ENHANCING AN INFORMATION EXTRACTION ENGINE TO SUPPORT SENTIMENT ANALYSIS OVER SOCIAL NETWORKS

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2012

By
Andres Martin Lopez
School of Computer Science

Contents

Word Count: 21,923

List of Tables

List of Figures

Abstract

ENHANCING AN INFORMATION EXTRACTION ENGINE TO SUPPORT SENTIMENT ANALYSIS OVER SOCIAL NETWORKS
Andres Martin Lopez
A dissertation submitted to the University of Manchester
for the degree of Master of Science, 2012

The aim of this dissertation is to enhance the Information Extraction Engine *Cafetiere* to conduct sentiment analysis over social network feeds from Twitter, by implementing a series of features that allow the automatic collection of *tweets* directly from the engine, as well as endowed it with more robust capabilities for capturing the sentiment contained within the collections retrieved.

Sentiment analysis over social networks is an area of research that has gained popularity during the recent years, due to potential applications that can be derived from understanding users' opinions.

The previous version of *Cafetiere* supported basic sentiment analysis functionality over a collection of Twitter feeds that were manually uploaded to the system. However, the results obtained with this approach were not very robust, as they simply relied on the presence or absence of subjective words matching entries from a dictionary.

The set of features developed for this dissertation make use of more advanced techniques, by exploiting sentiment lexicons and machine learning algorithms for evaluating the presence of sentiment.

The automatic collection of tweets was implemented using *Twitter Search API*, which allows real time searches of feeds, by using customized queries that are sent via HTTP requests against Twitter Servers. The data retrieved is then stored within the database currently integrated with the system.

Three different sentiment analysis approaches were developed for *Cafetiere*, in order to evaluate and compare their performance when applied within the Twitter context. The first approach relies on the use of *SentiWordNet*, a sentiment lexicon composed by *WordNet* synsets that have been automatically annotated with a sentiment score. Then, a second approach was implemented with the aim of enrich the results obtained from

SentiWordNet, when introducing *Word Sense Disambiguation Techniques* that consider word context for the final sentiment calculation.

Finally, *Cafetiere* was enhanced with supervised algorithms for sentiment analysis, by developing a two step classifier for subjectivity detection and polarity evaluation.

The performance obtained by each of these enhancements was evaluated with a collection of tweets that had been manually annotated with sentiment scores. The results from this evaluation are also presented, showing that supervised algorithms are more suitable for sentiment detection within the Twitter context, as well as revealed how sentiment lexicons approaches are constraint by the limited contextual information that can be extracted from the short number of words composing a Twitter post.

Finally, a series of recommendations are proposed to overcome with the deficiencies detected during the evaluation period, as well as a description of potential future works that could expand the enhancements here presented.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to express my sincere gratitude to my supervisor Mr. John McNaught, for all the support and guidance provided during this project. I would also like to thank Mr. William Black, for all the technical discussions that helped me to have an in depth-understanding of the subject.

Dedications

To my parents,

Chapter 1

Introduction

1.1 What is Sentiment Analysis?

As described by [1], *sentiment analysis*, sometimes also referred as *opinion mining* or *sentiment mining*, is an area of computational research that aims to identify the opinion, sentiment and subjectivity expressed by an author, or group of authors, within a piece of text or document. It mainly relies on the use of Natural Language Processing (NLP) techniques to automate the extraction and classification of sentiment, typically from unstructured text.

Early works conducted during the 1990s [2, 3] were mainly focused on identifying those parts of a document expressing points of view and beliefs, by analysing the presence or absence of certain linguistic elements like nouns, adjectives or phrases. More recent works deal with the problem of polarity classification, which consists on classifying opinionated documents depending on whether they express an overall positive or negative opinion [4, 5]. While some techniques make use of single classifiers for this purpose, others are based on more advanced implementations using a two step classifier that combines subjectivity detection with polarity evaluation [6, 7]. Overall, all these approaches rely on the property that opinion is commonly found in those parts of a text with subjective content [8].

Sentiment Analysis is an area of research that has gained popularity since the beginning of the 21st. century, when the mass adoption of the World Wide Web enabled a space where people could easily share their opinions and feelings, as well as created a useful source of data to conduct this type of research. [4, 9] are examples of works that have used the Web to detect users' opinions from review aggregation websites like

IMDb[1] or *Epinions*[2]. Furthermore, with the advent of the Web 2.0 and social networking services like *Twitter*[3], opinions and ideas from millions of users can now be shared in real-time, thus producing an enormous amount of data that can be also exploit for this type of analysis. The success of this new *social phenomenon* is one of the reasons behind the recent blossoming of papers on this field.

There are some additional facts that have also contributed in bringing popularity to the field and have been identified by [1] as:

- The rise of new machine learning methods for Natural Language Processing, text classification and information retrieval.
- Availability of datasets to train machine-learning algorithms, like review-aggregation websites or feeds from social networks.
- Potential commercial and scientific applications behind this area of research. Some of these applications are:
  - **Consumer Information:** sentiment analysis can be applied for product reviewing, detecting what consumers are thinking about a particular product or service, as well as measuring the consumer confidence over time [9, 10, 11].
  - **Marketing:** trends and consumer attitudes towards certain products, news or ideas can be extracted from social networks and web sites. This data could be used to predict future changes or impacts on the market [12]. Also, recommendation systems could be developed based on the knowledge of users' preferences.
  - **Politics:** governments could use sentiment analysis techniques to detect citizens' opinion about policies that are being applied. Also, political parties could detect what is the overall perception of the media and their voters towards their candidates [13, 14, 15].
  - **Social Sciences:** sentiment analysis over social networks can be used to conduct polls, in an easier and faster way than traditional polling methods [14, 15].

[1] *IMDb*, available at: http://www.imdb.com
[2] *Epinions*, available at: http://www.epinions.com
[3] *Twitter*, available at: http://www.twitter.com

– **Social Networks:** sentiment analysis and social networks can be combined to identify clusters of people and communities sharing the same opinions and ideas, thus linking people with the same mind-set.

1.2 *Cafetiere*: An Information Extraction Engine

Cafetiere [16] is a web-based *information extraction engine* that has been developed by *NaCTeM*, The National Centre for Text Mining at The University of Manchester, for text analysis over a collection of documents that are manually uploaded by users. It relies on a pipeline of text analytic processes that are executed following the *Apache UIMA*[4] framework for *entity recognition*, *term recognition* and also *basic sentiment analysis*.

The pipeline applied executes the following sequence of processes:

- **Tokenization and Part of Speech Tagging (PoS Tagging):** where each document is divided into elementary text units or *tokens*, such as words, numbers, punctuation and symbols. Then, each word is characterized with the part of speech it belongs to.
- **Dictionary Lookup:** each token matching an entry in a dictionary, that is also uploaded by the user, is annotated with meta-data containing the class of entity it represents.
- **Rule based analysis:** rules defined by the user are applied to identify and annotate entities, or other text elements, in a more general way than dictionaries do.

The result obtained after the analysis is a sequence of text elements annotated with the meta-information generated by each of the previous processes.

The current version of *Cafetiere* supports basic sentiment analysis functionality over a collection of *tweets*[5] that can be manually uploaded to the system. Each *tweet* is processed by the pipeline previously described, and its sentiment score is computed as follows:

[4] *Apache UIMA Project*, available at: http://uima.apache.org

[5] *tweet*, is a term commonly used for a Twitter message posts.

1. For each *tweet*, the system first looks for tokens matching subjective expressions from dictionary entries, then calculates its *sentiment score* based on the information provided by this dictionary.

2. Finally, the total *sentiment* is computed as the average score obtained for each *tweet* from the collection.

However, the results obtained with this technique are not very robust because they simply rely on the presence or absence of subjective words, without applying more advanced sentiment detection mechanisms that could take into consideration *context*, or additional *text features*, like those described in [4, 17].

1.3 Project Aim: Enhancing an Information Extraction Engine

The aim of this Project is to enhance the information extraction engine *Cafetiere* to improve its sentiment analysis capabilities over social network feeds from Twitter, by adding the following enhancements:

- **Enhancement 1:** *Cafetiere* will be enhanced to enable the collection of *tweets* directly from its Web client. Users will be able to type the set of keywords that will define the topic of the *tweets* to be requested. In addition, they will be able to select the date from/until the *tweets* have to be retrieved, their language, as well as the maximum number to collect.

  Then, the system will send queries to Twitter Servers requesting *tweets* matching the set of parameters previously provided.

  Finally, the collection of *tweets* retrieved will be stored within the user's database.

- **Enhancement 2:** The lexical resource *SentiWordNet* [18] will be integrated with *Cafetiere*, so the sentiment analysis will be computed using the information provided by this lexicon.

- **Enhancement 3:** *Word Sense Disambiguation* techniques will be combined with *SentiWordNet* to improve *Cafetiere's* sentiment analysis capabilities.

