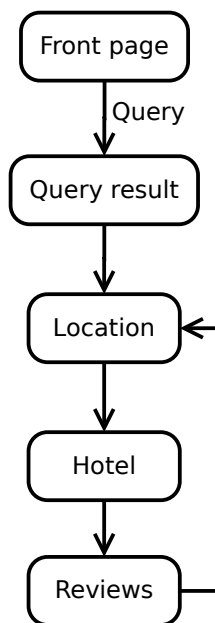


Figure 5.1: High level view of system flow in TripAdvisor and Booking.com crawlers

To initiate the crawler, manual input in the form of a search term is required. The search term should match a specific city or limited area, for instance "London", "Trondheim" or "New York City". Spelling mistakes should be avoided, but minor typographical errors are in most cases not a huge issue. This is however completely dependent on the target sites search functionality and not controlled by the crawler. When a search term has been provided, the crawler searches all predetermined sites and identifies all hotels present at the matched location, along with all corresponding reviews. When all identified hotels have been crawled and the reviews have been outputted, the crawler terminates. It is possible to provide the crawler with a list of search terms, which the crawler will then utilise to continue running and downloading data for all provided locations. This approach helps reduce the supervision the crawler normally requires.

The reason for such a deterministic structure, was to make sure all the information provided for each hotel was downloaded. Additionally, not all places are relevant since there are a lot of small places in the world with very little data, such as Hammerfest or Trondheim. With such a small amount of data, it is very hard to detect changes because a single review will have a tremendous impact on both average and temporal sentiment. Therefore, we made a more focused crawler to get information about the bigger and more popular places. The higher the amount of reviews per hotel at a given location, the better.

+ Nice clean rooms. Friendly staff. To warm at night in rooms. Ventilation very slow.

Figure 5.2: Positive review from Booking.com

## 5.1.2 Sources

The initial sources chosen as a basis for the data set collection were Booking.com<sup>1</sup> and TripAdvisor<sup>2</sup>. They have large databases of hotels along with corresponding reviews.

### Booking.com

Booking.com focuses mainly on hotels, and offers direct reservation of hotel rooms all over the world. The site is easy to navigate and reviews are clear and accessible. However, due to the fact that their main source of income is tied to offering hotel bookings, it is fair to question their neutrality.

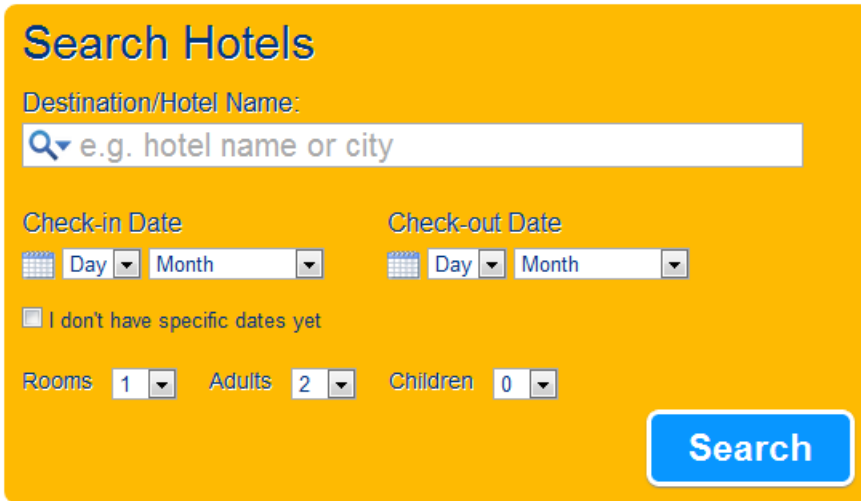
Content on Booking.com is generally user created. Reviews for a given hotel are only available when 5 or more reviews have been submitted. This is done to limit review spam and ensure fair average scores, as well as making sure the hotel is somewhat serious. One review score alone of 10 will not bring the hotel to the top of the pile. To be able to add a review, a user has to have booked their reservation through booking.com. This way reviews are written by people who have most likely stayed at the reviewed hotel and have first hand experience. Users are encouraged and reminded by email to write reviews after their stays. Therefore there are often more reviews available compared to sites like TripAdvisor, but the reviews often contain less text and are quick summaries of user experiences. Compared to other sites, where the review often contains a block of text of varying size, each review on Booking.com has clear and separate areas for pros and cons. Generally, this makes it easier to determine positive and negative sentences, as most text is already divided into positive and negative. Still, by manual examinations, it seems text is not always tagged correctly. For instance figure 5.2 shows part of a review from Booking.com. The text is marked as positive, although the last two sentences are clearly not meant as a plus. User errors such as this are hard to avoid completely.

Finding hotels and reviews on booking.com is fairly straight forward, and follows the high level view of the system flow seen in figure 5.1. The whole process begins by entering a location in the search box on the front page, seen in figure 5.3. Depending on how accurate the search term is, one will most likely be redirected to a list of suggestions based on the search term. Selecting the appropriate location brings the user to the localised page in figure 5.4, which includes a list of all matching hotels. From here one simply follows the links to each hotel. This brings you to figure 5.5. Here one can easily gather hotel names, addresses, star ratings and geographic coordinates. Further navigating leads to figure 5.6 and the all the reviews relating to the given hotel. Reviews are by default presented 20 at a time (editable by changing a URL parameter), and the pages containing them are

---

<sup>1</sup><http://www.booking.com/>

<sup>2</sup><http://www.tripadvisor.com/>



**Search Hotels**

Destination/Hotel Name:

Check-in Date:    
 Check-out Date:

I don't have specific dates yet

Rooms:  Adults:  Children:

**Search**

Figure 5.3: Search area used for finding locations

crawled until a non-English review is found. By default reviews are tagged and sorted based on language, meaning gathering all English reviews is a simple matter of matching the language tag and ignoring those that do not match. All hotels with at least 1 review are output to XML.

24 Hotels found in Trondheim, Showing 1 – 20 [Show map](#)

Sort by: Recommended ▾ Stars Review Score

---



**City Living Schøller Hotel** ★★★👍  
 Trondheim • [Show map](#)  
 In the middle of Trondheim, City Living Schøller Hotel provides air conditioned accommodations right across the street from Stiftsgården royal residence. It offers rooms with cable TV and free Wi-Fi. *There is 1 person looking at this hotel.* [More](#)  
 Latest Booking: 4 hours ago

**Good, 7.5**  
 Score from 448 reviews  
[Show prices](#)

---



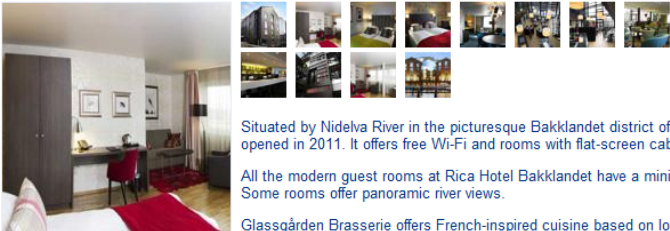
**Britannia Hotel** ★★★★★👍  
 Trondheim • [Show map](#)  
 Dating back to 1897, this elegant hotel is a 5 minute walk from Trondheim Central Train Station. It provides free luxury spa facilities, 4 restaurants and a piano bar. Wi-Fi is free. [More](#)  
 Latest Booking: yesterday

**Very good, 8.0**  
 Score from 174 reviews  
[Show prices](#)

Figure 5.4: Subset of search results

### Rica Bakklandet Hotel ★★★★★

Nedre Bakklandet 60, 7400 Trondheim [\(Show map\)](#)



Situated by Nidelva River in the picturesque Bakklandet district of Trondheim, Rica Bakklandet Hotel was opened in 2011. It offers free Wi-Fi and rooms with flat-screen cable TVs.

All the modern guest rooms at Rica Hotel Bakklandet have a minibar and a private bathroom with shower. Some rooms offer panoramic river views.

Glassgården Brasserie offers French-inspired cuisine based on local ingredients. Traditional Norwegian dishes with a modern twist are served at Glassgården Grill.

The restaurants, bars and shops at the lively Solsiden Marina are within 5 minutes' walk. Nordre Gate pedestrian street is 500 metres away.

*Hotel Rooms: 169, Hotel Chain: Rica Hotels.*


Book now

[See all reviews](#)

Wonderful, 9.1

Score from [55 reviews](#)

Figure 5.5: Presentation of hotel information




**William**  
Solo traveller  
+ St Peter Port, Guemsey, Guemsey  
26 October 2011

✔ Good location, good and comfortable room, friendly and efficient service.

1 of 1 people found this review helpful, did you?

8.3



**Mohamed**  
Group of friends  
+ Hawaii, Kuwait  
24 October 2011

✔ Location location location and cleanliness of rooms

✘ Rooms are so not sound proof that you feel the doors are open and no walls! You literally hear everything happening outside. Also, their payment system has problem as it took more than 45 minutes to settle my bill!

1 of 1 people found this review helpful, did you?

6.7

Figure 5.6: Subset of reviews

## TripAdvisor

TripAdvisor is a travel website that features reviews and advice on travel information like hotels, resorts, flights and so on. Most of the content is written by millions of different travellers, where the travellers share their travelling experience with each other. Like most websites TripAdvisor is free to users and supported by an advertising business model. There are two main differences between Booking.com and TripAdvisor. Booking.com only contains reviews about hotels, while TripAdvisor also includes restaurants, attractions and tours. Secondly, TripAdvisor does not control whether people has actually stayed at a hotel or a resort, leading to that anyone can easily create fake reviews. There have even been agencies that offer to post hundreds of positive comments or negative comments towards their competitors in return for a fee.<sup>3</sup>

The TripAdvisor crawler works similar to the Booking.com crawler by using the built in search functionality, seen in figure 5.7. However, because TripAdvisor does not only contain reviews from hotels, one also has to specify whether one searches for restaurants or hotels, for instance "London hotels". Searches return a list of possible locations. From there on a list of hotels, seen in figure 5.8, is supplied for the selected location. For each of these hotels, the crawler downloads hotel information and review data, as shown in figure 5.9 and figure 5.10. Only the reviews written in English are downloaded and stored in an XML file. Reviews are presented 5 at a time, meaning the amount of pages needed to be crawled can be substantial. Some services have thousands of reviews, so on average, the TripAdvisor is somewhat slower than the Booking.com crawler. Each review consists of a block of text, making sentiment detection more complex than what was the case in Booking.com. The added complexity does however also give us more control over the text and sentiment determination.



Figure 5.7: Search area used for finding locations and services

<sup>3</sup><http://www.dailymail.co.uk/travel/article-2032997/TripAdvisor-investigated-ASA-fake-reviews.html>

## Hotels travelers recommend

30 of 30 hotels shown

Sorted by Ranking List view | Large map

**Rica Bakklandet Hotel** ★★★★★

**Ranked #1 of 30 hotels in Trondheim**  
 ○○○○○ 30 reviews

"Enjoyed it a lot" 01/23/2012  
 "Splendid hotel in Trondheim" 12/30/2011

[Professional photos](#) | [Traveler photos \(6\)](#) | [Map](#)

**Show Prices**

Figure 5.8: Subset of search results

**Rica Bakklandet Hotel** ★★★★★ Like

Nedre Bakklandet 60, Trondheim 7014, Norway [Hotel amenities](#)

**Ranked #1 of 30 hotels in Trondheim**  
 ○○○○○ 30 Reviews

**Show the lowest price for this hotel\***

Check In: 3/2/2012 | Check Out: 3/4/2012 | Adults: 2

**Show Prices**

Hotels.com |  Booking.com  
 Expedia.no |  Apollohotels.no  
 Octopus.com |  Venere.com  
 hotel.info

\*from our partners

[Professional photos](#)

[6 traveler photos](#)

Figure 5.9: Presentation of hotel information

30 reviews sorted by Date Rating English first

**"Enjoyed it a lot"**  
 ○○○○○ Reviewed January 23, 2012  
 2 people found this review helpful

Tried this almost brand new hotel for the first time in Dec 2011, and had a great stay! We were a group of five friends, sharing two rooms - nice and clean, well-equipped and with good beds. The breakfast buffet was a real treat - lots of choice, both hot and cold. The staff was friendly and helpful, both in...

[More](#)

Was this review helpful? Yes [Problem with this review?](#)

Aksel76  
 Oslo, Norway  
 Senior Reviewer  
 7 reviews  
 11 helpful votes

Figure 5.10: Subset of reviews

### 5.1.3 Politeness Policy

An overly aggressive web crawler can in theory be seen as a denial-of-service attack, potentially resulting in both downtime for the target site and banning of the crawler's IP address. To avoid such issues and to limit the load on the target web servers, a short sleep time was implemented. It can be set independently for each separate source. After every page the crawler downloads, it waits before loading the next page. This reduces both the load on the servers and the probability of being blocked for excessive traffic. Average sleep times are in the interval between 1.5 and 15 seconds, dependent on source. The reason for such a big interval is that on Booking.com we got banned by using a sleeping time of 10 seconds. On TripAdvisor we have used 1.5 seconds all the time without issues.

### 5.1.4 Performance

The main limiting factor for the runtime of the crawler is the previously described politeness policy. The time spent actually crawling each page is negligible. Generally each source presents a set amount of reviews per page, normally in 5-100 range. More reviews results in more pages to crawl. Pages are then traversed until all reviews have been identified. The time spent crawling each page is fairly consistent, and in practice negligible compared to the sleep time. Therefore runtime in practice increases linearly based on the amount of reviews per service. Ignoring the initial search crawling and crawling of list of hotels, the average runtime broken down per hotel is roughly formalised as below. In the formula, R is reviews and T is time.

$$\frac{R_{avg}}{R_{page}} * T_{sleep} = T_{avg} \quad (5.1)$$

Average run times for the TripAdvisor and Booking.com crawlers are presented in tables 5.1 and 5.2. As can be seen there, even though the sleep time for the Booking.com crawler is ten times longer than the TripAdvisor sleep time, the Booking.com crawler has significantly lower run times. This is mainly due to the amount of reviews per page (5 versus 100). For example, crawling London with 1069 hotels at TripAdvisor, takes roughly 14 hours while crawling Booking.com with 1228 hotels in London takes about 11 hours. It is also worth noting that the run time for the Booking.com crawler is more stable, leading to more predictable performance measures. The TripAdvisor run time fluctuates significantly depending on review amount per hotel. Based on the examples in table 5.1 alone, it can be seen that run time varies by a factor of roughly 2.5. Those two locations do however contain some of the highest amounts of reviews for all the locations crawled. Therefore they may be seen as an upper limit when estimating performance. This holds true for the Booking.com crawler as well.

Table 5.1: Average crawler run time per hotel at TripAdvisor

Location	Sleep time	# hotels	Reviews/page	Avg. # reviews	Avg. runtime
New York	1.5 sec.	428	5	393	119.4 sec.
London	1.5 sec.	1069	5	157	48.6 sec.

Table 5.2: Average crawler run time per hotel at Booking.com

Location	Sleep time	# hotels	Reviews/page	Avg. # reviews	Avg. runtime
New York	15 sec.	495	100	93	28.9 sec.
London	15 sec.	1228	100	126	33.9 sec.