- **Enhancement 4:** *Cafetiere* will be enhanced with machine learning capabilities for text analysis, in order to build robust classifiers for subjective detection and

polarity evaluation. This will allow exploiting machine-learning algorithms that make use of text features, as well as Twitter syntax features, for the detection and evaluation of sentiment, similar to the approaches described in [17, 19].

Finally, the previous enhancements will be applied to a particular *Twitter topic* to analyse and evaluate its results with the improvements previously mentioned.

1.4 Dissertation Organization

This document is organized into five different chapters, where each of them represents the work conducted for this dissertation:

- Chapter 2 is dedicated to analyse the current state of the art in research on sentiment analysis, the different techniques that are commonly applied, its challenges, as well as to describe how the influence of social networks has helped to bring popularity to this field. In each case, detailed information from relevant literature is provided and its advantages and disadvantages are also discussed. In addition, there is a special section dedicated to describe the two *Twitter APIs* available for the collection of social network feeds.

- Chapter 3 focuses on the general methodologies and development tools that have been applied to conduct this dissertation, as well as the specific ones that were required for each enhancement implemented. In each case, a description of the requirements, the design choices and their implementation details is provided.

- Chapter 4 is dedicated to evaluate each of the enhancements developed, providing a detailed description of the different evaluation techniques applied, as well as comparing the performance obtained for each of the sentiment analysis approaches implemented.

- Chapter 5 summarizes this dissertation, analysing its current strengths and limitations, as well as proposing further developments that could be built based on the work here conducted.

Chapter 2

Background

2.1 Sentiment Analysis Approaches

Most of the work conducted on sentiment analysis is mainly focused on solving the problems of *subjectivity detection* and *polarity classification* over different types of corpora, which requires defining a classifier that will categorize the documents accordingly. The majority of these approaches are based on Natural Language Processing techniques combined with one of the following methods:

- *Machine learning algorithms.*
- *Lexical resources and sentiment lexicons.*

Examples of these different methodologies are covered in more detail from section 2.2 to section 2.6.

2.1.1 Polarity Classification

The problem of *polarity classification* consists on classifying documents as *positive* or *negative*, depending on their overall opinion towards the subject of matter. The work conducted by [4] represents an example where machine-learning techniques are applied to build a binary classifier that categorizes movie reviews from *IMDb*, as positive (*Thumbs up*) or negative (*Thumbs down*), based on the semantic orientation of the phrases contained within the review. Other works [5] propose a different approach, making use of information retrieval techniques to identify polarity on product reviewing.

2.1.2 Subjectivity Detection

Subjectivity detection involves identifying those parts of a document containing subjective information, usually as a previous step before polarity classification. Many works [6, 7, 20, 21] have shown that a two-step classifier, where firstly the subjective content is detected and then its sentiment polarity is evaluated, tends to provide better results.

This two-step classification approach relies on the property that opinion is commonly found in those parts of a text where subjective content is present, constituting a good source of data to determine the polarity of the document. This property was summarized by [8], based on the conclusions obtained from [7, 22, 23, 24], as follows:

"In fact, the problem of distinguishing subjective versus objective instances has often proved to be more difcult than subsequent polarity classication, so improvements in subjectivity classication promise to positively impact sentiment classication".

2.1.3 Other Sentiment Analysis Problems

Other works on sentiment analysis are focused on solving problems like: finding the target of the sentiment that a particular sentence is talking about (*aspect*) [10, 11], identifying different grades of sentiment using non-binary classifiers [25], or the detection of *emotions* [26], *humour* [27] and *mood* [28].

2.2 Natural Language Processing Techniques for Sentiment Analysis

A combination of different Natural Language Processing techniques is usually applied as a prior step to the approaches described in sections 2.3 and 2.6. These techniques are normally executed as a pipeline of processes, aiming to normalize and clean the data (text in this context) that will be used for further sentiment analysis purposes.

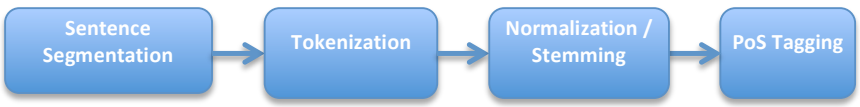

Figure 2.1: Example of an NLP pipeline of processes

Figure 2.1 shows an example of a sequence of processes that is normally executed. According to [29], each step involves:

- **Sentence segmentation:** where a stream of text is broken into sentences, based on the presence of elements like punctuations, white spaces or capital letters. It typically relies on the use of simple decision trees or more advanced machine learning techniques.

- **Tokenization:** in this step, each sentence is divided into elementary units or *tokens*, such as words, phrases punctuations, symbols or other meaningful elements.

- **Normalization:** where tokens are transformed into more consistent elements with the same textual form. This typically involves the use of rule-based text processing and dictionary-based text recognition for:

  - Expanding abbreviations and dates:

    $$\textit{13/11/1982} \rightarrow \textit{13 November 1982}$$
    $$\textit{Prof.} \rightarrow \textit{Professor}$$

  - Converting all letters to lower case.

  - Removing repeated characters:

    $$\textit{hellooooooo} \rightarrow \textit{hello}$$

  - *Lemmatization:* or reducing inflections or variant verb forms to their base form or *lemma*:

    $$\textit{am, are, is} \rightarrow \textit{be}$$
    $$\textit{book, books, books, books} \rightarrow \textit{book}$$

- **Stemming:** each individual term is reduced to its *stem*[1], i.e.: *"automate(s)", "automatic"* and *"automation"* are substituted by *"automat"*.

  A common simple and effective stemmer for English Language is the *Porter's algorithm* [31], which applies a set of replacing rules through 5 different phases.

  Both *stemming* and *lemmatization* aim to reduce inflectional forms of a word into a common base form, but they differ in the way this reduction is done. For instance, *stemming* involves removing the end of a word via a set of rules, while

[1] Morpheme is *"the lowest unit of language that can convey meaning"* [30]. There are two types of morphemes: *stems*, the core meaning-bearing unit of a word, and *affixes*, that are attached to a *stem* to form a new word.

lemmatization relies on the use of vocabularies and morphological analysis for this purpose.

Stemming is normally chosen as a faster an easier alternative to *lemmatization*, specially in those contexts where it is not require to generate words from a combination of a stem and an affix.

- **Part-of-Speech-Tagging (PoS Tagging):** in this step each word in the text is associated with its corresponding part of speech: noun, verb, adjective, adverb, etc. This association is based on the word definition and its context (the knowledge of its neighbouring words).

2.3 Lexical Resources and Sentiment Lexicons for Sentiment Analysis

Sentiment lexicons are commonly used in sentiment analysis to determine the orientation of the opinion contained in a text. These lexicons are constituted by a set of terms that have been annotated according to the notions of *positivity*, *negativity* and *neutrality* or *objectiveness*. This approach typically involves the following steps:

1. Firstly, each word in the document is processed to check if it is contained within the lexicon.
2. If a word is found, its meta-information about *positivity*, *negativity* and *neutrality* is used to calculate its polarity orientation.
3. Finally, the overall sentiment orientation of a piece of text, whether if it is *positive*, *negative* or *neutral*, is determined by the class with the highest frequency of each of these orientations.

Examples of works using this approach can be found at [32] and [11], where they use sentiment lexicons to determine the opinion orientation expressed by users on product feature reviews.

There are several sentiment lexicons available for research purposes, some of the most commonly used are:

- **The General Inquirer** [33].
- **MPQA** [7, 34].

- **SentiWordNet 3.0.**

2.3.1 The General Inquirer

As described in [33, 35], *The General Inquirer* is a content-analysis tool that maps part-of-speech tagged words with syntactic, semantic and pragmatic dictionary-supplied categories. Developed in 1966, is considered to be one of the first tools for content analysis. Its current version contains 182 categories, mainly from the *Harvard IV Psychological Dictionary*[2] and the *Lasswell Value Dictionary* [36]. Table 2.1 shows a fragment of this lexicon.

In spite of being developed for social-science content analysis research applications, it is also commonly used for text classification and other natural language processing analysis. Within the context of *sentiment analysis*, [37, 38] constitute examples of works that have used this lexicon to find correlations between sentiment values of words in the *Wall Street Journal*: "Abreast of the Market" column and the performance of the *Dow Jones Industrial Average*, demonstrating that the proportion of negative and positive words in a text are highly correlated with stock returns.

One of the advantages that this lexicon provides is that terms are categorized into *negative/positive*, *weak/strong* and *active/passive*, as well as other categories reflecting *pleasure, pain, motivation, cognitive orientation, etc.*, thus allowing additional sentiment classifications than just subjectivity or polarity detection. Also, users can create customized dictionaries containing domain specific terms.

	Entry	Positive	Negative	....Other Classes...	OthTags
1	A				DET ART
2	ABANDON		Negative		SUPV
3	ABANDONMENT		Negative		Noun
4	ABATE		Negative		SUPV
5	ABATEMENT				Noun
...					
174	ADJUST#1				SUPV
175	ADJUST#2	Positive			Modif
...					
11788	ZONE				Noun

Table 2.1: A fragment of the General Inquirer

[2] *The Harvard-IV Psychological Dictionary* is available at the *General Inquirer* website: http://www.wjh.harvard.edu/ inquirer/

2.3.2 MPQA

The *Multi-Perspective Question Answering* (MPQA) is an annotated corpus for opinions and other private states from several news articles. Created as part of the work conducted in [39], it consists of 6885 words from 8221 lemmas. Table 2.2 shows a fragment of this lexicon, where each entry contains the following properties:

- *type*: terms are labelled as *strongsubj/weaksubj*, depending on whether they are considered to be *strongly/weakly* subjective in the majority of the contexts where they are used.
- *len*: represents the number of words that form the term. All the terms in the lexicon are single words.
- *word1:* the token or stem of the term.
- *pos1:* the part of speech of the term. Possible values are *noun, adjective, verb, adverb* or *anypos* (any part of speech).
- *stemmed1:* if is set to *yes*, the entry should then match all unstemmed variants of the word with the corresponding part of speech.
- *priorpolarity:* this property represents the prior polarity of the term when is taken out of context. Possible values are: *positive, negative, both* or *neutral*. The lexicon contains 2718 words with a *positive* polarity and 4912 words with a *negative* polarity.

	Type	Length	Word	POS	Stemmed	PriorPolarity
1	weaksubj	1	abandoned	adj	n	negative
2	weaksubj	1	abandonment	noun	n	negative
3	weaksubj	1	abandon	verb	y	negative
4	strongsubj	1	abasement	anypos	y	negative
5	strongsubj	1	abash	verb	y	negative
...						
8211	strongsubj	1	Zest	noun	n	positive

Table 2.2: A fragment of the MPQA lexicon

2.3.3 SentiWordNet

This lexicon is an extension of *WordNet* [40] *synsets*[3] that have been annotated with scores representing *positivity*, *negativity* or *neutrality*.

WordNet 3.0

WordNet 3.0[4] is a lexicalized ontology where nouns, verbs, adjectives and adverbs are grouped into *synsets*, each expressing a distinct concept with unambiguous meaning, i.e., the concept *car* has 5 different *senses*[5] associated with the following *synsets*:

- car#1: "car, auto, auto-mobile, machine, motorcar".
- car#2: "car, rail-car, railway car, rail-road car".
- car#3: "car, gondola".
- car#4: "car, elevator car ".
- car#5: "cable car, car".

In addition to synonym relations between words, *WordNet* also provides the following information:

- *Gloss:* a synset definition, e.g., *car#1:"a motor vehicle with four wheels; usually propelled by an internal combustion engine"*.
- *Examples of use:* one or more sentences illustrating the use of the synset members, e.g., *car#1:"he needs a car to get to work"*.
- *Relations:* interlinked relations between synsets, by meanings of conceptual-semantic and lexical relations:
  - **Hyperonymy-Hyponymy:** an *is-a-relation* or *kind-of-relations*, e.g., car#1 *is-a-kind-of* motor-vehicle#1 and automotive-vehicle#1.
  - **Meronymy-Holonymy:** a *part-of-relations*, e.g., car#2 *is-part-of* elevator#1 and lift#8.

[3] *WordNet* defines a *synset* as: a set of one or more synonyms that are interchangeable in some context, without modifying the truth value of the proposition in which they are embedded

[4] *WordNet 3.0* is available at http://wordnet.princeton.edu/

[5] A *word sense* is a commonly accepted meaning of a word [41].

- **Troponyms:** between verbs, representing a *particular-way-of* relations, e.g., play#1 *is-a-particular-way-of* gamble#2.
- **Antonymy:** between adjectives, e.g., perfect#1 *is-an-antonym-of* imperfect#2.
- **Pertainyms**: between adjectives and pointing to those nouns they derived from, e.g., criminal#2 *derives-from* crime#2.
- **Nominalization:** representing nouns nominalizing verbs, e.g, service#2 *nominalizes-the-verb* serve#4.
- **Entailment:** a verb Y is entailed by a verb X, if by doing X it must also be doing Y, e.g, snore#1 *entails* sleep#1.
- **Similarity:** between adjectives, e.g., beautiful#1 *is-similar-to* pretty#1.

SentiWordNet 3.0

SentiWordNet 3.0 is a lexical resource that was created to support research on opinion mining and sentiment classification, and is the result of an automatic annotation of *WordNet synsets*, by using a combination of propagation methods with linguistic and statistical classifiers. For this reason, it should not be considered as a gold standard resource as it is *WordNet*, which has been compiled by humans. However, it has been successfully used for different sentiment analysis tasks.

In *SentiWordNet*, each synset is associated with three numerical scores:

$$\{Pos(s), Neg(s), Obj(s)\}$$

representing how *positive*, *negative* and *objective* are the terms belonging to that synset. Each of these scores are ranged within the interval $[0,1]$, and their total sum per synset is always 1, i.e., the sentiment scores for the synset *happy#1* are:

$$\{0.875, 0, 0.125\} \quad ,$$

where $Pos(happy\#1)$ =0.875, $Neg(happy\#1)$ =0, $Obj(happy\#1)$ =0.125.

Terms with multiple synsets can also have multiple sentiment scores associated. Table 2.3 shows an example of the different sense scores for the adjectives *happy* and *unable*.

By the time the latest version of *SentiWordNet* was released in 2008 [18], more than 300 research groups held a license of this lexical resource, and its effectiveness for sentiment analysis has been proved in a variety of research projects. Examples can be found in [42], which states that although machine-learning approaches tend to provide

POS	ID	PosScore	NegScore	SynsetTerms	Gloss
a	01148283	0.875	0	happy#1	enjoying or showing or marked by joy or pleasure
a	01048406	0.75	0	happy#2, felicitous#2	marked by good fortune
a	02565583	0.5	0	happy#3, glad#2	eagerly disposed to act or to be of service
a	01000442	0.125	0	happy#4, well-chosen#1	well expressed and to the point
a	00002098	0	0.75	unable#1	not having the necessary means or skill or know-how
a	00307794	0	0.375	unable#2	lacking necessary physical or mental ability
a	01825080	0.125	0.25	unable#3, ineffectual#3, ineffective#2	lacking in power or forcefulness

Table 2.3: A fragment of SentiWordNet

better results, the use of *SentiWordNet* is *"accurate enough"* for this purpose. Its potential has been also proved for sentiment analysis over short documents like *"message posts"* [43].

However, its use has some shortcomings. According to [11], this approach misses the information from *"context dependent opinion words"*. In addition, the results obtained can be biased, as they tend to rely too much on the automatic sentiment scores from *SentiWordNet*, without considering domain specific lexicons [43].

2.3.4 Disagreement between Sentiment Lexicons

The above lexicons are widely used in the research domain for solving the problem of polarity classification, but a certain level of disagreement has been found in the vocabulary they contain [44]. Table 2.4 illustrates these different levels of disagreement, which are related to the general problem of *subjectivity* and how domain effects, cultural and individual differences can affect the perception of a word polarity within a particular context.

In general, if a word is present in multiple lexicons, their disagreement tends to be very low, thus providing a similar behaviour for polarity evaluation purposes. However, an exception is found with *SentiWordNet*, which seems to have a higher level of disagreement with the other two lexicons. This difference is related with the way it was constructed, as it consists on automatic annotation of *WordNet* synsets, while the other two have been manually annotated by human subjects. Due to these structural differences, it is then difficult to establish a comparison between *SentiWordNet* and the other two lexical resources, apart from the difference in the performance obtained

when they are applied for a particular sentiment analysis task.

Table 2.4: Disagreement between sentiment lexicons.

Source	MPQA	Inquirer	SentiWordNet
MPQA	-	49/2867(2%)	1127/4214(27%)
Inquirer		-	520/2306(23%)

2.4 The Problem of Word Sense Disambiguation

When using lexicons like *SentiWordNet* for sentiment analysis, where terms can be associated with multiple synsets containing different sentiment scores, it is then necessary to identify their correct sense within the context where they are used. This process is known as *Word Sense Disambiguation* (WSD), which is complex enough to have its own area of research.

Many works using *SentiWordNet* for their analysis [42, 43] tend to avoid the WSD step, by calculating the sentiment score of a piece of text or document as follows:

1. Firstly, the *PoS* of each word is obtained.
2. Then, if a word exists within the lexicon, its sentiment score is calculated as the average score from all the synsets associated with the word and its *PoS*.
3. Finally, the sentiment score of the whole piece of text or document is averaged between the scores obtained for all its words.

However, better results can be achieved if a WSD technique is implemented before a lexicon lookup [41], as the precise sentiment score can be obtained from the senses disambiguated. The following section describes the approaches commonly applied for this purpose.