### 5.1.5 Locations

Locations were selected based on their popularity as tourist destinations. Based on Euromonitor International's top city destinations ranking from 2009, listing the leading 100 cities with regards to tourist arrivals,<sup>4</sup> a subset of the top 50 cities were chosen. The reason for choosing locations based on this list was the assumption that the most popular tourist locations would have a higher amount of hotels, relevant reviews and, more importantly, a higher ratio of reviews per hotel. The complete list of selected locations can be viewed in Appendix A.

## 5.2 Storing the Data Set

To be able to properly utilise the huge amount of collected data, it needed to be stored in a structured manner. It needed to be easily accessible, concise, and portable. The reason for needing it portable was because one of the requirements for the data set was its usability for future projects, both related and unrelated to this project. To that end, the choice fell on storing the collected data in a suitable XML format. The main alternative being a database, such as an SQL database. A properly implemented database would arguably be more efficient, but would be more complex with regards to portability and ease of use in future projects. Importing the XML files into a database in the future should be a trivial task anyway.

### 5.2.1 XML Structure

A specialised XML structure was created for this purpose. Data is grouped according to affiliation, and somewhat spread out to reduce the file size of any one file. Each source has its own folder, so for instance one folder for "booking.com", "tripadvisor" and so on. For each source there is one XML file per location. For the sake of simplicity and identifying

<sup>4</sup><http://blog.euromonitor.com/2011/01/euromonitor-internationals-top-city-destinations-ranking.html>



purposes, the file name of each XML file is the name of the location. An XML Schema was created to strictly define the format of the data sets. All data gathered from all sources will be converted and stored in this format. A current example of the layout can be seen in figure 5.11. An explanation for each tag is shown in table 5.3. Figure 5.12 shows a more concrete example.

There were basically two main reasons for splitting the data set into multiple files based on location. Firstly it would make extraction and storing easier. During initial crawling it was found to be more clear what had and had not been downloaded this way. Secondly, reading and working with the resulting XML files would be easier if the file size was limited. Combining all locations into a single data set would result in an XML file of several gigabytes. However, if and when needed the files can easily be merged, as the root nodes for each file are easily extracted and can be combined into fewer larger files.

```

▼<location name="" last_updated="">
  ▼<hotel name="" address="" stars="" latitude="" longitude="">
    ▼<review user="" review_count="" total_helpful="">
      <date/>
      <country/>
      <profile/>
      <text sentiment=""/>
      <number_helpful/>
      <score/>
    </review>
  </hotel>
</location>

```

Figure 5.11: XML format of data sets

```

▼<location name="london" last_updated="2012-02-20">
  ▼<hotel name="SuperHotel" address="Park 123" stars="4.0" latitude="51.12" longitude="23.90">
    ▼<review user="TheSentimentor" review_count="12" total_helpful="99">
      <date>2012-01-01</date>
      <country>Belarus</country>
      <profile>Young Couple</profile>
      <text sentiment="">Awesome hotel!</text>
      <number_helpful>2</number_helpful>
      <score>9.9</score>
    </review>
  </hotel>
</location>

```

Figure 5.12: Example of XML format of data sets

## 5.3 Preparing Data for Future Use

The raw data set contains a lot of data, and reading and processing this data continuously is time consuming and unnecessary. To reduce this problem, the resulting output from run-

Table 5.3: XML tag explanation

<b>Name</b>	<b>Type</b>	<b>Explanation</b>
<location>	tag	Location tag for each location
name	attribute	Name of the location
last_updated	attribute	Last update of the XML file
<hotel>	tag	Type of service provided. In this case hotel
address	attribute	Address of the hotel or restaurant
stars	attribute	Which star rating it has
latitude	attribute	North-south position of a point on the Earth's surface
longitude	attribute	East-west position of a point on the Earth's surface
<review>	tag	Review tag for each user review
user	attribute	User name
review_count	attribute	Number of reviews from this user
total_helpful	attribute	Amount of people who have found reviews from this user helpful
<date>	tag	Date of when the user review was written
<country>	tag	Name of the location where the reviewer are from
<profile>	tag	Whether the reviewer went there as a young couple, family with children, on business, with friends or alone
<text>	tag	Review text, also optional included is the sentiment, whether the text is positive or negative, if this is known
<number_helpful>	tag	Number of people who have found this review helpful
<score>	tag	The score by the reviewer

ning SentiWordNet and the other techniques described in previous chapters, is stored for future use. These are a list of XML files, similar to the raw data set files, but considerably smaller and easier to work with. They are still divided into tags based first on location, then hotel. Within each hotel, all reviews have had their textual contents analysed, and a score has been output. Each review node only contains the calculated scores. The exact format of this can be seen figure 5.13.

```

▼<review>
  <date>2012-01-27</date>
  <score>8.8</score>
  <pos_score>1.75</pos_score>
  <neg_score>1.5</neg_score>
</review>
▼<review>
  <date>2011-12-19</date>
  <score>7.1</score>
  <pos_score>4.75</pos_score>
  <neg_score>2.875</neg_score>
</review>
▼<review>
  <date>2011-09-19</date>
  <score>6.3</score>
  <pos_score>2.375</pos_score>
  <neg_score>1.75</neg_score>
</review>

```

Figure 5.13: Calculated review score output

All the information needed to determine sentiment and average scores is contained in the previous XML files. However, they can be somewhat large, and the calculations can take time. Therefore all calculations are done once, and are once again output in a new XML file. These files only contain average sentiment scores over different intervals, seen in figure 5.14. The main reason for doing this was to limit on-the-fly calculations when integrating the files into our map prototype. This makes for a more fluent user experience.

The formula for calculating the sentiment of each review is fairly simple. It takes the calculated positive score for the text and divides it by the sum of the positive score and the negative score. In other words,  $\frac{score_p}{score_n + score_p} = Sentiment$ . The resulting score is a value between 0 and 1, where 1 is perfectly positive, 0 perfectly negative and 0.5 has an equal positive and negative score.

```

▼<hotel name="St Giles Hotel & Leisure Club" address="Bedford Avenue,
  Bloomsbury, London, WC1B 3GH" latitude="51.517562683094795"
  longitude="-0.13061821460723902">
  ▼<monthly>
    <score year="2010" month="12" reviews="43">0.5531428138698486</score>
    <score year="2011" month="1" reviews="173">0.6235826242567364</score>
    <score year="2011" month="2" reviews="254">0.6078268534210163</score>
    <score year="2011" month="3" reviews="191">0.621031258928609</score>
    <score year="2011" month="4" reviews="188">0.6006688036008876</score>
    <score year="2011" month="5" reviews="187">0.6106139996137278</score>
    <score year="2011" month="6" reviews="217">0.5902874223076233</score>
    <score year="2011" month="7" reviews="285">0.5893453811497744</score>
    <score year="2011" month="8" reviews="278">0.5728834171362681</score>
    <score year="2011" month="9" reviews="211">0.5837360345405397</score>
    <score year="2011" month="10" reviews="302">0.5876321611278565</score>
    <score year="2011" month="11" reviews="251">0.5809637963075571</score>
    <score year="2011" month="12" reviews="213">0.6204554004696746</score>
    <score year="2012" month="1" reviews="287">0.6317230000552791</score>
    <score year="2012" month="2" reviews="50">0.6288491608708128</score>
  </monthly>
  ▼<total>
    <score reviews="3130">0.6001459941327449</score>
  </total>
</hotel>

```

Figure 5.14: Average scores

## 5.4 Problems, Experiences, Setbacks

This section details and explains some of road blocks we ran into while creating and running the crawler.

### 5.4.1 Connection Problems

We had some connection problems in our crawling. When we crawled TripAdvisor, the connection sometimes reset. We tried to crawl less aggressively to see if that had some effect, but the connection resets still persisted and they seemed very random. But since the crawler checks which hotels have already been crawled, this was not a huge problem. It only can take a bit longer time to crawl a place, since the crawler has to check whether a hotel has been crawled or not an extra time if the connection resets. Additionally, a workaround was implemented which caught the connection reset error and retried the last connection. This way the crawler continues without any serious hits to the total run time.

There were also a connection problem to Booking.com. The reason was that Booking.com has Java headers blocked, probably to limit page crawling. It returned a 400 HTTP bad request error. But this was easy to work around, since one could change the request header to pretend this was not a Java request. In the request header there is a user-agent field, and if one change that from Java to Firefox or some other web browsers, the web server thinks

it is a regular web browser and not a Java crawling program.

### 5.4.2 Crawling Aggression

A typical crawler can theoretically open thousands of connections and read thousands of pages in a matter of seconds. However, depending on the responding web server, this can be a serious problem. Most web servers scale their bandwidth to support their average amount of requests, with the occasional peak during certain time slots. A sudden and substantial increase in requests made by a systematic web crawler can easily bring down an unprepared server, and may even be construed as a denial-of-service attack. Naturally the owners of the web servers do not appreciate such events, and will rightfully do what they can to block overly aggressive crawlers. Therefore it is in our interest as well to limit the amount of connections the crawler makes while gathering data. Finding a decent ratio between run time and aggression can be difficult, and is therefore often a result of trial and error.

As a result of this, the first version of our crawler was found to be too aggressive. This was mostly found to be the case regarding Booking.com, probably due it being the smaller site with a lower amount of bandwidth available. The crawler therefore ran into several issues while downloading data. The biggest problems were from connections being blocked and missing data when connections went through. This indicated booking.com filtered requests based on request headers and sent limited responses or error messages when encountering the default Java headers. Adding a sleep cycle of a certain amount of seconds to the crawler and editing the request headers to match commonly used web browsers alleviated the problems and the crawler functioned properly after this.

### 5.4.3 Updating the Data Sets

Occasionally the crawler needed to be stopped due to various issues, for instance due to rebooting of the machine the crawler was running on. When restarting the crawler, we wanted it to skip already downloaded data. The main reasons for this was to make the crawler finish within a reasonable time frame and reduce the amount of crawling needed. As such every hotel found during crawling is matched against all hotels already present in the corresponding XML data set file. If it exists, it is skipped. This functions fairly well, however there is one issue with this implementation. Reviews for existing hotels are not updated. Any changes or additions to the list of reviews in the data sources are ignored if the hotel has already been crawled. Over short periods of time and for the use of the data sets in this project, this is a non-issue, however, at some point in the future it might be interesting to include reviews written after the initial crawling.

#### 5.4.4 File Size

We quickly noticed that one location can contain huge amount of data. For example the locations New York and London at TripAdvisor each contain more than 150,000 reviews and over 25 million characters in the review texts alone. This results in file sizes of up to 200MB for a single location. Such sizes are still manageable, but are getting close to the practical limit for reading performance using a DOM parser. Larger files take significantly longer to read, and require larger amounts of memory and computational power to adequately process. Spreading the data over multiple files is a simple way to reduce this problem. This is especially true since we for our purposes do not need to read all the data available at all times anyway.

## 5.5 Merging Data Sets

When crawling the main sources of TripAdvisor and Booking.com, two separate data sets were created. In an effort to have more reviews per hotel, we found that it would be a good idea to merge data from the two sources. However, this was initially not as easy a task as expected. It should be pointed out that a 100% match between sources is practically impossible. Each source has a different amount of hotels and some hotels exist in one source but not the other.

### 5.5.1 Hotel Matching

The logical first step of comparing hotel names to each other and finding duplicates only resulted in a match of about 20% of the hotels from each source. A fair starting point, but likely not completely accurate. Another approach was to use GPS locations to find overlapping points. However, the coordinates found on Booking.com and TripAdvisor were occasionally very far off. They seemed to be set either manually or using an address, which results in coordinates varying quite a lot depending on sources used and size of the hotel. For instance, Bab Al Shams Desert Resort & Spa in Dubai spans a huge area. Using the coordinates from Booking.com (24.8163127229457, 55.2300953865051) and TripAdvisor (24.90699, 55.440754) puts the hotel at two different locations roughly 23.5 kilometres apart. This is a rather extreme example, but it is useful in showing that distances may vary from just a few metres to several kilometres, making the accuracy highly debatable.

### 5.5.2 Name and GPS Matching

As such a combination of hotel name and coordinate matching was the logical next step. Different sources may have slightly different spellings for the same hotel. Therefore a

way to compare two strings and determine how similar they are was needed. Several algorithms exist which attempt to solve this problem, among them the Levenshtein distance, Hammer distance, Jaccard similarity and Jaro-Winkler distance [11]. The Jaro-Winkler distance was chosen because it is specifically designed to be efficient and accurate on shorter strings. This was helpful with regards to this project, as it was mainly used to compare names of hotels from different data sets in an effort to determine whether or not they were the same hotel. The resulting score was a value between 0 and 1, where 0 was no similarity and 1 meant identical.

A Java implementation of the Jaro-Winkler formula is bundled with the LingPipe library used in the testing framework.<sup>5</sup> Formally, the Jaro-Winkler distance is calculated as below:

$$d = \frac{1}{3} \left( \frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right) \quad (5.2)$$

where:

- d distance between the two strings  $s_1$  and  $s_2$
- m matching characters in the two strings  $s_1$  and  $s_2$
- t transpositions needed divided by 2

A character from  $s_1$  is a match with a character from  $s_2$  if they are identical and their positions are not further apart than  $\lfloor \frac{\max(|s_1|, |s_2|)}{2} \rfloor - 1$ .

The Jaro-Winkler distance for each of the hotels in the original data sets was calculated. A closest match for each was determined. If the Jaro-Winkler distance of the closest match was greater than 0.9 it was assumed to be a match. If the distance was between 0.8 and 0.9, the Jaro-Winkler distance of a subset of both hotels addresses were compared. Additionally, the distance between each hotel's GPS coordinates was calculated using the Haversine Formula. If the GPS distance is below 500 metres, and the Jaro-Winkler distance of the addresses were a match greater than 0.9, they were assumed to be a match. If the closest match did not qualify for these terms, it was assumed no match had been found.

The Haversine Formula is used to calculate the distance between two points, based on their latitude and longitude [57]. The exact formula is shown below.

$$d = 2r \arcsin \left( \sqrt{\sin^2 \left( \frac{Lat_2 - Lat_1}{2} \right) + \cos(Lat_1) \cos(Lat_2) \sin^2 \left( \frac{Lng_2 - Lng_1}{2} \right)} \right) \quad (5.3)$$

where:

- d distance between two points 1 and 2
- r is in this case the radius of Earth
- $Lat_1, Lng_1$  is the latitude and longitude of point #1
- $Lat_2, Lng_2$  is the latitude and longitude of point #2

<sup>5</sup><http://alias-i.com/lingpipe/docs/api/com/aliasi/spell/JaroWinklerDistance.html>

Table 5.4: Number of services in each data set

Location	TripAdvisor	Booking.com	Combined	% matched
London	1062	1160	723	68.08%
New York	404	414	290	71.78%
Athens	298	221	159	71.95%
Paris	1827	1206	869	72.06%
Dubai	427	371	295	79.51%

The final result from determining hotel matches using a combination Jaro-Winkler and Haversine was a significant increase to the initial 20% achieved with direct matching. Table 5.4 shows a subset of the amount of services matched into the combined data sets. The combined data set contains around 70% of the hotels. Hotels without a match in both original data sets are discarded from the new data set.