2.5 Word Sense Disambiguation Approaches

WSD is one of the traditional research fields within the area of Natural Language Processing, and involves the analysis of text and its context to associate words with senses. This process typically involves the following steps [41]:

1. Firstly, several NLP techniques are applied to transform the intrinsic unstructured nature of text into a more structured format, allowing further automatic processing to conduct the disambiguation step. The list of NLP methods described in section 2.2 are typically applied for this purpose.

2. Depending on the WSD approach chosen, an external *knowledge resource* may be used. These type of resources contain the information required for the association between senses and words.

3. Finally, a *classification method* will be applied to solve the *disambiguation* problem.

2.5.1 Knowledge Resources

There are several *knowledge resources* that are widely used by the research community to perform the task of *disambiguation*. Some of the most common ones are [41]:

- **Structured resources:**
  - *Thesauri:* a lexical resource where words are grouped according to their similarity of meaning, mainly based on *synonym* and *antonym* relationships. A commonly used thesaurus for disambiguation is the *Roget's International Thesaurus* [45].
  - *Machine readable dictionaries:* like *Collins English Dictionary* [46] or the *Oxford Dictionary of English* [47].
  - *Ontologies: "which are specifications of conceptualizations of specific domains of interest"* [48]. *WordNet* and its extensions like *SentiWordNet* are examples of widely used lexical ontologies.
- **Unstructured resources:**
  - *Corpora:* collections of texts that have been created for language modelling and language analysis tasks. There are two different types of *Corpora*:
    - *Raw Corpora*, like the *British National Corpus* [49] or the *Wall Street Journal Corpus* [50].
    - *Sense-annotated Corpora*, like *SemCor* [51] or *Open Mind Word Expert data set* [52].

- *Collocation resources:* these lexical resources contain information about the tendency of word co-occurrence within a particular context. The *The British National Corpus Collocations*[6] and the *Web1T corpus* [53] are typically used *collocation resources*.
- *Other resources:* like word frequency lists, *stoplists* or *domain labels* [54].

2.5.2 Word Sense Disambiguation Classification Methods

There are three main classification approaches for WSD [41]: *Supervised*, *Unsupervised* and *Knowledge-Based* disambiguation approaches.

Supervised Disambiguation

These techniques rely on machine-learning classifiers and manually sense-annotated training data sets for solving the *disambiguation* problem. Popular methods used for this purpose are: *Decision Lists*, *Decision Trees*, *Naive Bayes*, *Neural Networks* or *Suport Vector Machines*.
According to [55], supervised methods tend to provide better results. However, they are out of the scope of this dissertation due to the implementation choice described in section 3.2. For this reason, no more detailed description is provided within the following sections. More information about these approaches can be found in [55].

Unsupervised Disambiguation

Unsupervised disambiguation approaches make use of unlabelled corpora and are based on the idea that *"the same sense of a word will have similar neighbouring words"* [41]. Their goal is to identify sense clusters and infer word senses based on clustering word occurrences, rather than do a direct association between senses and words.

As a result, they are unsuitable for the purpose of this dissertation, where a fined-grained association between senses and words is required for obtaining an accurate *sentiment analysis*. For this reason, no more detailed description about these techniques is provided in this section.

Typical examples of *unsupervised disambiguation* techniques are: *context clustering*, *word clustering* and *co-occurrence graphs*.

[6]The *British National Corpus Collocations*, available at: http://www.natcorp.ox.ac.uk.

Knowledge-Based Disambiguation

Knowledge-based disambiguation methods rely on the contextual information provided by external *knowledge resources*, mainly *WordNet*, to associate words with senses.

According to [41], although they tend to have a lower performance in comparison with the *supervised* approaches, they are commonly used for research purposes because of the wide coverage provided by the external resources involved. Commonly used *knowledge-based* methods are:

- **Overlap of Sense Definitions**, also known as the *Lesk algorithm* [56], is a technique that disambiguates terms by calculating the word overlap between *WordNet glosses* (sense definitions) of two or more synsets within a given context.

  According to [41], this method provides a very variable accuracy (between 50-70%), as it is very dependent on the short number of words contained within the glosses. [57] proposed an improved version of the *Lesk algorithm*, by expanding the synset gloss overlapping with glosses of other concepts related via *WordNet* relationships. This new measure, known as the *extended gloss overlap*, is calculated as follows [41]:

  $$score_{ExtLesk}(S) = \Sigma_{S':S \xrightarrow{rel} S' or S \equiv S'} |context(w) \cap gloss(S')|,$$

  where *context(w)* represents the list of words in the context around the word *w* and the *gloss(S)*, represents the list of words in the *gloss* of the current sense *S*, or any other sense *S'* related to *S* through a relation *rel*.

  The work conducted by [58] demonstrates that this improved version of the *Lesk algorithm*, performs better than any of the *WordNet Similarity Measures* approaches also described below.

- **Selectional Preferences:** are based on *"constraints on the semantic type that a word sense imposes on the words with which it combines in sentences, usually through grammatical relationships"* [41]. These approaches do not perform as well as the other methods here described, and they have been discarded for the purpose of this dissertation.

- **Structural Approaches:** these methods solve the disambiguation problem by exploring the semantic network of concepts from lexicons like *WordNet*. There

are two main approaches of this kind [41] : *Similarity Measures* and *Graph-Based Approaches*, which are covered in more detail below, as they are the WSD techniques chosen to improve *Cafetiere* capabilities. Section 3.2 outlines the reasons behind this choice.

Structural Approaches: Similarity Measures

This WSD approach is based on measures of semantic similarity over *WordNet* relations. Given a metric $rel(c_1, c_2)$ between two concepts, c_1 and c_2, the relatedness $rel(w_1, w_2)$ between two words, w_1 and w_2, can be calculated as [59]:

$$rel(w_1, w_2) = \max_{c_1 \in s(w_1), c_2 \in s(w_2)} [rel(c_1, c_2)],$$

where $s(w_i)$ represents the whole list of senses of the word w_i within the lexicon.

There are several approaches of word sense disambiguation using *similarity measures*, and the difference between them relies on the metrics chosen to calculate the semantic relatedness between concepts. Commonly used metrics are:

- *Path Lenght* [60]: where words are disambiguated based on the length of the shortest path between word senses in the *WordNet taxnomy* (measured in edges or nodes):

$$rel(c_1, c_2) = len(c_1, c_2)$$

- *Leacock and Chodorow* [61]: this similarity measure is similar to *Path Lenght*, but the distance between two concepts is scaled by the maximum depth of their overall *WordNet* hypernymy taxonomy:

$$rel(c_1, c_2) = -log \frac{len(c_1, c_2)}{2x \max\limits_{c \in WordNet} depth(c)}$$

- *Information Content*: similarity between concepts can be measured by the *"extent to which they share information in common"* [62]. In a lexicon like *WordNet*, which contains an hyperonymy taxonomy, the information content can be determined by *"inspecting the relative position of the most-specific concept that subsumes the targeted words"* [62]. According to this principle, the more specific is the concept that subsumes two or more words, the more semantically related they are. Figure 2.2 illustrates this concept.

  The metric is calculated as follows:

$$rel(c_1, c_2) = -log[p(lso(c_1, c_2)],$$

where $p(c)$ represents the probability of an instance of the concept c within the text or referenced corpus and $-logp(c)$ is the *information content* of c ($IC(c)$).

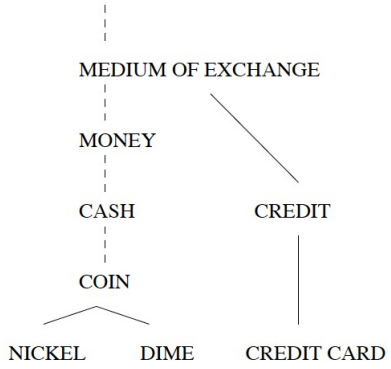

Figure 2.2: Fragment of the *WordNet* taxonomy. Solid lines represent *is-a* links; dashed lines indicate that some intervening nodes were omitted to save space. From [62], Figure 1.

- *Jiang & Conrath* [63]: this approach relies on the *Information Content* concept previously described, but considering the conditional probability of finding an instance of a child sense, given an instance of an ancestor sense:

$$rel(c_1, c_2) = 2log[p(lso(c_1, c_2))] - (log(p(c_1)) + log(p(c_2))).$$

As it has been demonstrated by [58], *Jiang-Conrath* metric outperforms all the other *similarity measures* here mentioned for solving the problem of word-sense disambiguation.

Structural Approaches: Graph-Based Approaches

Graph-Based approaches rely on the notion of *lexical chains*, or *"sequences of semantically related words by a lexico-semantic relation i.e.: is-a, has-part, etc."* [41], as well as graph structures to determine the most appropriate sense of a word within a given context. Some examples of commonly used *Graph-Based* methods are:

- *PageRank and Semantic Networks* [64], this technique solves the disambiguation problem as follows:

1. Firstly, creating a graph that represents all the words from the text that are interconnected via meaningful relations from *WordNet*.
2. Then, a *PageRank* [65] algorithm is applied, obtaining for each word in the graph a ranked list with its associated senses.
3. Finally, the disambiguated sense chosen, will be the highest ranked sense from the previous list generated.

- *SSI Algorithm* [66]. The *Structural Semantic Interconnections* (SSI) algorithm disambiguates words by creating a *lexical knowledge-base* from semantic relations explicit encoded in *WordNet*, from other semantic relations taken from annotated corpora like *SemCor* and *LDC-DSO* [67], as well as from dictionaries of collocations. Then, the algorithm executes the following set of steps:

  1. Given a word within a context, SSI first builds a labelled directed graph representing its senses and the intermediate concepts that connect them with other senses (*semantic graph*). Figure 2.3 shows an example of the graph representation for the senses: *bus#1* and *bus#2*.
  2. Then, the algorithm uses a Context-Free grammar to select those senses maximizing their degrees of connectivity between the generated graphs.

  According to [66], and considering the intrinsic difficult nature of WSD problems, the SSI algorithm outperformed the state-of-the art of unsupervised WSD methods in the *Senseval-3-all-words* competition[7], with an overall precision and recall of 60.4%.

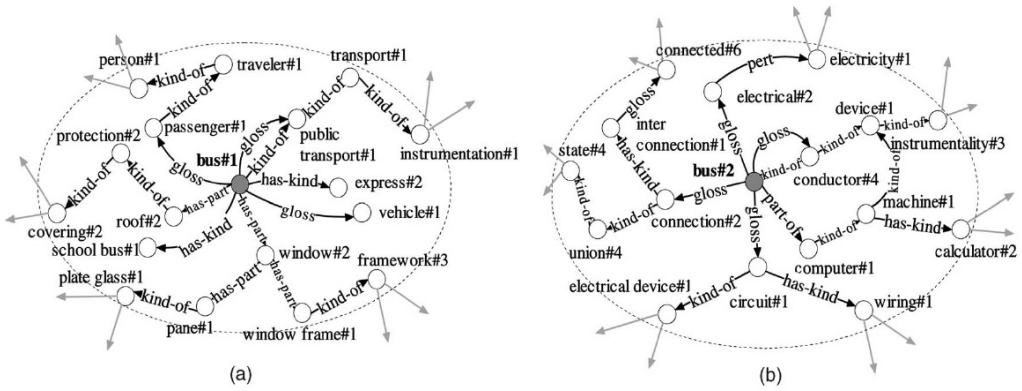

Figure 2.3: Graph representations for (a) sense #1 and (b) sense #2 of the word *bus*. From [66], Figure 1.

[7] *Senseval-3* test consisted of approximately 5,000 words of running texts from two *Wall Street Journal* articles and an excerpt of the *Brown Corpus*.

Recently, [68] has made available a *Java* API that provides access to an implementation of the SSI algorithm, which can also be enriched with two new graphed-based knowledge resources: *BabelNet* [69] and *WordNet++* [70] .

BabelNet is a very large multilingual extension of *WordNet*, where sense relations are expanded with an automatic mapping between word senses and *Wikipedia*[8] pages. *WordNet++* constitutes the English subset of *BabelNet*.

Their structure consists of a labelled directed graph, where concepts and named entities form the graph nodes and the semantic relations are represented by edges. Figure 2.4 shows an example of a *BabelNet* entry for the word *balloon*.

Overall, the results obtained in the work conducted by [68], when using this API for different WSD tasks, demonstrated the robustness of this approach, as well as the good performance provided by the SSI algorithm when compared with the current state-of-the-art.

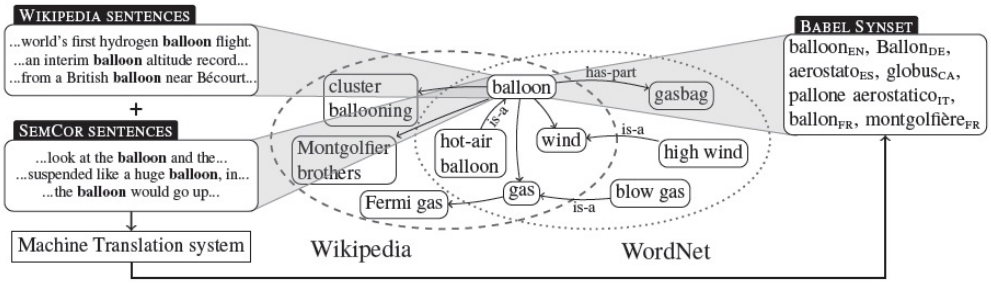

Figure 2.4: An example of BabelNet. From [69], Figure 1.

2.6 Sentiment Analysis Using Machine Learning Techniques

When sentiment analysis is applied using supervised learning algorithms, to either solve the problem of *subjectivity* or *polarity detection* of a piece of text or document, it is then required to build a classifier that can categorize those instances accordingly.

The classifier will take as an input instances of text represented by a set of features, thus identifying the subset of relevant features, as well as choosing the algorithm to be applied, are crucial steps when building robust classifiers for sentiment analysis.

[8] *Wikipedia*, available at: http://www.wikipedia.org/.

2.6.1 Formalizing the Text Classification Problem

The text classification problem can be formalize as follows [71]:

Given a document $d \in D$, represented as a vector of features $(x_1, x_2, ..., x_3)$, the goal is to classify d into a fixed set of *classes* $C = \{c_1, c_2, ..., c_n\}$, by using a *classification function* or *learning algorithm* γ:

$$\gamma : D \rightarrow C$$

This type of classification is known as *supervised classification*, as the function γ is built on labelled *training data* set, containing typical examples of each document class.

Finally, once the function γ has been learned, a test data set is used to check the performance of the classifier produced.

2.6.2 Text Feature Selection for Supervised Sentiment Analysis

Based on recent works that applied machine learning techniques for sentiment analysis purposes, [1] summarizes the following set of typically used text-features:

- **Term Presence vs. Term Frequency:** As described by [71], *term frequency* has been commonly chosen in traditional *Information Retrieval* and *NLP* approaches, specially for text categorization [72], where a high frequency of certain domain specific terms provides a high correlation with the topic it belongs.

  However, for polarity classification [4] demonstrated that *term presence* tends to provide better performance than *term frequency*. This result manifests the differences between traditional text topic categorization and sentiment classification.

  Sometimes a mixed approach combining linguistic an statistical information is also used. This is the case of *TerMine*[73], a text tool that allows the discovering of key terms and concepts mentioned in a document, based on their frequency and syntax.

- **N-grams:**[9] A typical approach for text classification is to combine n adjacent tokens to create an *n-gram*. An *n-gram* carries more contextual information that becomes useful for text analysis.

[9] An *n-gram* is a sequence of n text items.

However, it is important to remark that many of the *n-grams* formed could have no linguistic meaning. For this reason, some approaches include a PoS-tagging pre-step to remove undesirable occurrences of grams, i.e.:

$$university\ of\ manchester \rightarrow university\_manchester$$

Choosing a particular order of *n-grams* can also modify the performance of the classifier, but the results obtained seem to vary depending on the domain and how the rest of features are selected. For instance, the work conducted by [5] showed that for product-reviewing polarity classification, the use of *bi-grams* and *tri-grams* produced better performance than higher-order *n-grams*. However, [4] obtained better results with *uni-grams* for movie-review polarity classification.

- **Part Of Speech Tagging (PoS Tagging):** As it has been explained in section 2.2, is the process of associating a word in a text with its corresponding part of speech.

  PoS tagging is commonly used as a feature for supervised sentiment analysis and, in some scenarios, its identification becomes a crucial step. This was the case of the works conducted by [9, 74], which showed that adjectives are strong indicators of subjectivity and their polarity has a high impact on the overall sentiment classification [8]. However, [4] demonstrated that adjectives are not the only indicators of sentiment, so considering the presence of all words tends to produce better results.

- **Syntax Features:** Some researchers have also explored the used of syntax features for sentiment analysis purposes, which seems to be more relevant for short pieces of text [1]. In fact, [17] combined *text meta-features* with Twitter *syntax features* to build a robust sentiment classifier for Twitter feeds .

- **Negation:** the presence of negation in a sentence has an important contextual impact. For example, the following sentences: *"I like"* and *"I don't like"*, have an opposite sentiment polarity due to the presence of *"don't"*. Negation then needs special attention for correct sentiment classification.

  A common approach for these type of situations is the use of the technique introduced by [75], which consists on pre-pending *"NOT_"* to every word between a negation word and the following punctuation. So, phrases like:

  I don't like watching sports on TV, and also

  become:

I don't NOT_like NOT_watching NOT_sports NOT_on NOT_TV and also

Thanks to this strategy, each word between the negation word and the punctuation now represents a negative sentiment.