## 5.6 Statistics

Following are a number of charts detailing certain aspects of the data sets. Figure 5.15 shows the distribution of reviews according to their allotted scores. It seems that Booking.com does not have any reviews with scores below 2.5, and that for both sources scores are somewhat inflated toward the higher end of the scale.

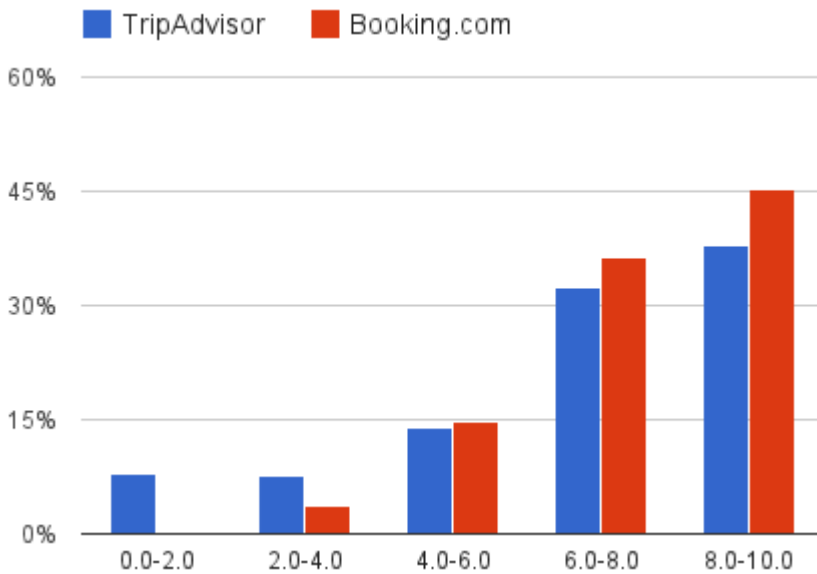


Figure 5.15: Distribution of review scores



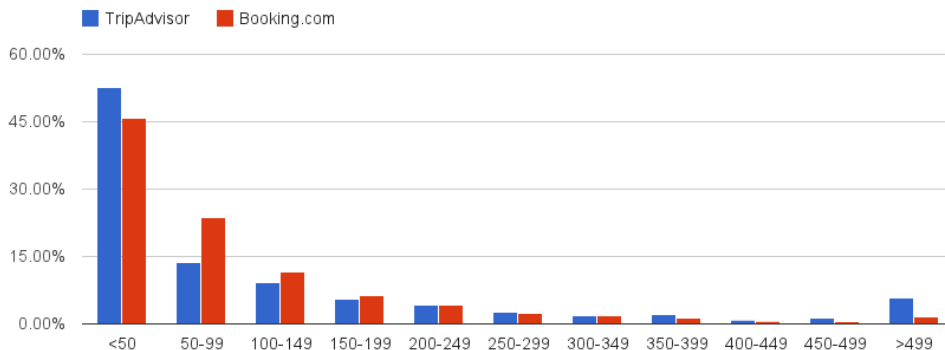


Figure 5.16: Percentage of hotels for each amount of reviews interval

Figure 5.16 is a simple graph showing how many hotels have specific amounts of reviews. As seen, most hotels have less than 50, with declining numbers all the up to 500. The reason for the upward trend at above 500 is due to that range being a much larger interval than the lower ones. Figure 5.17 compares the amount of reviews per location from Booking.com and TripAdvisor. Overall, TripAdvisor has more reviews than Booking.com. Figure 5.18 shows the average scores of hotels and groups them within specific intervals. It is related to figure 5.15, in that average hotel scores are based on the review scores. Therefore the graphs are somewhat similar, with scores leaning toward the higher end of the scale. Booking.com even more than TripAdvisor.

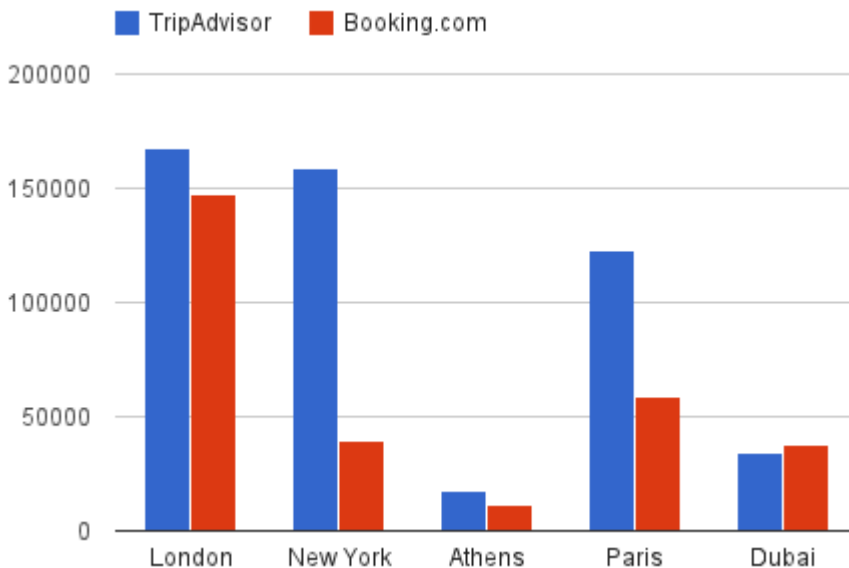


Figure 5.17: Difference in number of reviews per location

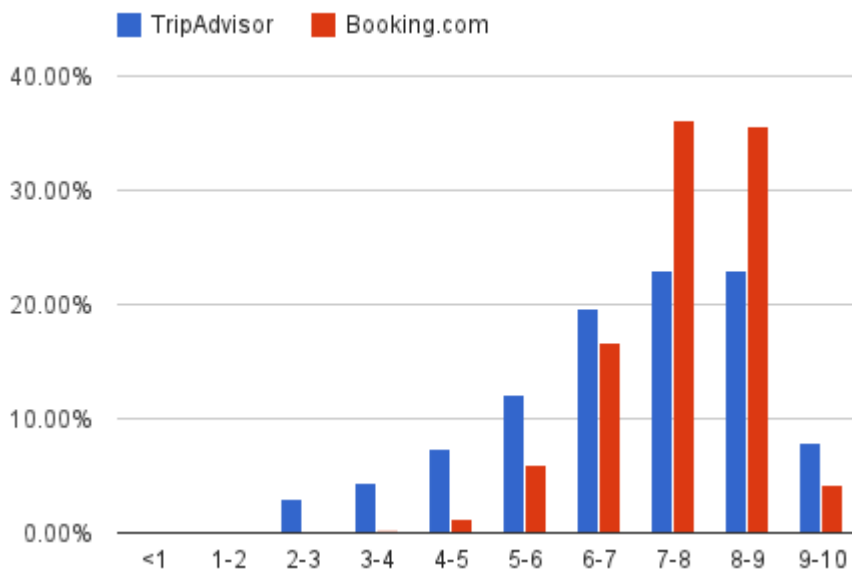


Figure 5.18: Score distribution for TripAdvisor and Booking.com hotels



## Chapter 6

# Opinion Visualisation Prototype

To be able to test our ideas and theories in a controlled fashion, we needed a testing prototype. To this end we created a prototype for visualising sentiment changes over time. With the data sets we gathered, we had all the data we needed to look at sentiment changes over time. We also briefly describe the opinion mining and crawling framework in this chapter.

## 6.1 Opinion Mining Framework

This framework was written in Java and consists of data set crawling, burst detection, pos tagging and different methods for calculating the sentiment of the reviews. We created an easy to use command line interface, so the user can easily run calculations and crawl data. The user interface can be seen in figure 6.1 and consists of 5 different task:

1. Crawl TripAdvisor or Booking.com - Simply crawls all the hotels at TripAdvisor or Booking.com
2. Write feature XML files - Calculates the feature scores for each hotel, and outputs to a XML file.
3. Write score XML files - Calculates all the scores for each hotel, and outputs to a XML file.
4. Write average score XML files - Calculates average monthly score for each hotel. Both the opinion scores and the actual scores are calculated. Then outputs it to a XML file.
5. Run burst detection - Starts the burst detection algorithm for all hotels.

```
1: Crawl TripAdvisor or Booking.com
2: Write feature XML files
3: Write score XML files
4: Write average score XML files
5: Run burst detection
0: Quit
Choose menu item:
```

Figure 6.1: Command line menu for our opinion mining framework

## 6.2 Map Prototype

The prototype was created to visualise certain aspects of the data set. The intention was to have a simple portal to evaluate any and all parts of the data set deemed interesting. The resulting prototype is a web site matching several interesting functions. Among them are the abilities to view a map with averaged hotel sentiments and quickly identify hotels with a high rating. For each hotel, it is also possible to view a graph visualising the fluctuations in customer opinion toward the hotel. From the instance the first review is written, until the last, users can easily see if a hotel is improving or falling apart. Additionally, it is possible to filter hotels based on specific features, not only overall sentiment. For instance, rank hotels based on the quality of their service or friendliness of their staff. Finding areas with clusters of highly thought of hotels is also available, by evaluating the area sentiment. With this function, one can easily find out in which areas one may find the best hotels, and which areas one perhaps should avoid. The rest of this section is dedicated to explaining the functions in more detail, and show what the inner workings of the prototype are based on.

### 6.2.1 Data Set Preparation

Before detailing the prototype, a short introduction to the basic idea is presented. Review sentiment is calculated for each review. Reviews are then grouped based on the date they were written. For each of these groups, every hotel is then marked on a map. The idea here is to have different colours on the markers based on average sentiment. For instance, a good score would have green markers, while a poor score red markers. Colours gradually shift with values between these points. The intervals used to determine groups of reviews are customisable, with everything from a week to a month or more being possible selections.

### 6.2.2 Prototype Description

With the needed subset of data extracted and processed from the data set, data could be mapped and graphed accordingly. This makes it possible to view changes over time and

determine increasing or decreasing popularity of services and areas. Site layout is shown in figure 6.2. It is essentially split into two columns. The wider right column contains the map. This is where all the output is presented to the user. The narrow left column contains interactable controls for manipulating the map. With these controls it is possible to select what and where data should be displayed. The left column also contains some information about what is currently showing on the map, for instance the number of hotels marked and currently used period of time.

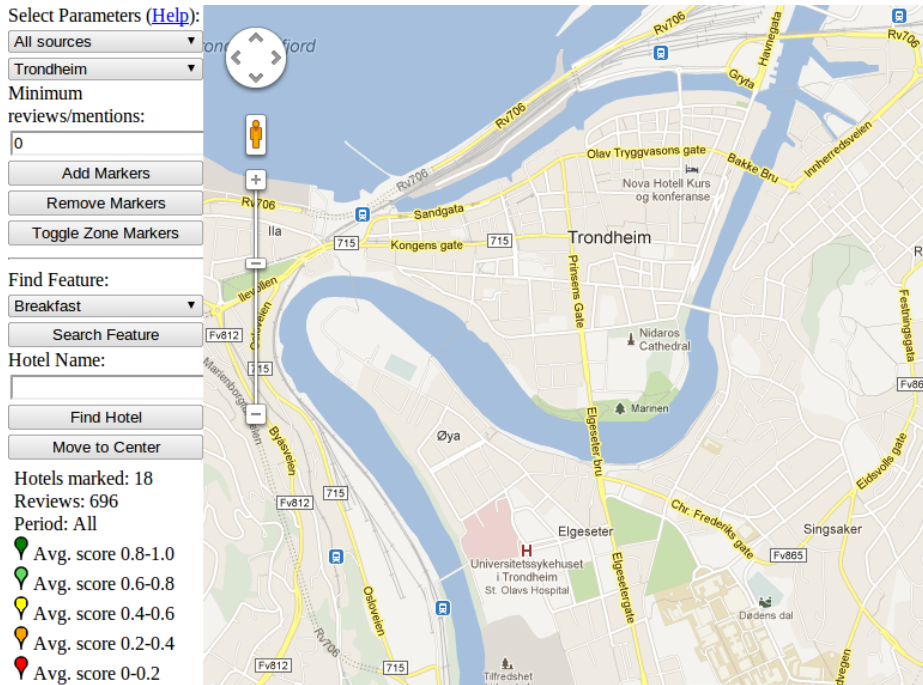


Figure 6.2: Map prototype layout

Figure 6.3 shows an example of the approach. Average score range is from 0 to 1, where each marker corresponds to an interval of 0.2. Figure 6.5 shows the different map markers used and what score they each indicate. Clicking a marker brings up an information window, seen in figure 6.4. This pop-up contains the name of the hotel along with latitude and longitude. Additionally, it displays the average score at that time interval, along with the number of reviews that score has been based on.

The visual representation is set up as website, created with JavaScript, jQuery<sup>1</sup> and Google Maps<sup>2</sup> javascript API version 3.<sup>3</sup> It has been tested with Google Chrome<sup>4</sup> on a basic install

<sup>1</sup><http://www.jquery.com/>

<sup>2</sup><http://maps.google.com/>

<sup>3</sup><http://code.google.com/apis/maps/documentation/javascript/>

<sup>4</sup><http://www.google.com/chrome/>

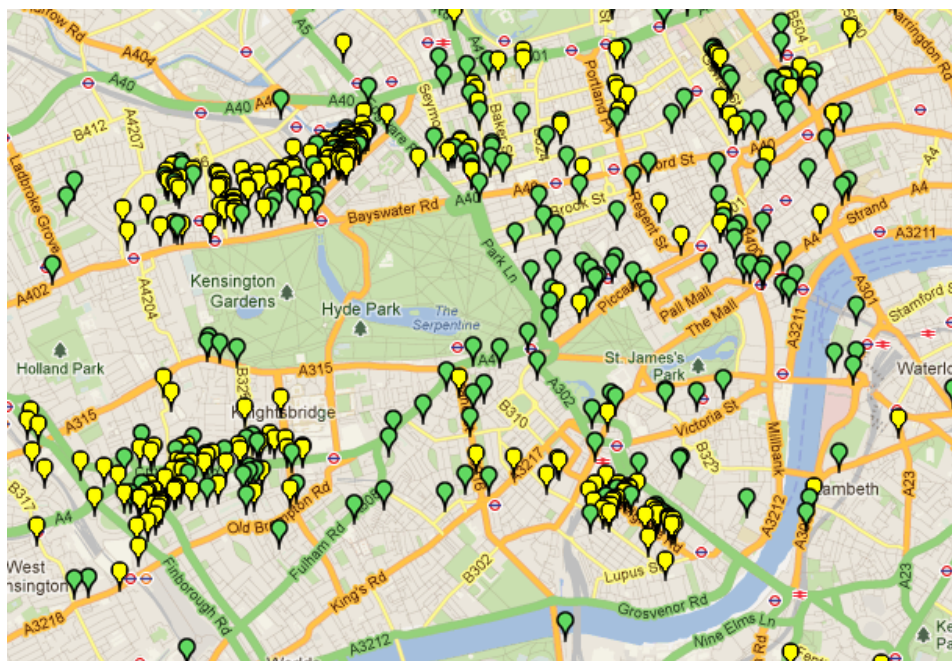


Figure 6.3: Hotels in London from booking.com marked on map without area calculations. See figure 6.5 for marker descriptions.

of Apache HTTP Server version 2.2<sup>5</sup> on Windows 7, as well as an unknown configuration of version 2.2.8 on Ubuntu Server hosted by NTNU(folk.ntnu.no). It should however function as intended on most HTTP servers and browsers with JavaScript support.