However, sometimes the presence of a negative word does not necessary imply a negative sentiment. For example, the sentence:

I wonder why no one said anything,

represents an objective sentence where the word *"no"* does not carry a negative polarity. So, special consideration needs to be taken for this type of situations.

2.6.3 Supervised Classification Algorithms for Sentiment Analysis

As it has been described in the previous sections, the performance of a classifier depends on how robust is the feature vector selected and the algorithm chosen. Commonly used supervised algorithms for text classification are [71]:

- **Naive Bayes.**
- **Support Vector Machines (SVM).**
- **Maximum Entropy (MaxEnt).**

The performance of these algorithms when using different sets of features, has been compared in the work conducted by [4], showing that machine-learning approaches tend to produce better results than human generated baselines. They also claimed that the best performance was achieved when using *uni-grams* with a presence-based frequency model and an *SVM* classifier (82.9% accuracy), followed by *MaxEnt* and *Naive Bayes*, although the differences between them are not very high. Examples of other works using this type of classifiers can be found in [5, 17].

The rest of this section provides a brief description of the list of *supervised-classification* methods previously mentioned.

Naive Bayes

A *Naive Bayes classifier* is a simple probabilistic classifier that is based on the *Baye's Theorem*, which defines the relation between conditional probabilities as follows:

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

In the context of text classification, the *Naive Bayes* classifier applies the following principle [76]: Given a document d, the class c_d in which the document will be classified, will be the class maximizing the following expression:

$$c_d = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} = \arg\max_{c \in C} P(d|c)P(c)$$

Documents are represented as a vector of features $(x_1, x_2, ..., x_n)$, so the previous expression can be re-written as:

$$c_d = \arg\max_{c \in C} P(x_1, x_2, ..., x_n|c)P(c)$$

Finally, this approach relies on the *naive* assumption that the set of features are conditionally independent given the class, thus the previous expression can be simplified as:

$$c_d = \arg\max_{c \in C} P(x_1|c) \cdot P(x_2|c) \cdot ... \cdot P(x_n|c)P(c) = \arg\max_{c_j \in C} P(c_j) \prod_{x_i \in X} P(x_i|c_j)$$

$P(c_j)$ is calculated as the relative occurrence of the frequencies of the class c_j within the training data set, while $P(x_i|c_j)$ is the relative occurrence of frequencies of the class c_j and the feature x_i, in the presence of the class c_j, within the training data set.

Zero counts of frequencies need to be taking into consideration, so smoothing techniques like the *Laplace smoothing* may be applied [71] to reduce their impact.

In spite of the *naive* assumption that features are considered independent given a class, the *Naive Bayes* classifier provides a high degree of accuracy for text classification purposes, with a performance that is comparable with other supervised methods [4, 17, 71, 72].

Support Vector Machines (SVM)

SVM is a widely-used supervised classifier that has been successfully applied for text classification problems [4, 17, 77].

This is a vector space-based supervised learning method, that aims to define decision boundaries between classes. The boundaries are located in a way that maximizes their distance to any point in the training data set [71].

The small subset of points that is used to define the decision boundaries are called *support vectors*, and the distance from the boundaries to the closet data point determines the *margin of the classifier*. Figure 2.5 illustrates these geometrical concepts in

a two dimensional space. As a result, the higher is the margin obtained, the lower is the classification error produced.

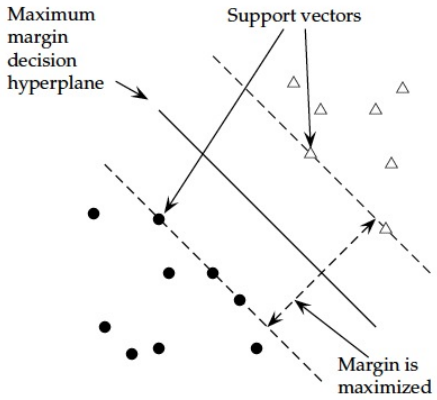

Figure 2.5: The support vectors are the 5 points right up against the margin of the classifier. From [71], Figure 15.1.

A generalization of the linear binary SVM classifier, $\gamma(x)$, classifying instances of $x \in X$ described by a vector of features (x_1, x_2), into *classes* $C = \{positive, negative\}$, can be expressed as follows:

$$\gamma(x) = w \cdot x + b,$$

where w represents a vector that is perpendicular to the boundary and b determines the offset of the boundary from the origin. Based on this formalization, instances of X will be classified as *positive* if $\gamma(x) \geq 0$ or *negative* if $\gamma(x) \leq 0$.

Support Vector Machines is not restricted to use only linear boundaries, it can also include extra non linear terms in the function to allow the definition of multi-dimensional decision boundaries (*quadratic, cubic, etc.*).

Maximum Entropy (MaxEnt)

Maximum Entropy classifiers are commonly used for text classification as an alternative to *Naive Bayes* and *SVM*. Their main principle is to learn probability distributions subjected to the constrains observed from the training data set [76].

The approach is based on the concept of *entropy*, which is a way of measuring the uncertainty of a distribution. Given an event x, with a certain probability p_x, its *entropy* (H) is defined by [76]:

$$H(p) = E_p[log_2 \frac{1}{p_x}] = -\sum_x p_x \log_2 p_x,$$

The goal is then to select the class that maximizes the entropy, subjected to the constraints added by the vector of features used to represent this event.

In the context of text classification, where a document $d \in D$ is classified into a class $c \in C$, the *maximum entropy* classification assumes that the set of features from the training data should be also present in the learned distribution, building a model that maximizes the conditional entropy of $p(c|d)$ as follows:

$$H(p) = -\sum_{d,c} \check{p}(d)p(c|d)\log p(c|d),$$

where $\check{p}(d)$ represents the document distribution extracted from the training data set.

Maximizing the entropy implies making the distribution as uniform as possible, so the model does not need to make additional assumptions about the data available.

Examples of works that have used *Maximum Entropy* classifiers for sentiment analysis purposes can be found at [4, 17].

2.7 Twitter and The Influence of Social Networks

Twitter is an on-line micro-blogging service where users can post messages in real-time (*tweets*) containing up to 140 characters. Although the number of characters might seem small, the service has gained enormous popularity since it was launched in July 2006, having now *"140 million active users posting an average of 340 million* tweets *per day"* [78].

One of the key features that Twitter offers is the use of *hashtags*, which are words or phrases prefixed by the symbol *"#"*. *Hashtags* act like meta-data storing the topic of the *tweet*. This meta-data is automatically incorporated to the post, allowing users to become actively creators of content that can be easily followed by other users. *Hashtags* also enable the detection of *trending topics*[10] . In fact, and as it has been demonstrated by [79], the majority of these trending topics are *"headline news or persistent news in nature"*.

The power and influence that social networking services like Twitter have in our society has been seen in recent political protests, like *Occupy Wall Street* [80] in New York or the *Arab Spring Revolution* around the Arab World during 2010 and 2011. Some academic studies have also demonstrated its influence; [13] analysed how Twitter activity evolved during a presidential debate between *Obama* and *McCain* in 2008;

[10] *Trending topics* are keywords within users' *tweets* that have a high increase in frequency.

[15] showed that the number of *tweets* containing political content *"reflects voters preferences and comes close to traditional election polls"*. Even there has been research on analysing the relation between Twitter and the stock market [12]. This variety of works simply reflects the potential that Twitter data offers for research on multiple areas.

2.8 Sentiment Analysis and Twitter

Twitter has also been exploit for sentiment analysis. In fact, the number of research papers on this area has increased considerably during the last two years. Examples of this type of works can be found in [14], where they found a high correlation between the sentiment expressed by Twitter users and the results from *Gallup Polls*[11] . Also, [12] showed that some sentiment attributes extracted from Twitter can predict the *Down Jones Index* with 3 days in advanced.

The majority of the research conducted on sentiment analysis over Twitter is based on subjectivity detection and polarity classification, by using supervised machine learning methods [17, 19] and lexicons [14].

However, as it has been demonstrated by [17], using classifiers with the same set of features as for sentiment analysis over large texts [4, 6] is not a valid approach for Twitter feeds, where the restriction of 140 characters constraints the contextual information that can be obtained from *tweets*. Based on this limitation, they proposed a more robust classifier by combining text-meta features (PoS Tags) with specific Twitter syntax features.

Other works have explored different set of features to identify sentiment and build classifiers, using punctuations, frequency of words, *n-grams*, *hashtags* or *emoticons*[12] [19].

2.9 Challenges Behind Twitter Sentiment Analysis

The automatic identification of sentiment is not considered to be an easy task, as it requires a good understanding of the language and the context where it is present. As a result, overcoming with computer's lack of intuition to process contextual information, is the main challenge involved with this area of research.

[11] *Gallup*, available at: http://www.gallup.com

[12] *Emoticons*, are textual expression representing facial expressions. i.e. ":-)" , ";-)" , ":-P"

However, the work conducted by [4] demonstrated that identifying sentiment in text is also difficult for humans, showing how machine-learning approaches can achieved significantly better results than human subjects. Their work also revealed that there is a certain level of disagreement between humans, when they were asked to identify a set of key words to determine the sentiment orientation of a piece of text. Figure 2.6 reflects the levels of accuracy they obtained for the same sentiment analysis task, when using the list of keywords chosen by human subjects and when applying a statistics-based approach.

	Proposed word lists	Accuracy (%)	Ties (%)
Human 1	positive: *dazzling, brilliant, phenomenal, excellent, fantastic* negative: *suck, terrible, awful, unwatchable, hideous*	58	75
Human 2	positive: *gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting* negative: *bad, cliched, sucks, boring, stupid, slow*	64	39
Statistics-based	positive: *love, wonderful, best, great, superb, still, beautiful* negative: *bad, worst, stupid, waste, boring, ?, !*	69	16

Figure 2.6: Sentiment classification using a list of keyword created by human subjects and when applying statistics over the test data set. Adapted from [4], Figure 1 and 2.

These results remark the fact that sentiment is sometimes difficult to identify, as it can be expressed in a very subtle manner, i.e., the following statement:

"you are telling me to join this cause, but where are the real evidences? everything seems too good to be true, something else must be hidden"

reflects a negative opinion that is hidden by the absence of explicit negative words. In addition, sometimes it is also difficult to differentiate between facts and opinions, i.e., in a review saying:

"A canon camera is better than the Fisher Price one",

can refer the objective fact that this type of camera is better than a *Fisher Price* one, or can reflect a negative opinion saying that its quality can be a bit better than a toy camera.

According to [1], sentiment and subjectivity are *"quite context-sensitive, and, at a coarser granularity, quite domain dependent"*, for example expressions like *"go read the book"* tends to indicate a positive sentiment in book reviews, but it is more likely to express a negative sentiment in a movie review context .

When sentiment analysis is brought into the Web and Social Networking context, the previous challenges are reinforced by the presence of abbreviations and slang words, i.e., expressions like *"lol"*, meaning *"laughing out loud"* or sometimes *"lots of laughs"*. This is specially remarkable in services like Twitter, where the limitation of a 140 characters per *tweet* encourage users to become more creative when writing their opinion and ideas, thus increasing the presence of this type of abbreviations, as well as *sarcasm* and *irony*.

Sarcasm is *"the use of remarks which clearly mean the opposite of what they say"* [30] and its presence changes the polarity of a comment to its opposite. As a result, sarcastic and ironic comments, together with the short length of *tweets*, represent the main challenges for sentiment analysis over Twitter. Their effects have been analysed in more detailed by [81, 82].

2.10 Collecting *Tweets* from Twitter

Any work conducting sentiment analysis over Twitter will require to implement a way to collect and store the *tweets* that will be analysed further on. For this purpose, Twitter provides the following two APIs [83] to access and interact with their services:

- **Search API.**
- **Streaming API.**

The differences between each of these API are explained in the following sections.

2.10.1 Search API

This API allows real-time searches of *tweets* via *RESTful* [84] methods, where queries are composed and sent using simple *HTTP* invocations of the following resource URL:

http://search.twitter.com/search.json

Table 2.5 shows the list of *Search API* parameters that can be appended to the previous resource URL to create queries, e.g., the following query asks for *recent tweets* containing the word *"manchester"*, since the 4th of May 2012:

```
http://search.twitter.com/search.json?q=manchester&rpp=100&include_
entities=true&result_type=recent&since=2012-05-04
```

Twitter replies to these requests with *HTTP* messages containing the lists of *tweets*, represented as *JSON* objects, and with a maximum of 1500 *tweets* per request.

Parameters	Optional/ Required	Description
q	required	A UTF-8, URL-encoded search query of 1,000 characters maximum, including operators.
callback	optional	The response will use the JSONP format with a callback of the given name.
geocode	optional	Returns tweets by users located within a given radius of the given latitude/longitude.
lang	optional	Restricts tweets to the given language, given by an ISO 639-1 code.
locale	optional	Specify the language of the query you are sending
page	optional	The page number (starting at 1) to return, up to a max of roughly 1500 results (based on rpp * page).
result_type	optional	Specifies what type of search results you would prefer to receive. Possible values are: *mixed* (default), *recent* and *popular*
rpp	optional	The number of tweets to return per page, up to a max of 100
show_user	optional	When true, prepends ":" to the beginning of the tweet
until	optional	Returns tweets generated before the given date as YYYY-MM-DD
since_id	optional	Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of tweets that can be accessed through the API. If the limit of tweets has occurred since the *since_id*, the *since_id* will be forced to the oldest ID available
max_id	optional	Returns results with an ID less than (that is, older than) or equal to the specified ID.
include_entities	optional	When set to either *true* (*t* or 1) each tweet will include a node called "*entities*". This node offers a variety of metadata about the tweet in a discrete structure, including: urls, media and hashtags.

Table 2.5: List of Twitter Search API parameters. From [83], *Search API* documentation.

The two main benefits when using this API is that it does not require any prior authentication against Twitter Services, as well as users can modify the queries with every new request sent. However, this approach has the following limitations [83]:

- It can only be used to retrieve an index of recent *tweets*, making impossible to collect *tweets* older than 6-9 days.
- Queries are limited in complexity and cannot contain more than 10 keywords and/or operators.
- Search is *"focused in relevance and not in completeness"*, as filtering ranks are applied to only return relevant statuses.
- Rate limits are applied based on complexity and frequency of the queries. No more than 150 requests per hour can be sent from the same IP address. If these

rate limits are exceeded, the user's IP address will be blocked and further queries will be rejected for a certain period time.

Finally, Twitter recommends the following set of *Best Practices* when constructing *Search API* queries [83]:

1. Query parameters must be properly URL encoded.
2. The *since_id* parameter should be set to the value of the last *tweet* received, or the *max_id* obtained from the previous response.
3. A meaningful and unique *User Agent* string should be included. This will help to identify the traffic generated and can be used for issue reporting.
4. Queries must be limited to a maximum of 10 keywords and/or operators.

2.10.2 Streaming API

The *Streaming API* provides a real-time and low-latency delivery of *tweets*. Its main difference with the *Search API* is that it requires establishing a persistent HTTP connection against Twitter Servers. As a result, once a query is sent, a continuous stream of *tweets* will be received for as log as the connection is kept alive.

The *Streaming API* offers the following list of services depending on the type of application to develop [83]:

- **Public Streams:** giving access to streams of public *tweets*.
- **User Streams:** providing access to a single-user's stream of *tweets*.
- **Site Streams:** when applications need to connect to Twitter services and request multiple *User Streams*.

Another difference with the *Search API* is that it allows the sampling of the entire Twitter Stream, without having to continuously poll Twitter Servers and without being affected by rate limits or status filtering. However, this service also has the following limitations [83]:

- *Streaming API* is designed to provide a real-time access to streams, thus making impossible to retrieve old *tweets*.
- Authentication against Twitter Servers is always required.

- It is designed to keep persistent connections, receiving a continuous stream of *tweets* from a particular query. If additional queries are need to be created, the connection needs to be dropped and re-established again.
- IP addresses from clients opening and closing streaming connections too frequently are blocked, so further queries are rejected during a certain period of time. For this reason, its use is not recommended for applications that need to send multiple queries in response to multiple user's requests.

Chapter 3

System Design and Implementation

3.1 Methodologies and Development Tools

This section covers the different methodologies and development tools that have been applied in this dissertation. Some of them were imposed by *Cafetiere* initial design, while new ones have been introduced for the implementation of the set of features described in section 3.2.

3.1.1 *Cafetiere* Intrinsic Methodologies and Development Tools

Cafetiere is a web-based application written in *Java* and containing the following three core software components:

- *Google Web Toolkit* [1], for the Web infrastructure.
- *H2 database system*[2], for data storing.
- *Apache UIMA Framework*, for NLP techniques implementation.

Cafetiere Web Infrastructure

The web infrastructure was implemented using *Google Web Toolkit* (GWT), an open source web development framework, created by *Google*, for building complex web applications. *GWT* allows the creation of rich *AJAX* [85] applications using *Java* code, which is then compiled into highly optimized *JavaScript* that can be used across all browsers.

[1] *Google Web Toolkit* (GWT), available at: https://developers.google.com/web-toolkit
[2] *H2 database*, available at: http://www.h2database.com

Finally, *Cafetiere* was implemented according to the following design patterns for Web Applications :

- **AJAX**, for asynchronous client-server communications via the *GWT Remote Procedure Calls* (RPC) framework, which provides a high-level of abstraction relying on *Java servlets* and *HTTP* requests, for sending *Java* objects to and from the server.
- **Model-View-Controller (MVC)** [86], for the implementation of the User Interface. This design pattern allows decoupling data access and business logic from their presentation to the user.