The site reads in processed review scores in a predefined XML format using a DOM parser,<sup>6</sup> and outputs correct markers on the map based on location and score. It should be noted that the XML input files may be somewhat large, and if they are not cached (or stored locally by default) before running it may take a few minutes to download and process the files. This is mostly dependent on available bandwidth, but the efficiency of the underlying DOM parser may also result in a few seconds extra wait time.

Map output is controlled by the user with controls on the page. The user may select source, location and the minimum amount of reviews a hotel needs to be displayed. Figure 6.6 shows these controls.

<sup>5</sup><http://httpd.apache.org/docs/2.2/>

<sup>6</sup>[http://www.w3schools.com/dom/dom\\_parser.asp](http://www.w3schools.com/dom/dom_parser.asp)

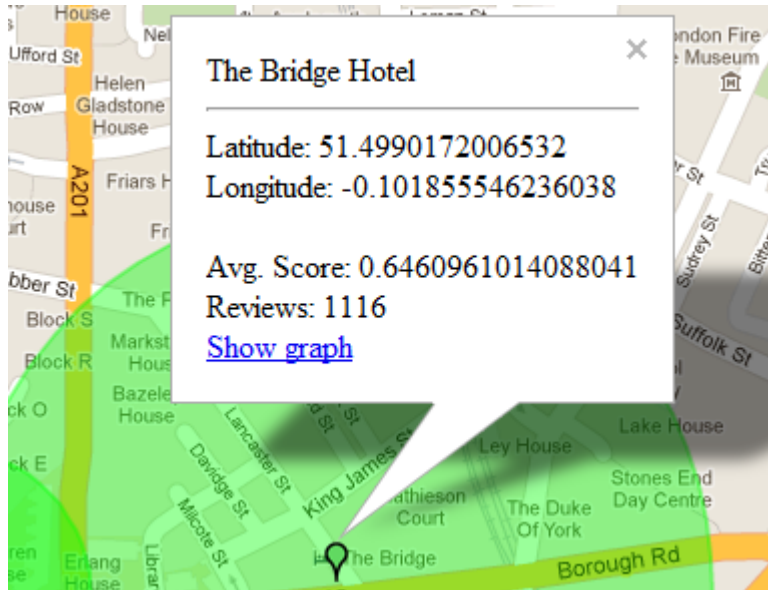


Figure 6.4: Map marker info

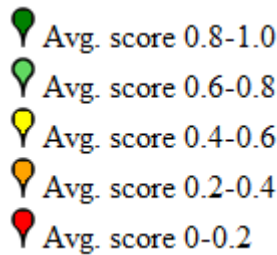


Figure 6.5: Map legend

### 6.2.3 Google Maps

Google Maps is the map API used in our framework.<sup>7</sup> It is free to use and provides an easy way to integrate all the functionality from Google Maps into websites. It also has support for geospatial visualisation and supports various ways to customise the map. Google Maps is one of the most heavily used web application development API, and along with good documentation, it provided everything we needed for our test framework.

<sup>7</sup><http://developers.google.com/maps>



Select Parameters:

All sources

London

Minimum  
reviews/mentions:

0

Add Markers

Remove Markers

Toggle Zone Markers

---

Find Feature:

Breakfast

Search Feature

Figure 6.6: Control panel for the map prototype

## 6.2.4 Area Sentiment

In an attempt to more easily identify areas with a high degree of good hotels, functionality for displaying the score of a hotel as a colouration of a small radius was added. The radius of the coloured circle is dependent on the amount of reviews a hotel has, although with a minimum and maximum radius. The basic calculation for determining the radius is 6.1, where  $r$  is the radius.

$$r = \max(20, \min((reviewCount) * 0.25, 300)) \quad (6.1)$$

The resulting radius lies in the range 20-300 metres. If filtering on amount of reviews is used, that is minimum reviews for a marker to be displayed is greater than 0, this formula has some flaws. If all the hotels displayed have more than 1200 reviews, all circles have a radius of 300, and separating the hotels becomes difficult. To remedy this, a modified version of the formula is used in such cases. The radius is scaled to still lie within the same range and avoid all circles having radiuses close to the maximum limit. The modified version is formula 6.2, where  $r$  is the radius and  $reviewCount$  is always greater than or equal to  $minimumReviews$ . This ensures the whole range of radiuses is still used, and the larger circles have notably more reviews than the smaller ones.

$$r = \max(20, \min((reviewCount - minimumReviews) * 0.25, 300)) \quad (6.2)$$

In addition to scaling the radius of each circle based on review counts, the colour of each circle is based on the calculated sentiment score. Scores of 0.6 and higher are coloured

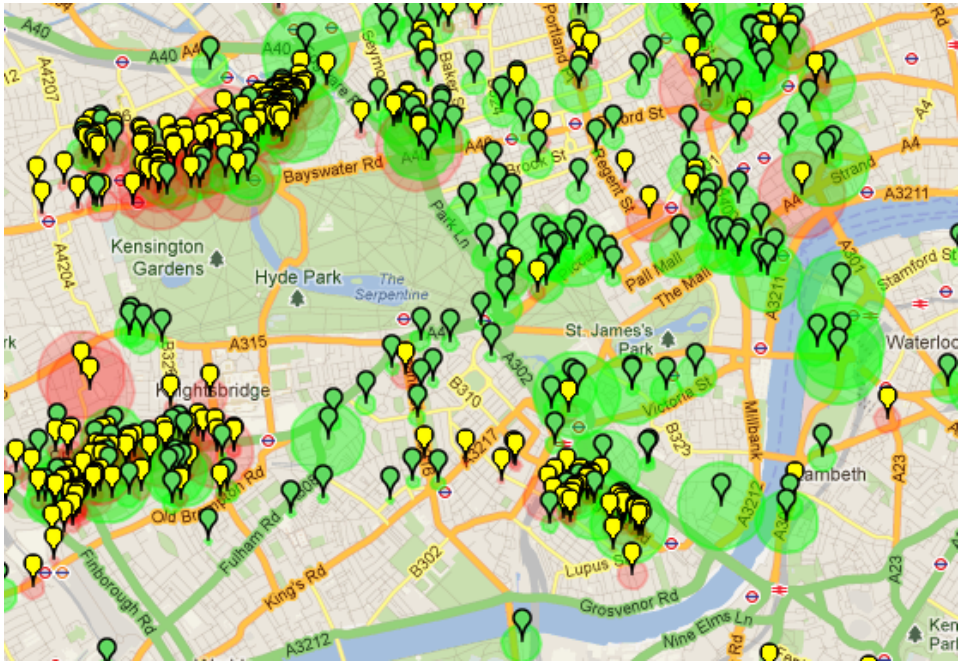


Figure 6.7: Hotels in London from booking.com marked on map with area calculations

green, while lower are coloured red. Also, colours are further divided into varying degrees of opacity. For the green circles, higher scores means a more solid circle, while for the red circles the lower the score the more solid the circle. Table 6.1 summarises the full range of colours and opacities. The colours and opacities make sure that multiple overlapping circles of the same colour result in less opacity on the overlapping areas. This gives the illusion that certain areas containing all green or all red circles are very good or very bad areas, respectively.

Figures 6.8 and 6.7 show graphical representations of the sentiment for different areas.

Table 6.1: Circle colouring for area sentiment

Score	Colour	Opacity
1.00 - 0.80	Green (#00FF00)	0.8
0.79 - 0.60	Green (#00FF00)	0.4
0.59 - 0.40	Red (#FF0000)	0.2
0.39 - 0.20	Red (#FF0000)	0.5
0.19 - 0.00	Red (#FF0000)	0.8

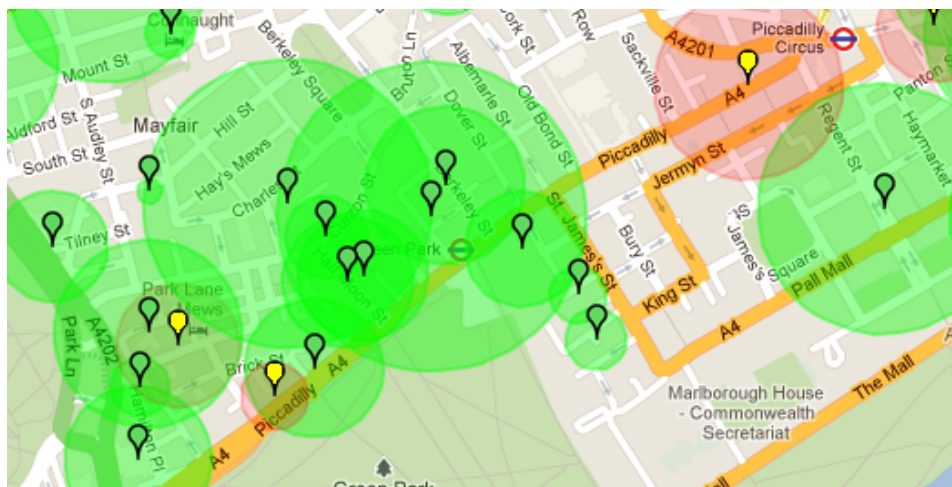


Figure 6.8: Area marked as relatively positive

### 6.2.5 Graphing Changes

To plot sentiment changes on graphs, we used a jQuery plugin called Flot.<sup>8</sup> Figure 6.9 shows what a graph may look like when showing changes on a month by month basis. The x-axis shows dates and is dynamically determined based on the dates the reviews were written. It starts at the earliest review and ends at the last. The y-axis shows score values between -1 and 1 and is static. The data on the graphs show:

- **Sentiment Score:** The calculated sentiment scores averaged over all reviews within each month. Always between 0 and 1.
- **Actual Score:** The actual score review authors gave the hotel. Always between 0 and 1.
- **Sentiment Score Change:** Change to the sentiment score from month to month. Always between -1 and 1.
- **Actual Score Change:** Change to the actual score from month to month. Always between -1 and 1.

Most of the graphs have large fluctuations early on, but they stabilise more and more for the more recent periods. This is mainly due to the distribution of reviews. Earlier periods have significantly fewer reviews to base their scores on, and as such a single review has a bigger impact when averaging over equally sized periods. Table 6.2 shows the distribution of reviews on a year by year basis from 2004 up until march 2012. From 2004 up until roughly 2008, there are quite few reviews. From 2009 up until march 2012 the number of reviews increases drastically. Note that from Booking.com there are no reviews

<sup>8</sup><http://code.google.com/p/flot/>

Table 6.2: Percentage of reviews year by year

Year	TripAdvisor		Booking.com	
	London	New York	London	New York
Earlier	0.94%	1.47%	0.0%	0.0%
2004	2.91%	3.90%	0.0%	0.0%
2005	3.91%	5.83%	0.0%	0.0%
2006	5.88%	6.39%	0.0%	0.0%
2007	7.71%	8.45%	0.0%	0.0%
2008	8.52%	9.29%	0.0%	0.0%
2009	13.52%	11.99%	0.0%	0.0%
2010	18.39%	17.48%	1.16%	0.63%
2011	32.24%	28.85%	85.79%	86.16%
2012 (≈march)	5.97%	6.34%	13.05%	13.2%

before 2010. The merged data set contains roughly the average of these two for each year, although there is a slight skew in advantage tripadvisor, which has a somewhat higher average number of reviews per hotel.

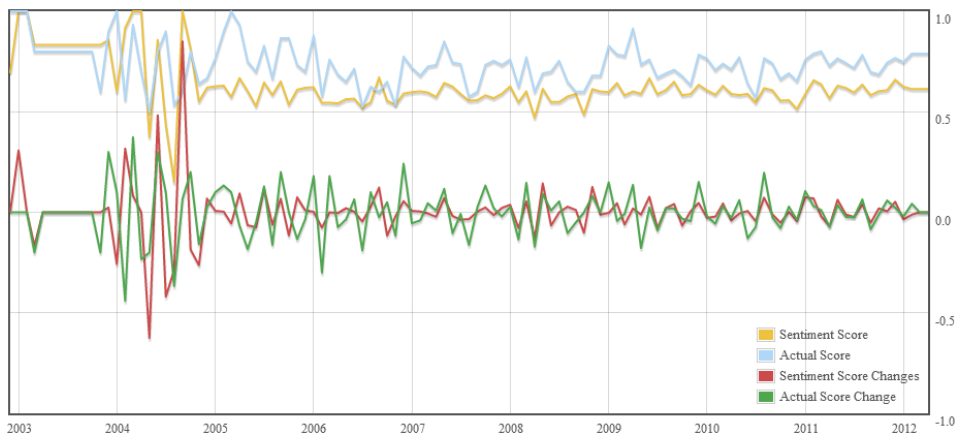


Figure 6.9: Month by month score changes

## 6.2.6 Feature Search

One of the things we wanted to test with the framework was arranging and grading hotels based on certain features. For instance emphasise hotels offering a good breakfast, or hotels with a friendly and helpful staff. This is a helpful way for users to find hotels which have the facilities that each individual user finds important. If one does not need a breakfast, but require good accessibility to nearby public transportation, it should be

Table 6.3: List of features

Feature
Breakfast
Location
Staff
Service
Clean
Internet

possible to filter out what previous users think about specific services and filter results accordingly.

These features are defined ahead of time and are generally meant to be domain specific. Table 6.3 shows the list of features we focused on that are available for the user to grade the hotels by.

In figure 6.10 one can see the format of the XML input files used in the feature search. For each hotel, all sentences in all the corresponding reviews have been analysed and a summary score for each sentence is calculated. If a sentence contains a mention of one of the predetermined features, the feature is mapped to that specific sentence.

The interesting thing about this approach is its domain independence. The predefined list of features may contain any desired feature, and can be run on almost any type of data set containing a sufficient amount of raw text. One drawback is it is somewhat difficult to take into account typographical errors. The most common mistakes may in theory be added as synonyms to existing features, but considering all alternatives may be problematic and time-consuming. However, assuming no spelling mistakes should still achieve an adequate amount of feature mentions.

```
<hotel name="City Living Schøller Hotel" pos_score="48.125" neg_score="33.375" number_features="62"...>
  <features name="location" pos_score="13.625" neg_score="5.625" number="17">
    <sentence pos_score="1.0" neg_score="0.125">the location was good</sentence>
    <sentence pos_score="0.75" neg_score="0.0">good location in the center of trondheim</sentence>
    <sentence pos_score="0.25" neg_score="0.125">it's location and the price</sentence>
    <sentence pos_score="0.75" neg_score="0.0">good location in center of trondheim</sentence>
    <sentence pos_score="1.25" neg_score="0.25">extremely convenient location</sentence>
    ...
  </features>
  ...
</hotel>
```

Figure 6.10: XML format for feature scores

The look of how the features are presented in our test framework is shown in figure 6.11 and 6.12.

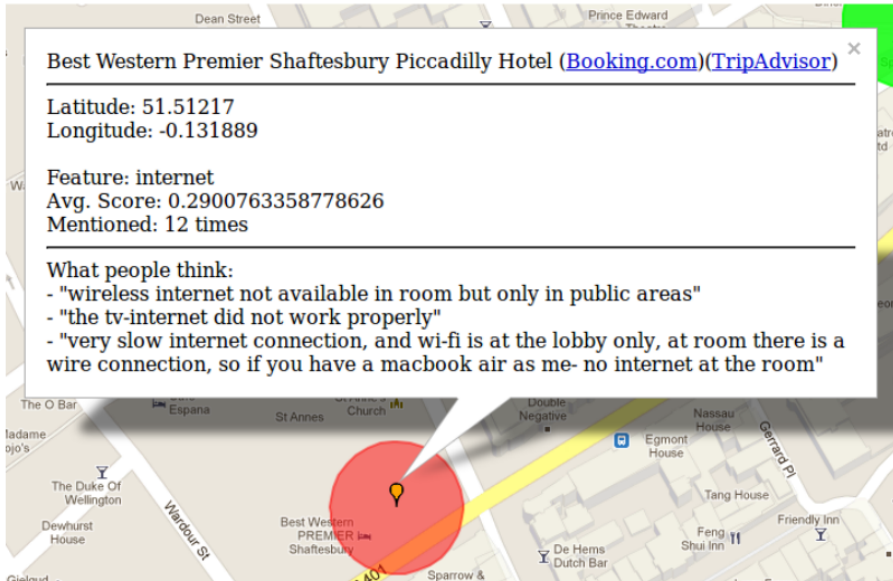


Figure 6.11: Selected feature "internet" with negative sentiment

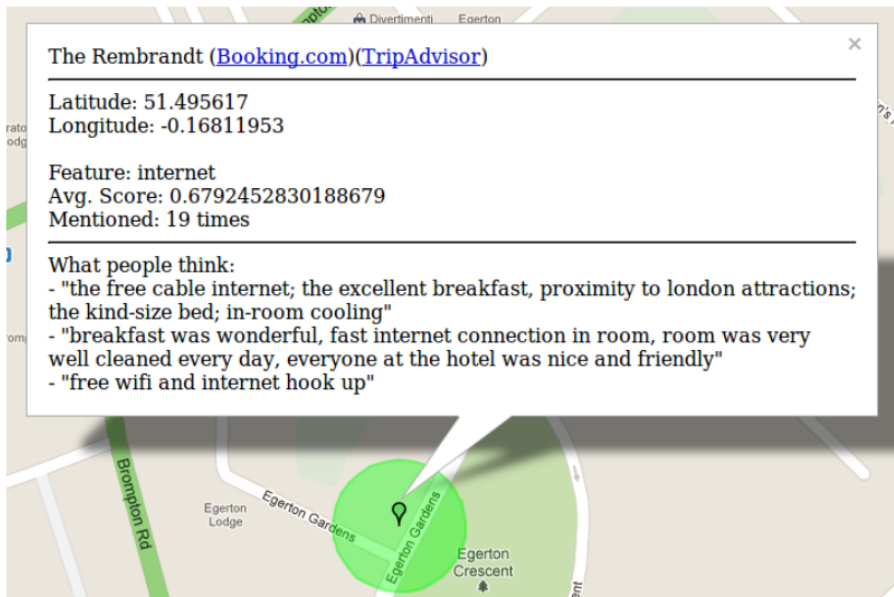


Figure 6.12: Selected feature "internet" with positive sentiment



## Chapter 7

# Experiments and Results

This chapter presents results from the prototype. It is divided into separate sections depending on the type of data presented. First are the results testing the accuracy of SentiWordNet and machine learning using naive Bayes and the dynamic language model classifier. After that burst detection and temporal change results, along with sentiment mapping and visualisation.

All the results are based on a subset of data, selected for testing based on location. The locations used are the same as the ones used as examples in previous chapters, namely Athens, Dubai, London, New York City and Paris. Trondheim was occasionally also used for quick, small scale testing. The locations were selected due to being spread out across the world, and having varying amounts of reviews. London, for instance, has a high amount of reviews from both Booking.com and TripAdvisor. New York has many reviews from TripAdvisor, but not so many from Booking.com. Dubai and Paris all have an average amount of reviews, while Athens is on the low side. This makes it possible to limit data set bias in comparisons.

## 7.1 SentiWordNet Accuracy

This section details the accuracy of SentiWordNet when tested on our data set. The reason for this testing is to determine if SentiWordNet works well enough for our purpose.

Previous studies which have incorporated SentiWordNet have generally achieved accuracies of around 65-70%. In a 2009 study by Ohana and Tierney, they used SentiWordNet to classify a data set of movie reviews [44]. The data set consisted of 1000 reviews. They concluded that SentiWordNet was quite accurate, with an average accuracy of around 65%. However, their data set was arguably not very large.



Table 7.1: SentiWordNet accuracy results

Source	Location	# correct	# reviews	% correct
TripAdvisor	London	133,492	167,655	79.62
	New York	130,049	158,950	81.82
	Athens	14,479	17,579	82.37
	Paris	99,776	122,883	81.20
	Dubai	29,327	34,016	86.22
Booking.com	London	100,082	147,076	68.05
	New York	27,059	39,485	68.53
	Athens	8,476	11,327	74.83
	Paris	40,812	58,363	69.93
	Dubai	26,683	37,628	68.26

### 7.1.1 Standard SentiWordNet

Table 7.1 shows the resulting accuracy from running SentiWordNet on our range of data sets. Generally, reviews from TripAdvisor achieved an accuracy of around 80%, which is a very good result. It performed slightly worse with Booking.com, being correct slightly below 70% of the time on average. This is more in line with previous studies, and was closer to the expected result. Although compared to the results from TripAdvisor, it is slightly lower than desirable. However, it was not entirely unexpected that TripAdvisor reviews were easier to determine than Booking.com reviews. Reviews from TripAdvisor generally consist of full sentence texts with multiple paragraphs, and therefore contain a lot of text for SentiWordNet to work with. Booking.com reviews are more often than not key word based in comparison. Along with the innate split between pros and cons, this often results in short summaries of the things that went well, but longer, more detailed texts with the negative things, or vice versa. Even comments such as "Nothing in particular" may exist for one of the sentiments. This can skew the ratio of positive to negative text, and may for instance make the text as a whole seem negative, even though the opposite might be true.

### 7.1.2 Simplified Lesk

The standard SentiWordNet implementation simply selects the most commonly used definition of a word. Most of the time this is fairly accurate. However, some words have several very different definitions. In an attempt to achieve a higher degree of accuracy, a simplified Lesk algorithm was implemented. The Lesk algorithm attempts to find the definition which is most likely correct for the given context. It does this by matching other words in the original setting with each word in the descriptive definition. The definition with the highest amount of overlapping words is assumed to be the most likely choice. See section 3.3.2 for a detailed description of the Lesk algorithm. Table 7.2 lists the results of SentiWordNet using a simplified Lesk algorithm. Figures 7.1 and 7.2 compare the

Table 7.2: SentiWordNet accuracy results with a simplified Lesk Algorithm

Source	Location	# correct	# reviews	% correct
TripAdvisor	London	129,822	167,655	77.43
	New York	125,889	158,950	79.20
	Athens	14,145	17,579	80.47
	Paris	98,026	122,883	79.77
	Dubai	28,666	34,016	84.27
Booking.com	London	99,294	147,076	67.51
	New York	26,712	39,485	67.65
	Athens	8,285	11,327	73.14
	Paris	40,343	58,363	69.12
	Dubai	25,095	37,628	66.69

accuracy from the Lesk algorithm versus the standard SentiWordNet algorithm.

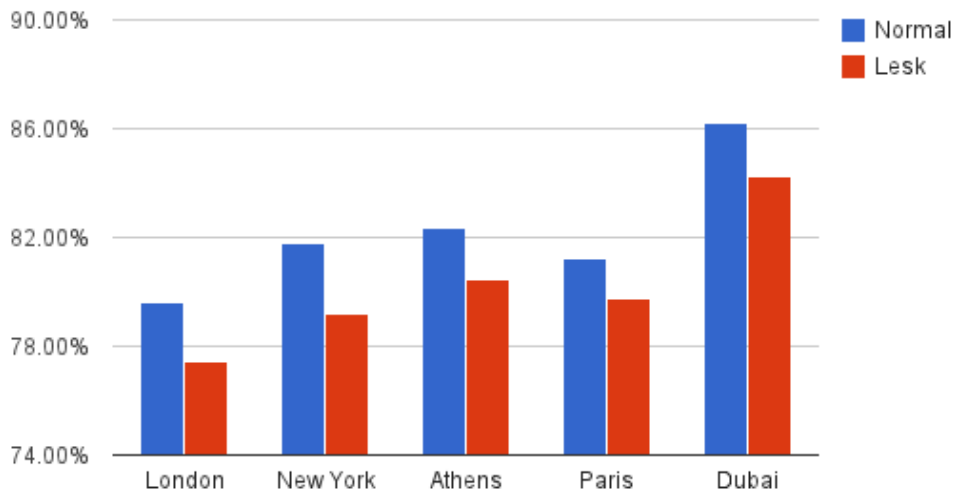


Figure 7.1: Accuracy of the Lesk algorithm vs Standard SentiWordNet for TripAdvisor

As one can see from figure 7.1 and 7.2, the Lesk algorithm performs worse than one that simply selects the most commonly used definition of a word. This was a bit surprising, but could possibly be explained by the simplicity of the Lesk algorithm. There are many different fine-tuning tools to adapt an improved Lesk algorithm for better results. We did, however, decide not to focus on this, since selection of the most commonly used definition of a word performed very well.

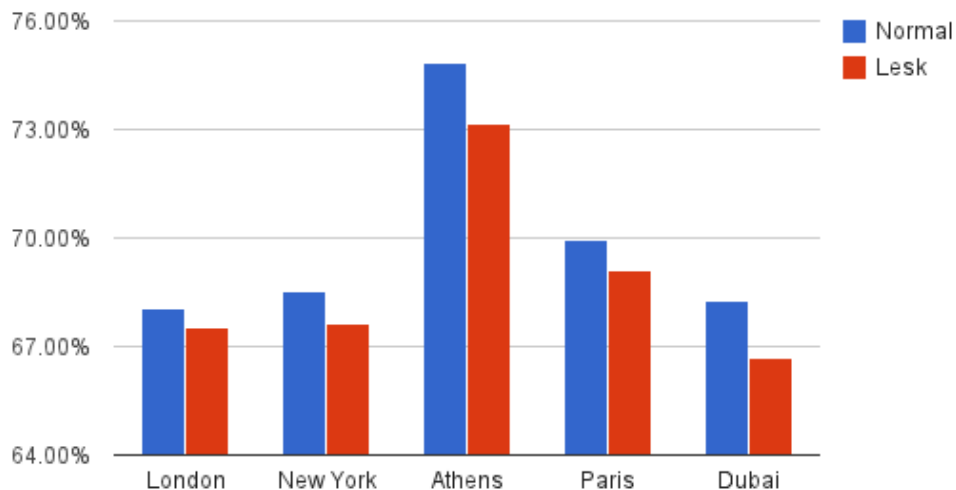


Figure 7.2: Accuracy of the Lesk algorithm vs Standard SentiWordNet for Booking.com

### 7.1.3 Five Categories

We also tested the SentiWordNet results on five different categories: strong positive, weak positive, neutral, weak negative, strong negative. The results can be seen in table 7.3. More categories generally means increased difficulty in achieving higher amounts of accuracy, especially when categorising based on numeric values. This is due to the way categorisation occurs. Different methods and techniques have different ways of estimating category values. Placing them within the same range of values provides a way to compare them, but with smaller ranges for each category (due to more categories), getting matches naturally becomes harder [48].

Table 7.3: SentiWordNet accuracy results with five categories

Source	Location	# correct	# reviews	% correct
TripAdvisor	London	103,326	167,655	61.63
	New York	104,640	158,950	65.83
	Athens	11,544	17,579	65.67
	Paris	78,750	122,883	64.09
	Dubai	24,724	34,016	72.68
Booking.com	London	63,396	147,076	43.10
	New York	18,185	39,485	46.06
	Athens	6,503	11,327	53.44
	Paris	26,850	58,363	46.01
	Dubai	16,783	37,628	44.60

Table 7.4: Result table for machine learning algorithms

Source	Algorithm	Correct	Total	Percentage
TripAdvisor	Naive Bayes	8,924	10,000	89.24%
	Dynamic LM Classifier	9,003	10,000	90.03%
Booking.com	Naive Bayes	6,411	10,000	64.11%
	Dynamic LM Classifier	6,592	10,000	65.92%

Table 7.5: Result table for machine learning algorithms with five categories

Source	Algorithm	Correct	Total	Percentage
TripAdvisor	Naive Bayes	5,742	10,000	57.42%
	Dynamic LM Classifier	5,712	10,000	57.12%
Booking.com	Naive Bayes	4,408	10,000	44.08%
	Dynamic LM Classifier	4,618	10,000	46.18%

## 7.2 Machine Learning

We also tested some machine learning techniques to see how they performed versus SentiWordNet. We used 20 000 reviews as training data and tested with 10 000 reviews. The results can be seen in table 7.4. Despite good results, we do not get the same polarity degree as in SentiWordNet. SentiWordNet is unique since it has an unlimited amount of polarity degree because of its score-based categorisation. However, we did test machine learning with five categories, to get some degree of polarity. The results from five categories are seen in table 7.5.

Machine learning does perform very well on few categories, usually better than SentiWordNet. Nevertheless, we decided to go forward with SentiWordNet because of its unlimited polarity degrees and the polarity degree results being better than machine learning.

## 7.3 Burst Detection

The main point of burst detection is to try to detect abnormal changes in hotel reviews, and then try to analyse why those changes have occurred. But first we have to decide what constitutes a burst. We define a burst as a sudden change in review score exceeding a given threshold. In our burst detection algorithm, different threshold has been tested, and the results can be seen in figure 4.6. The algorithm is described in 4.7.1.

The results can be seen in table 7.6. The algorithm has a 51.75% correctness with a burst defined threshold of 1.0 grade. However, if one increases the threshold, it has higher degree of correctness. This is because we set the definition of what a burst is by a threshold, and then compare our sentiment grade score with the actual one. So, if there is greater changes

in our sentiment score, there is also a higher possibility that that is the case in the actual score. With a 2.0 threshold it reaches a correctness of 64.17%, but finds substantial fewer bursts.

When the bursts were found, we had to find out why those changes had occurred. This was done by implementing a simple feature extraction where we grouped two and two words together, namely adverbs and verbs together in pairs and adjectives and nouns in pairs. Some examples can be seen in table 7.7 and 7.8. Table 7.7 contains some features from user reviews about the hotel Grand Midwest Express Hotel Apartments in Dubai. The period the reviews are taken from is July of 2011, and the average score in that month decreased by around 1.15 points from June 2011. Some of the features from June is shown in table 7.8. As can be seen in the tables there is a more mixed response in the month of June. The actual score for that month was 6.70 based on 9 reviews. In July 2011 the average score was 5.55 based on 13 reviews. The reason for the decline in its average monthly grading score seems to be that some of the customers have experienced insects or rats in the hotel.

Our main goal of burst detection was to see if we could detect big changes, and potentially outer, external factors, in peoples attitude towards a hotel. However, this proved to be very difficult mainly because of too little data. Even though we combined reviews from both Booking.com and TripAdvisor, we still only got between 5-15 reviews per month. If we had more data it could also be interesting to look at smaller periods of time, for instance weekly or daily data.

## 7.4 Visualising Temporal Sentiment

Following are the results from the prototype and the attempts to visualise sentiment and temporality. The section is divided into multiple sections based on the prototype function the results stem from.

### 7.4.1 Temporal Sentiment

The graphs in this section show sentiment changes over time. The time frame depends on the input reviews. The value of a graph is closely linked to the amount of reviews the data is based on. A single review should ideally have little impact on overall scores. Stable averages over time are most useful, as they make it easier to visualise possible bursts. This is a fairly natural consequence of more data giving higher confidence in the results.

Figure 7.3 is a good example of a hotel with a very consistent and high sentiment. 1891 reviews are used in this graph, and actual and sentiment scores both fluctuate very little and follow each other closely. The reviews are clustered around a rather short period of time (March 2010 - April 2012), resulting in a more stable average result. No bursts are

Table 7.6: Burst detection results with different thresholds

Threshold	Location	# correct	# bursts	% correct
1.0	London	380	712	53.37
1.0	New York	239	389	61.44
1.0	Athens	39	90	43.33
1.0	Paris	232	478	49.79
1.0	Dubai	125	246	50.81
1.0	Average			51.75
1.5	London	74	137	54.01
1.5	New York	44	66	66.67
1.5	Athens	12	23	52.17
1.5	Paris	68	124	54.84
1.5	Dubai	34	61	55.74
1.5	Average			56.69
2.0	London	18	28	64.29
2.0	New York	12	19	63.16
2.0	Athens	4	7	57.14
2.0	Paris	18	32	56.25
2.0	Dubai	8	10	80.00
2.0	Average			64.17

Table 7.7: Features from Grand Midwest Express Hotel Apartments in Dubai from a positive burst month

Feature	# mentioned
dangerous toilet	2
poor insect	1
mouldy bed	2
infested hotel	2
unprofessional staff	1

Table 7.8: Features from Grand Midwest Express Hotel Apartments in Dubai month before burst

Feature	# mentioned
shabby bed	1
clean rats	1
friendly staff	2
limited parking	1
good room	2

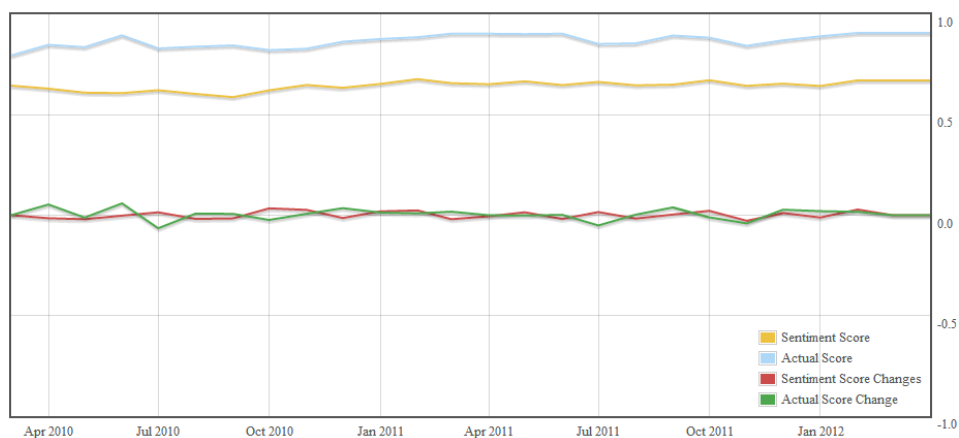


Figure 7.3: Graph from 1891 reviews about Park Plaza Westminster Bridge in London, UK

found in this graph however, but from a "user looking for a good hotel" point of view, this hotel seems like an excellent choice.

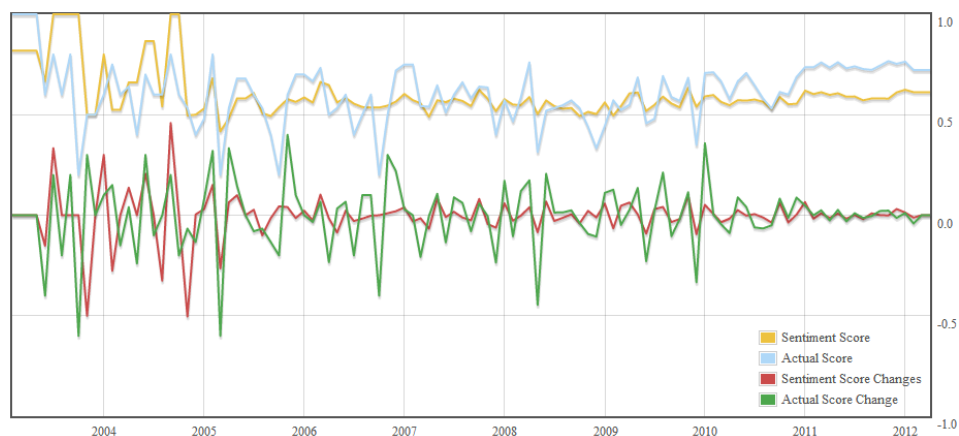


Figure 7.4: Graph from 4250 reviews about St Giles Hotel & Leisure Club in London, UK

Figure 7.4 on the other hand is based on 4250 reviews. At first look this is very high number, and should result in a high amount of confidence regarding the results. However, the data is based on a larger period of time (Early 2003 until March 2012), resulting in fewer reviews month by month. The amount of reviews is still enough to be interesting, but the scores are very erratic, even when considering the lower amount of reviews at the beginning of the graph. This leads to the conclusion that the sentiment of people staying at this particular hotel is very divided.

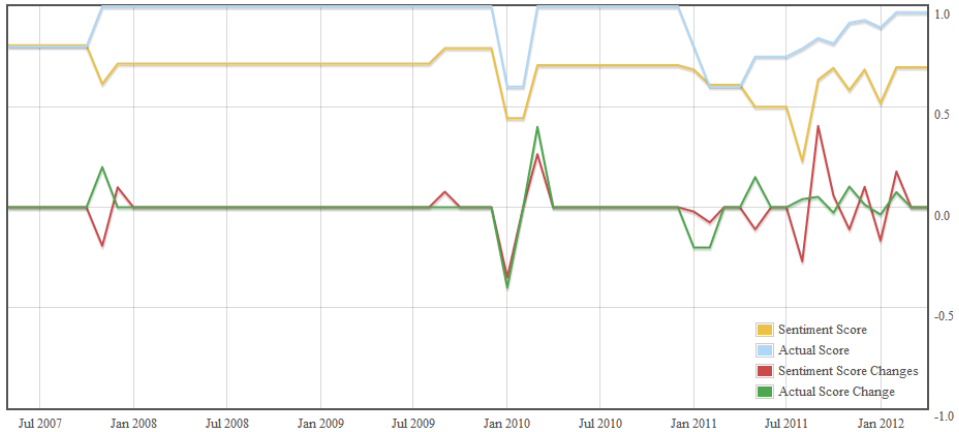


Figure 7.5: Graph from 40 reviews about San Domenico House in London, UK

Not all graphs are particularly useful at all. Figure 7.5 for instance holds little value. Due to the very limited amount of reviews found for this particular hotel, with only 40 reviews from May 2007 until March 2012, the temporal sentiment does not give any useful conclusions about sentiment changes. Every month a review appears, the scores fluctuates wildly, making it very difficult to assess any accurate sentiment.

### 7.4.2 Rating and Sentiment Correlation

Figures 7.3, 7.4 and 7.5 show examples of different aspects of temporal sentiment. Although they result in different conclusions, it is interesting to note the similarities between them. With all the graphs, the correlation between actual score and sentiment score is very consistent. As a reminder, actual score is based on the score set by each reviewer, while sentiment score is calculated based on the content of each review. These two graphs follow each other very closely. One may have larger fluctuations than the other, but overall if one goes down, so does the other. This indicates a clear correlation between what reviewers write and how they rank their visits.

Sentiment score is consistently a few points lower than the actual score, but this is to be expected. Sentiment score is calculated using SentiWordNet, which utilises the whole range of scores from 0 to 10. Actual scores given by reviewers are somewhat skewed toward the higher end of the scale. Figure 5.15 highlights this issue. Very few reviewers rank their stays lower than 4. A score of 4 is usually complemented by very negative review texts. Therefore sentiment score will always be somewhat lower than the actual score.



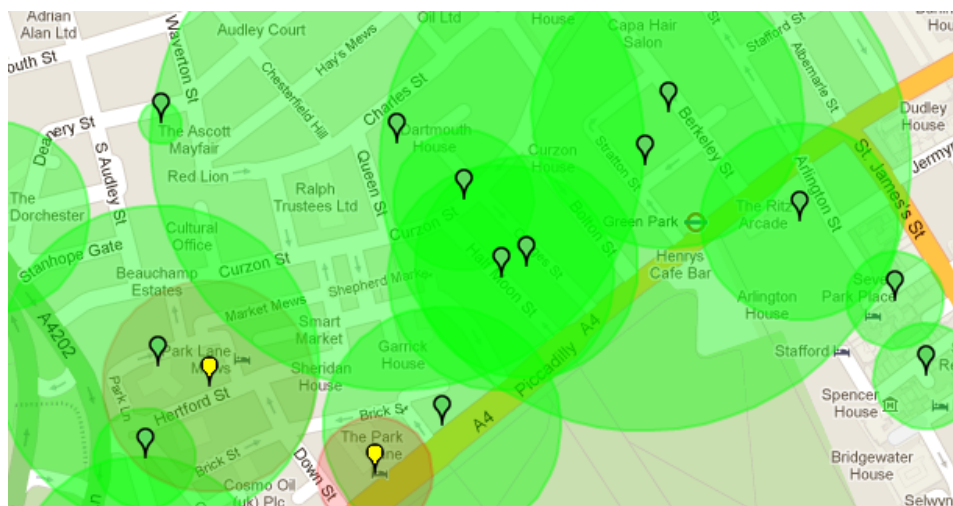


Figure 7.6: Area with a mostly positive sentiment

### 7.4.3 Area Sentiment

This section highlights some of the data found when visualising the sentiment for bounded areas. Figure 7.6 shows an area with a cluster of hotels with high average sentiment scores. Figure 7.7 an area with hotels with lower average sentiment scores. In figure 7.8 one can see an area with containing hotels with a wide range of sentiment scores. To give a feel of how one may evaluate if an area is positive or negative compared to others, figure 7.9 shows a larger area containing multiple clusters with different sentiments. Among others, a larger negative cluster can be seen to the left and centre of the figure, and a smaller positive cluster in the lower right corner.

### 7.4.4 Feature Search

The Feature Search functionality gives the user the ability to filter hotels based on the sentiment regarding a given feature. The features are predetermined and domain specific.

Table 7.9: Manually feature results based on whole sentences and sentence clauses

Sentence split	# Correct	# Total	% Correctness
Punctuation	80	100	80%
Clauses	89	100	89%

We also did some minor testing on the accuracy of the features. We manually checked 100 random feature scores based on whole sentences and 100 random feature scores split on

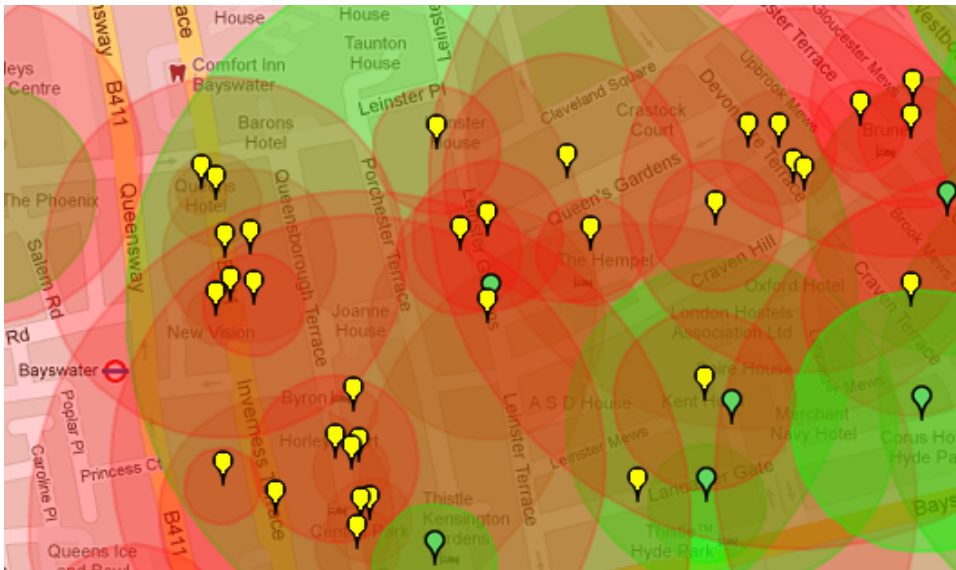


Figure 7.7: Area with a mostly negative sentiment

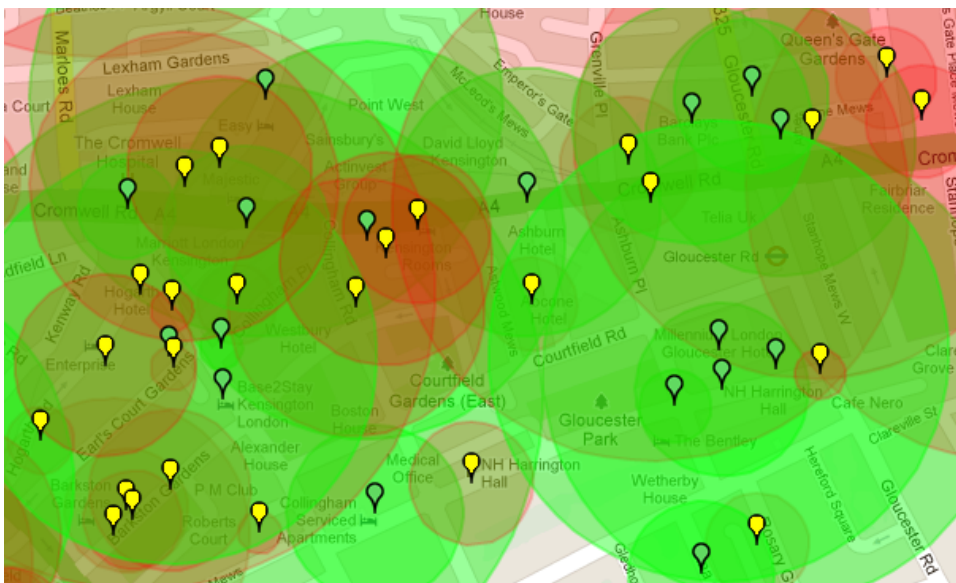


Figure 7.8: Area with no clear one-sided sentiment

clauses. Then we decided whether the feature score given corresponded with our judgement about the individual sentences the feature score is based on. This is of course very subjective, but was done to get an indicator of the correctness. The results can be seen in

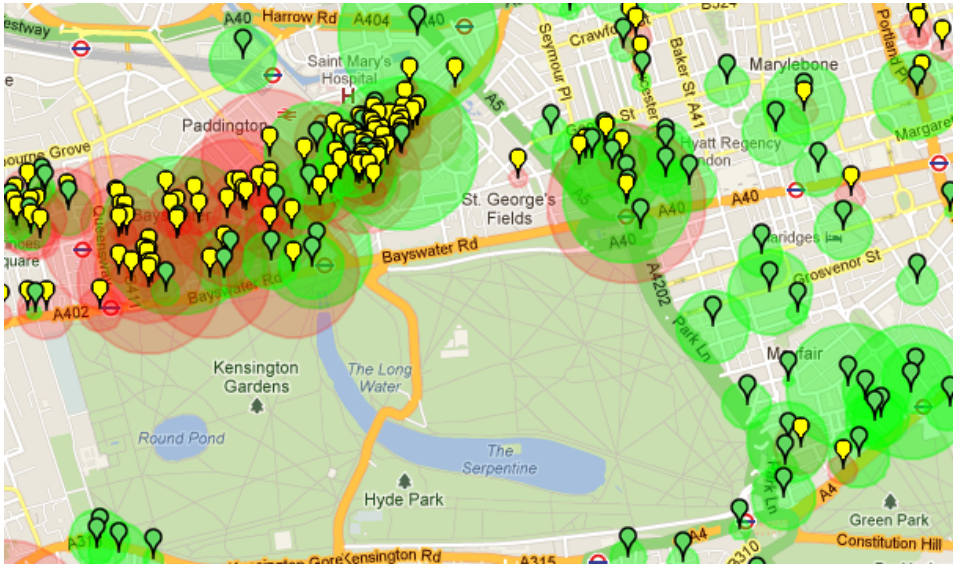


Figure 7.9: Overview of different sentiments in neighbouring areas

table 7.9.

# Chapter 8

## Discussion and Conclusion

This chapter aims to give a discussion of the results presented in chapter 7 together with a conclusion. In the final section of this chapter possible further work is presented.

### 8.1 Result Summary

Chapter 7 presented the results from the prototype and our hypotheses. The results can be divided into three major areas: burst detection and temporal sentiment changes, feature selection, and sentiment visualisation and mapping. This section therefore first discusses the results from these parts separately, as well as an evaluation of the different data set sources.

#### 8.1.1 Burst Detection and Temporal Sentiment

The goal of the burst detection algorithm was to detect sudden and lasting changes in sentiment. Results can be found in section 7.3. Several bursts were found in the data sets. However, the value of these results was rather limited. Most of the bursts seemed to stem from normal variations in the SentiWordNet calculations. The main reason for this is likely due the limited amount of reviews. Although some hotels had thousands, roughly 95% had less than 500, and 45% had less than 50 (figure 5.16). Spread out over 2-8 years, this does not result in a high amount of data points for a proper temporal evaluation. Finding useful bursts containing enough mentions on a specific topic, with enough reviews before, during and after the burst, was therefore quite difficult. The bursts we did find did not provide much useful data about why the sentiment changed either. The accompanying review texts mostly contained generic terms and consequently the same terms and features as before the burst took place. However, some interesting points were still identified. A small subset

of the detected bursts had some features which indicated a cause behind the sudden change in sentiment. With a larger data set and further tweaking of the algorithm, positive results are not unlikely.

## 8.1.2 Feature Selection

The initial idea for the feature extraction was to identify sentences containing each feature and determine scores based on these. Mostly this approach worked fine. However, some sentences contain multiple clauses, and not all parts are necessarily relevant. Sometimes they branch of into completely different topics. This results in quite a bit of noise when it comes to calculating the sentiment scores. Consider for instance the following sentence from a Booking.com/TripAdvisor review, and the selected feature "staff":

"[..](1) location is good, (2) staff is very helpful and friendly, (3) rooms are modern and functional, (4) roof bar is nice to have some drinks and breakfast with super views of acropolis... (5) a perfect stay[..]"

This sentence contains multiple clauses containing sentiments about several features (numbered 1-5 to simplify referring). In this example, only the second part, "staff is very helpful and friendly", relates to the staff sentiment. The rest of the sentence is about location, decor and available facilities. The sentiment scores for these additional clauses are not relevant with regards to the intended score. As such, one can not be certain of the accuracy of the rankings.

An alternative to full sentence analysis was to split sentences further by seeing each clause as a separate entity. This increased the likelihood that the part containing the feature was indeed about the desired feature. Most of the noise found by the previous approach disappeared. In the mentioned example, only the part "staff is very helpful and friendly" would be used in staff sentiment calculation. The drawback of this new technique was of course that the calculations lost some of clauses which were indeed relevant to the feature. Looking at the same sentence, but from a "location" point of view, one would find the first clause "location is good". Initially this seems fine. However, the last part of clause 4, "[roof bar]with super views of acropolis", can arguably also be considered quite relevant regarding the location. With this new approach, that information would be lost.

The results from these two approaches are found in section 7.4.4. As seen in table 7.9, splitting on clauses performs better. The difference is not huge though, and the correctness for both is very high, between 80%-89%. However, these results are based on a very small data sample, and it will have a high potential error rate and therefore should be interpreted with care. The reason for the small amount is simply because we had to do it manually to be able to judge the scores correctly, and this does take a substantial amount of time. Because of the limited time available for this project, we decided to limit the test data. Nevertheless, it might be an indicator that the feature search works to some extent.

It should also be pointed out that the feature search can sometimes be quite a bit wrong, especially on smaller data sets. Figure 8.1 shows a hotel which has gotten a very positive

Nova Hotel ([Booking.com](#))([TripAdvisor](#))

---

Latitude: 63.43268

Longitude: 10.400764

Feature: internet

Avg. Score: 0.81818181818182

Mentioned: 1 times

---

What people think:

- "internet is slow as heck, there is no toilet paper, soap, shampoo, mold around floor, wall paper peeling off, rooms super tiny, wires painted over all over the walls"

Figure 8.1: Selected feature 'internet' with positive score, but clearly negative content

score for their internet (0.81), but reading the sentence the score is based on clearly states the internet is bad, along with several other negative sentiments.

### 8.1.3 Sentiment Visualisation and Mapping

Results are in 7.4.3 and 7.4. With sentiment visualisation and mapping the point was to determine average review scores for each hotel and mark each hotel on a map. Using the average scores, the hotel markings differ in colour. Green at the top end of the scale, with dark red at the lower spectrum. The purpose of this approach was to see if it was possible to identify specific areas with a very high degree of either positive or negative sentiment.

With our prototype today, one can easily identify "good" and "bad" areas by looking at the colours on the map, but there is no automatic way of doing this. This is something that can be added in the future, and possibly creating a ranking of the best areas in the city.

### 8.1.4 Data Set Comparison

The data set is gathered from two main sources, TripAdvisor and Booking.com. General statistics regarding these two sources are presented in section 5.6. Overall the data from these two sources are quite similar. They both consist of user generated content with a short amount of text and an arbitrarily determined overall score. TripAdvisor reviews contain slightly more full-sentence text, while Booking.com consists mostly of short summaries and keywords. TripAdvisor contains more reviews for most locations. Scores from both sources are shifted toward the higher end of the scale, but Booking.com scores are noticeably higher, with more than 80% of review scores being higher than 6.0 on a scale from 0-10.

It should also be noted that from a control set of 662,991 reviews from Booking.com, the lowest score from this set was 2.5. No reviews at all were rated less than 2.5. Part of this likely due to the way scores are set on Booking.com. Scores are an average of a range of subscores. Each reviewer must rate each of six different criteria individually, including comfort, location and value for money. This would likely increase the average scores somewhat, due to most people finding something they like about a part of their stay, and the extremely negative scores therefore are not so common. However, one would still assume that some very angry customers would rate all the subcriteria with the bottom score, at least when checking hundreds of thousands of reviews. It is therefore possible Booking.com employ some sort of filtering mechanism, and ratings with all negative scores are regarded as spam.

## 8.2 Conclusion

Opinion mining, especially with a temporal aspect, is a fairly new field of study. There are a limited amount of well-tested techniques. The main goal of this project was therefore to create a prototype and with that evaluate techniques for presenting the temporal aspect of a set of opinionated user-generated text. Due to the large amount of authors, the quality, quantity and subjectivity of the text would vary greatly. As such, part of the goal was to identify general-purpose techniques that were domain independent and provided satisfyingly consistent results. As detailed in the previous paragraphs, the burst detection and the temporality of hotel reviews were not as successful as hoped. A number of bursts were found, but extracting features from around the bursts did not provide a consistent and clear insight into why the burst occurred. Visualisation results were more successful. Here we were able to identify areas containing clusters of positive and negative hotels in a clear interface. Filtering hotels based on specific features also functioned well, and gave users an easy way to find hotels with a focus on specific terms.

The prototype we have created is designed for evaluating customer sentiment regarding hotels. The underlying techniques used are fairly general, and could be applied to any relevant and sufficiently large data set containing sentiment data. To fully utilise the prototype, the only requirement is that the data set contains both a geographical and a temporal aspect. However, all techniques and functions can also be used separately. We collected a relevant data set for this purpose, which incorporates all these aspects using reviews in the hotel domain. The most relevant multipurpose functions are perhaps the feature search and burst detection. Feature search and extraction may be used on any opinionated data set, although presentation of results in the prototype would need altering to some sort of ranked list, for instance. Burst detection may be used on any temporal data set, and might even function better on other types of data sets, for instance movie or video game reviews.

```

<monthly>
  <score year="2009" month="9" reviews="1">1.0</score>
  <score year="2009" month="10" reviews="1">0.4</score>
  <score year="2009" month="11" reviews="0">0.4</score>
</monthly>

```

Figure 8.2: XML score structure in prototype

## 8.3 Future Work

This section details some potential improvements to the work done and interesting directions the project could take with more time available.

### 8.3.1 Prototype Performance

The main limiting factor for the performance of the prototype is the size of the XML files and the available bandwidth of the user. Average score XML files can be reduced in size by removing hotels with less than N reviews. Also, month by month scores take quite a bit of space with the original format. Changing this to a something simpler could also significantly reduce file sizes. How the scores are structured now can be seen in figure 8.2. An example of a potential improvement is shown in figure 8.3. Here each score tag has a start and end attribute, and all the scores are listed with number of reviews separated by "\_". In average 63 lines of score is added to each hotel, while the improved only will have one row. On the other hand the one row will have more text, but will still save around 25 characters per line. So in average 1575 characters less per hotel.

```

<score start="2009-09" end="2009-11">1.0_1, 0.4_1, 0.4_0</score>

```

Figure 8.3: A possible improvement of XML score structure

Performance could also be significantly enhanced by implementing a database (SQL, for instance), and importing the XML datasets. This way one could retrieve only the information needed at any given point, reducing the initial cost of downloading potentially large XML files. For instance, the monthly scores used for graphing temporal changes could be retrieved when the user wants to see the graph. Currently, this data is downloaded and stored for all the hotels at a given location. This would however result in more requests sent between server and client, but should not be much of an issue. The main reason for not implementing this already, is performance was not a priority for a majority of the projects life time. Initially the goal was just testing certain aspects and visualising data locally, and



Table 8.1: List of features

Feature	Relevant Terms
Breakfast	
Location	View, Area
Staff	Reception(ist), Employee
Service	Serve(d)
Clean	Fresh, Washed, Immaculate, Dirty
Internet	Wi-Fi, WiFi

time constraints prevented us from correcting this later on. However, for future use of the prototype, performance should be of a high priority.

### 8.3.2 Visualisation

For the moment there does not exist a way to automatically detect "good" and "bad" areas in our prototype. This is definitely functionality that could be interesting to add, because one could then rank different areas, look at area rankings over time and try to detect why whole areas have had a change in sentiment automatically.

Also, one can improve the map markings by having more of a colour transition instead of separated circles. Today we only have individual circles, where their size depend on how strong the sentiment is. These colours do overlap in crowded hotel areas, and do show area sentiment, but it would have been better to have an algorithm that can sketch out colour areas with a more natural form. This could make it easier to detect the sentiment areas.

### 8.3.3 Feature Synonyms

The feature identifier and sentiment calculations use only the features found in table 6.3. For some, such as breakfast, this is sufficient. It is somewhat difficult to write about breakfast without using that specific word. However, this is not necessarily true for all features. For instance clean. When describing the cleanliness of a hotel, one might not use variations of the word clean at all. Words such as dirty, spotless or smelly are all relevant and commonly used. Implementing support for closely related terms to the list of domain dependent features would likely provide a more accurate filter for visualising sentiment regarding specific features. Adding some of the more common spelling mistakes could potentially lead to further improvements as well. Table 8.1 lists some potential synonyms one could have used to identify more mentions of specific topics. The list is based on the similar list presented in chapter 6.

### 8.3.4 Expand Data Sets

It would also be interesting to expand the data sets to services other than hotels. For example restaurants or products like music, books et cetera. What would be very interesting is to test our prototype on products that we know have an external variable that changes the average score, and see if we are able to detect it. Examples of that can be politicians that are involved in a scandal, or a book or a CD that we know contain a lot of spam reviews.

### 8.3.5 Other Pre-Processing Steps

There are also possible improvements in our pre-processing steps. For example we could include support for interjections. The problem today is that SentiWordNet does not support it, but this can potentially be added manually. The rest of the framework does support interjections, including the part of speech tagger. Of course, it will be challenging to judge interjection sentiment, but it is an area which can be interesting to see the effect.

Better word disambiguation. This is very important for calculating the sentiment scores, since it directly reflects the scores. One improvement could be to improve the Lesk algorithm. Today the implemented version is very simplified, and therefore there is a big potential to improve word disambiguation. One can of course try other algorithms than the Lesk algorithm also.

Grammatical errors and spelling mistakes. A lot of spelling mistakes exist in the collected reviews. This might limit the amount of correct tags with WordNet. However, since the content is user created and collected from third-party sites, it is not an area we have direct control over. The main alternative to correct this problem would be spell check with automatic correction on the data set, which should correct most errors. Correcting other peoples texts is an area which should be approached with caution though, as intentions may be altered.



# Appendix A

## List of Data Set Locations

Table A.1: List of crawling locations

London	Bangkok	Singapore
Kuala Lumpur	Antalya	New York City
Dubai	Paris	Istanbul
Hong Kong	Mecca	Rome
Miami	Las Vegas	Los Angeles
Barcelona	Cairo	Shanghai
Pattaya	Dublin	Bucharest
Macau	Amsterdam	Prague
Moscow	Kiev	Beijing
Vienna	Phuket	Madrid
Tokyo	Mugla	San Francisco
Orlando	Taipei	Berlin
Budapest	Rio De Janeiro	Edirne
Toronto	Stockholm	Mexico City
Seoul	Denpasar	Delhi
St Petersburg	Brussels	Warsaw
Jerusalem	Guangzhou	

Table A.1 shows all the locations crawled for data. The list is based on the top 50 location found at: <http://blog.euromonitor.com/2011/01/euromonitor-internationals-top-city-destinations-ranking.html>.



# Appendix B

## Class Diagrams

List of packages found in the code base of the opinion mining framework, along with some details and descriptions.

### B.1 Package Overview

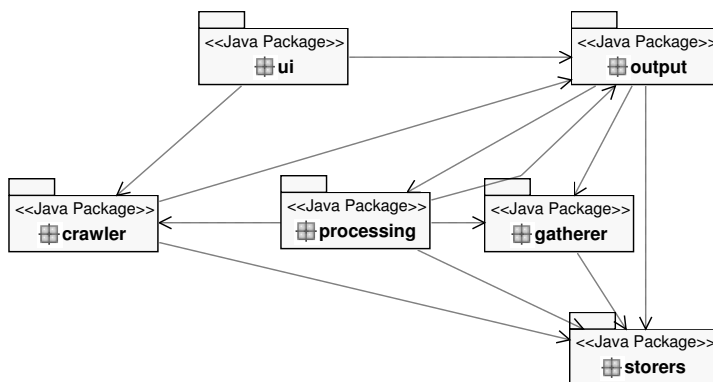


Figure B.1: Package map with dependencies

An overview of package dependencies can be seen in figure B.1.

## B.2 Crawler

Package containing classes responsible for crawling. This includes crawling TripAdvisor and Booking.com. Figure B.2 shows the package.

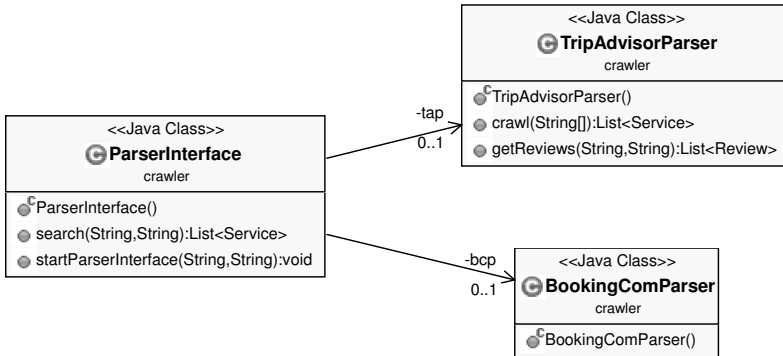


Figure B.2: Classes in crawler package

## B.3 Gatherer

The gatherer package consists of functionality to read the different XML files and store them in memory.

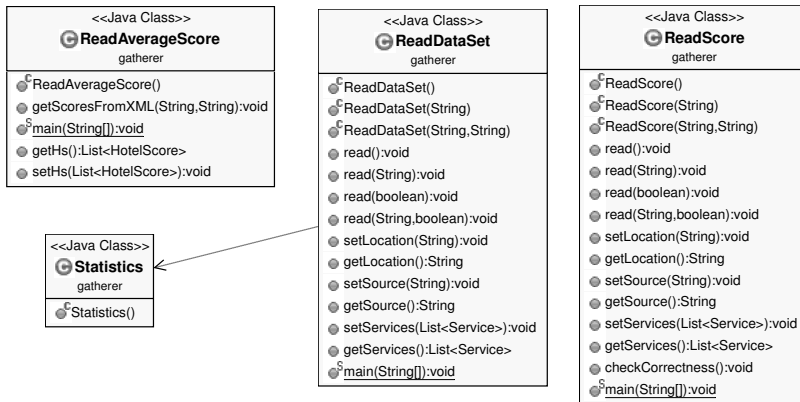


Figure B.3: Classes in gatherer package



# B.4 Output

Package containing functionality to write to XML.

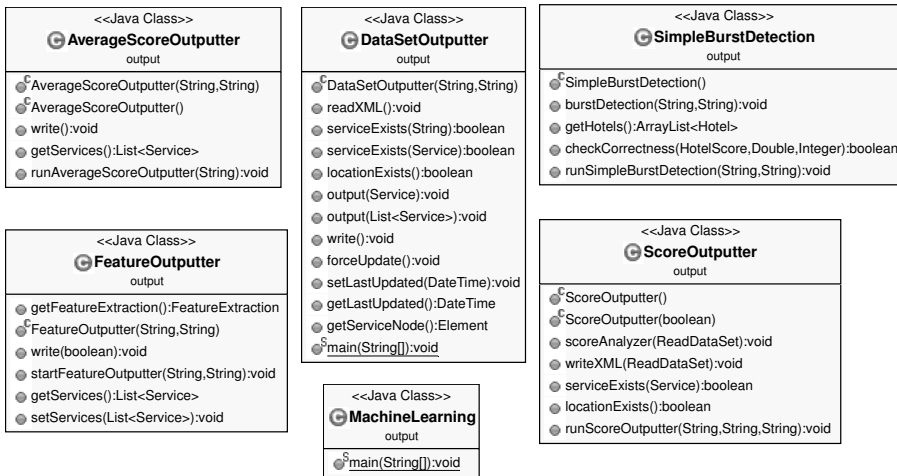


Figure B.4: Classes in output package

# B.5 Processing

This package is responsible for all the calculations done with the inputted reviews.

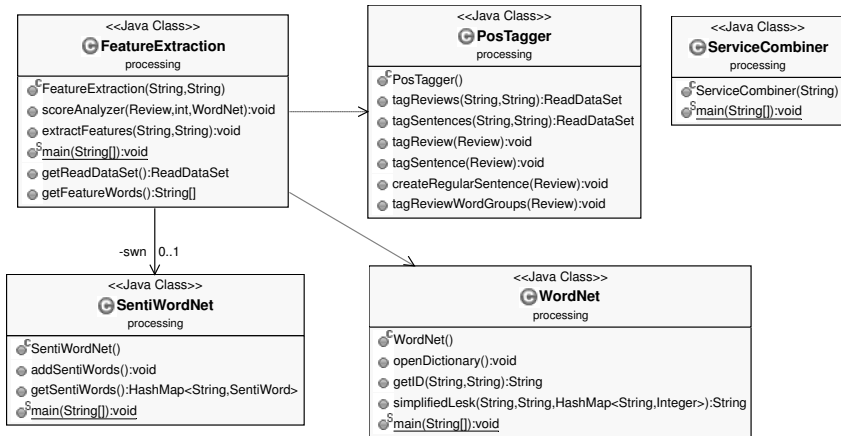


Figure B.5: Classes in processing package

## B.6 Storers

Everything read from the XML files are stored in memory in these classes.

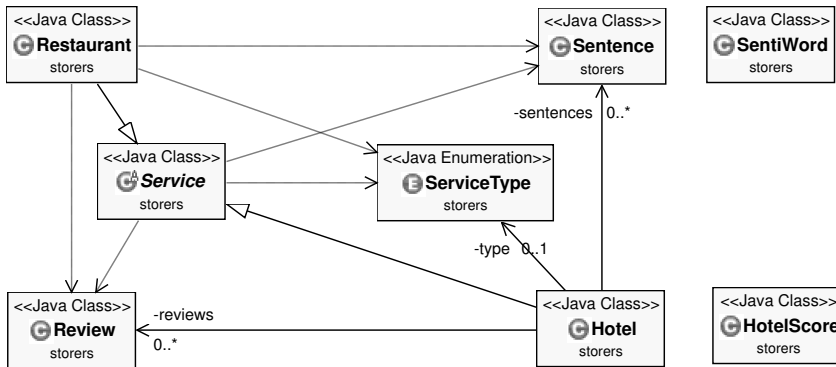


Figure B.6: Classes in storers package

## B.7 UI

Package with class for our textual user interface.



Figure B.7: Class in ui package



# Bibliography

- [1] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 2010.
- [2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
- [3] S. Banerjee and T. Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, 2002.
- [4] P. Beineke, T. Hastie, C. Manning, and S. Vaithyanathan. An Exploration of Sentiment Summarization. AAAI, 2003.
- [5] S. Blair-Goldensohn, T. Neylon, K. Hannan, G. A. Reis, R. Mcdonald, and J. Reynar. Building a sentiment summarizer for local service reviews. In *NLP in the Information Explosion Era*, 2008.
- [6] E. Brill. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Comput. Linguist.*, 21:543–565, 1995.
- [7] J. Carbonell. *Subjective Understanding: Computer Models of Belief Systems*. PhD thesis, Yale, 1979.
- [8] P. Chaovalit and L. Zhou. Movie review mining: a comparison between supervised and unsupervised classification approaches. In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, 2005.
- [9] L.-C. Cheng, Z.-H. Ke, and B.-M. Shiue. Detecting changes of opinion from customer reviews. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, volume 3, 2011.
- [10] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, 1990.

- [11] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration*, 2003.
- [12] D. Das, A. K. Kolya, A. Ekbal, and S. Bandyopadhyay. Temporal analysis of sentiment events: a visual realization and tracking. In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing, CILing'11*, 2011.
- [13] S. R. Das and M. Y. Chen. Yahoo! for Amazon: Sentiment extraction from small talk on the Web. *Management Science*, 53(9):1375–1388, 2007.
- [14] K. Dave, S. Lawrence, and D. M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, 2003.
- [15] K. T. Durant and M. D. Smith. Mining sentiment classification from political web logs. In *Proceedings of Workshop on Web Mining and Web Usage Analysis of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining WebKDD2006*, 2006.
- [16] A. Esuli and F. Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06)*, 2006.
- [17] T. Fukuhara, H. Nakagawa, and T. Nishida. Understanding sentiment of people from news articles: Temporal sentiment analysis of social events. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007.
- [18] C. K. Group. Online Consumer-Generated Reviews Have Significant Impact on Offline Purchase Behavior. 2007.
- [19] V. Hatzivassiloglou and K. R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, 1997.
- [20] V. Hatzivassiloglou and J. M. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th conference on Computational linguistics*, volume 1, 2000.
- [21] M. Hearst. Direction-based text interpretation as an information access refinement. In P. Jacobs, editor, *Text-Based Intelligent Systems*, pages 257–274. Lawrence Erlbaum Associates, 1992.
- [22] J. A. Horrigan. Online shopping. *Pew Internet & American Life Project Report*, 2008.
- [23] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

- [24] M. Hu and B. Liu. Mining opinion features in customer reviews. In *Proceedings of the 19th national conference on Artificial intelligence*, 2004.
- [25] A. Huettner and P. Subasic. Fuzzy typing for document management. In *ACL 2000 Companion Volume: Tutorial Abstracts and Demonstration Notes*, 2000.
- [26] N. Ide and J. Veronis. Word sense disambiguation: The state of the art. *Computational Linguistics*, 24:1–40, 1998.
- [27] N. Jindal, B. Liu, and E.-P. Lim. Finding unusual review patterns using unexpected rules. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010.
- [28] D. Jurafsky and J. H. Martin. *Speech and Language Processing (2nd Edition)*. Pearson Prentice Hall, 2 edition, 2008.
- [29] D. Keim. Information visualization and visual data mining. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):1–8, 2002.
- [30] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Min. Knowl. Discov.*, 7(4):373–397, 2003.
- [31] L. Ku, Y. Liang, and H. Chen. Opinion extraction, summarization and tracking in news and blog corpora. In *Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs*, 2006.
- [32] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.
- [33] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, 1986.
- [34] B. Liu. Opinion observer: Analyzing and comparing opinions on the web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, 2005.
- [35] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., 2006.
- [36] B. Liu. Opinion Mining. *Proceedings of WWW 2008*, 2008.
- [37] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press., 2008.
- [38] A. Mccallum. Text Classification by Bootstrapping with Keywords, EM and Shrinkage. In *ACL99 - Workshop for Unsupervised Learning in Natural Language Processing*, 1999.
- [39] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990.



- [40] G. Mishne and M. de Rijke. Moodviews: Tools for blog mood analysis. In *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW)*, 2006.
- [41] S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima. Mining product reputations on the web. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.
- [42] T. Nasukawa and J. Yi. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the Conference on Knowledge Capture (K-CAP)*, 2003.
- [43] K. Nørvåg and O. K. Fivelstad. Semantic-based temporal text-rule mining. In *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing*, 2009.
- [44] B. Ohana and B. Tierney. Sentiment classification of reviews using SentiWordNet. In *9th. IT & T Conference*, 2009.
- [45] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, 2011.
- [46] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, 2004.
- [47] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 2005.
- [48] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. *Foundation and Trends in Information Retrieval* 2, pages 1–135, 2008.
- [49] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, volume 10, 2002.
- [50] A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 2005.
- [51] D. R. Radev, E. Hovy, and K. McKeown. Introduction to the special issue on summarization. *Comput. Linguist.*, 28(4):399–408, 2002.
- [52] I. Rish. An empirical study of the Naive Bayes classifier. In *Proceedings of IJCAI-01 workshop on Empirical Methods in AI*, 2001.
- [53] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here. In *Proceedings of the IEEE*, 2000.
- [54] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003.

- [55] W. Sack. On the computation of point of view. In *Proceedings of AAAI*, 1994. Student abstract.
- [56] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34:1–47, 2002.
- [57] R. Sinnott. Virtues of the haversine. *Sky and Telescope*, 68(2):159, 1984.
- [58] B. Snyder and R. Barzilay. Multiple Aspect Ranking using the Good Grief Algorithm. In *Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (HLT-NAACL)*, 2007.
- [59] R. M. Tong. An operational system for detecting and tracking opinions in on-line discussion. In *Proceedings of the Workshop on Operational Text Classification (OTC)*, 2001.
- [60] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1, 2003.
- [61] P. Turney, C. Canada, M. Littman, and F. A. H. billion-word Corpus. Unsupervised learning of semantic orientation from a hundred-billion-word corpus, 2002.
- [62] P. D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2002.
- [63] P. D. Turney and M. L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21:315–346, 2003.
- [64] A. Voutilainen. A syntax-based part-of-speech analyser. In *EACL-95*, 1995.
- [65] T. Wilson, J. Wiebe, and R. Hwa. Just how mad are you? finding strong and weak opinion clauses. In *Proceedings of the 19th national conference on Artificial intelligence*, 2004.
- [66] H. Yang, L. Si, and J. Callan. Knowledge transfer and opinion detection in the TREC2006 blog track. In *Proceedings of TREC*, 2006.
- [67] J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *IEEE Intl. Conf. on Data Mining (ICDM)*, 2003.