Cafetiere Database system

A *Java SQL* database *H2* is integrated with *Cafetiere* for those component requiring permanent data storage, such as: *gazetteer*, rules, *tweets*, etc..

UIMA Framework for NLP techniques

Cafetiere executes the pipeline of text analytic processes described in section 1.2 under the *Apache UIMA* framework for *Java*. This is a collection of frameworks and software components allowing the analysis of large volumes of unstructured information.

UIMA has been designed in a way where applications can be broken into individual components that are executed as a pipeline of processes. Each component implements interfaces and self-describing metadata using XML descriptor files. *UIMA* is then responsible for managing these components and the data flow between them.

3.1.2 General Methodologies and Development Tools for *Cafetiere's* Enhancements

New methodologies and tools were introduced during the the development of *Cafetiere's* improvements described in section 1.3. Some of them were inherited from traditional web development approaches, while others relied on advanced NLP techniques.

Due to *Cafetiere* design, *Java* was the programming language chosen for this dissertation, together with the development environment *Eclipse*[3] (Indigo release), as it can be easily integrated with *UIMA* and *GWT SDKs* via plugins. Software versioning

[3] *Eclipse*, available at: http://www.eclipse.org

and revision control was done using *SVN* via *Subeclipse*[4], a plugin providing support for subversion within the *Eclipse* development environment.

3.1.3 Software Development Methodology

The software development cycle was been planned following a one-person *Agile Methodology* [87], where three different periods were established:

- **Period 1. Definition of Requirements:** a list of requirements were defined for each of the enhancements implemented. The requirements were set based on the scope of this project, *Cafetiere's* originally design and the information obtained from the literature review.
- **Period 2. Development of features, testing and evaluation:** each feature was developed to satisfy the list of requirements defined in period 1. For this purpose, the following software development cycle was applied:
    1. Once a new feature was implemented, a period for testing and evaluation was allocated.
    2. If software bugs were detected during the testing and evaluation period, the code was refactored and new test cases were created. This step was repeated until the requirements were completely satisfied.
    3. After a new feature was completed, a new software release was generated.
- **Period 3. Documentation:** during this final period, the enhancements implemented and the dissertation where both documented.

3.2 Design and Implementation

The following sections are dedicated to cover in detail the design and implementation choices made when developing the list of enhancements described in section 1.3.

[4] *Subeclipse*, available at: http://subeclipse.tigris.org

3.2.1 Enhancement 1: Collecting *tweets* from *Cafetiere*

Enhancement 1: Requirements

Cafetiere will be enhanced to allow the collection of tweets directly from its Web client. Users will be able to type the set of keywords that will be contained within the tweets requested. In addition, they will be able to select the date from/until the tweets have to be retrieved, their language, as well as the maximum number to collect.

Based on the data previously provided, the system will then send requests to Twitter Servers.

Finally, the collection of tweets retrieved will be stored within the user's database.

Enhancement 1: Design Scenario

Considering *Cafetiere* web scenario and to satisfy the requirements defined for this enhancement, the following elements were also taken into consideration:

- The collection of *tweets* had to be implemented in a way where different users could send multiple requests directly from *Cafetiere* Web-client.
- *Cafetiere* keeps a separate database for each user that has been previously registered in the system, and it uses the details provided during the logging step to know which database to access. As a result, *Cafetiere* needed to identify the user that originated the query, so the *tweets* retrieved could be stored within the correct database.
- *Cafetiere* user interface (UI) had to be modified to enable query customization, allowing users to:
  - Enter an arbitrary set of keywords.
  - Specify a date from/until when the *tweets* have to be retrieved.
  - Select the language of the *tweets*.
  - Enter the maximum number of *tweets* that are to be collected.
- The UI should be modified so users can stop the requests generated at any time, as well as send new queries containing a different set of keywords and parameters.

All these features were implemented according to the *MVC* design pattern, as well as to the principles of UI development outlined in [88] for *functionality, reliability, usability, efficiency, maintainability* and *portability*.

Enhancement 1 Implementation

As described in section 2.10, Twitter provides two different APIS to retrieve *tweets* from their servers: the *Search API* and the *Streamming API*.

Due to the design considerations mentioned in the previous section, the *Streaming API* was not applied when implementing this enhancement. Mainly because this API requires establishing persistent HTTP connections against Twitter Servers for each individual query that is sent, while blocks queries coming from IP addresses that rely on short-lived connections

Streaming API is more oriented for applications collecting streams of *tweets* from a single and pre-defined query.

However, *Cafetiere* has been built for a more dynamic scenario, where multiple users can simultaneously interact with the system in many different ways. For this reasons, the *Search API* becomes more suitable to develop the enhancement here discussed as:

- It allows the creation of multiple queries that users can manually customize.
- It relies on short-lived HTTP connections.
- Is the only API that can be used to retrieve *old tweets* (up to one week old), thus allowing the collection of past data for measuring the evolution of the *overall sentiment* perceived over time.

Figure 3.1 shows the design principle that was applied to overcome with the limitation of 150 requests per hour and per IP address imposed by this API. According to this figure, requests against Twitter Servers will be sent directly from the user's browser, by executing the following steps:

1. Firstly, a user will get access to *Cafetiere* Web client UI and will enter the set of keywords and parameters to generate a query.
2. *Cafetiere* Web client will then send a request to *Twitter Servers* with the query created .

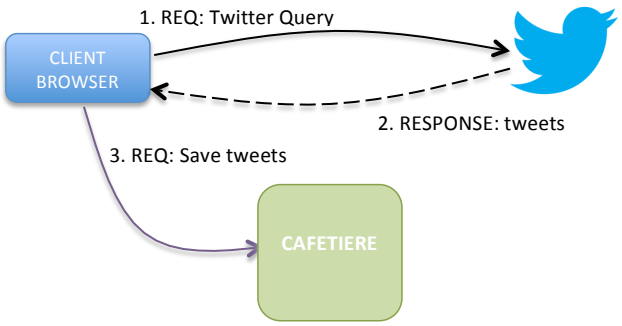

Figure 3.1: *Cafetiere* integration with Twitter Services.

3. Then, Twitter Servers will send back a response to *Cafetiere* Web client with the collection of *tweets* matching the query requested. Due to *Search API* limitations, a maximum of 1500 *tweets* can be retrieved per request sent [83].

4. Once the collection of *tweets* has been retrieved, it is sent back from *Cafetiere* Web Client to the Server, where it is stored within the user's database.

Finally, the previous four steps will be repeated in intervals of 30 seconds until the total number of *tweets* defined is collected.

In order to avoid requesting duplicate *tweets* when polling every 30 seconds, each new query is formulated by updating the parameter *since_id* with the *maximum tweetId* (*max_id*) retrieved from the previous result. Thanks to this approach, the implementation guarantees that each new request will only contain *tweets* with a *tweetID* that is higher (hence newer) than the maximum *tweetId* previously collected. Also, the following limitations are overcome:

- Rate limits imposed by the *Search API* are measured against each individual user's IP address, instead of measuring all the requests directly from *Cafetiere* server IP address.

- Polling Twitter servers every 30 seconds guarantees that each user, therefore each individual IP address, will send less than 150 requests per hour.

A state machine was implemented in order to coordinate the behaviour between the client and the server in each of the steps previously involved. Thanks to this state machine, both the client and the server are aware at any time of their current status, so they can react and recover from error messages or connectivity problems.

GWT applies a *Same Origin Policy* (SOP) access restriction, thus limiting the client-side of an application to only send requests to the server it has been designed to work with. To avoid this limitation, requests against Twitter Servers are sent using the *JSONP* (JSON with padding) interface from *GWT*.

Communication between *Cafetiere's* client and server is done using the *GWT RPC* mechanism described in section 3.1.1, and the *tweets* collected are stored within the *H2* user's database that is integrated with *Cafetiere*.

Enhancement 1: Modifying the User Interface

Cafetiere UI interface was modified to include a new tab within the top menu bar, named *Input Data*, containing a series of fields for the customization of some of the query parameters described in table 2.5. Figure 3.2 shows in more detail the composition of this tab.

The fields contained within the *Input Data* tab are:

- **Query:** a text field where the user can specify the keyword or set of keywords that will be contained within the *tweets* collected.
- **Number of *tweets* to collect:** a numeric text field where the maximum number of tweets to retrieve is set.
- **Result type:** a list box containing the type of search (see table table 2.5 for more details).
- **Since/Until:** a calendar entry with the format YYYY-MM-DD to specify the date *since/until* the *tweets* have to be retrieved.
- **Language:** a list box containing the languages to restrict the query results.

For the set of parameter values defined in figure 3.2, the following query is constructed when the *Collect tweets* button is pressed:

```
http://search.twitter.com/search.json?q=manchester&rpp=100&include_
entities=true&result_type=recent&since=2012-08-13
```

Once the request has been sent, the UI provides real-time information to the user about the number of requests sent and the total number of tweets that have been collected. During this process, the user can stop the request at any time by pressing the button *Stop*.

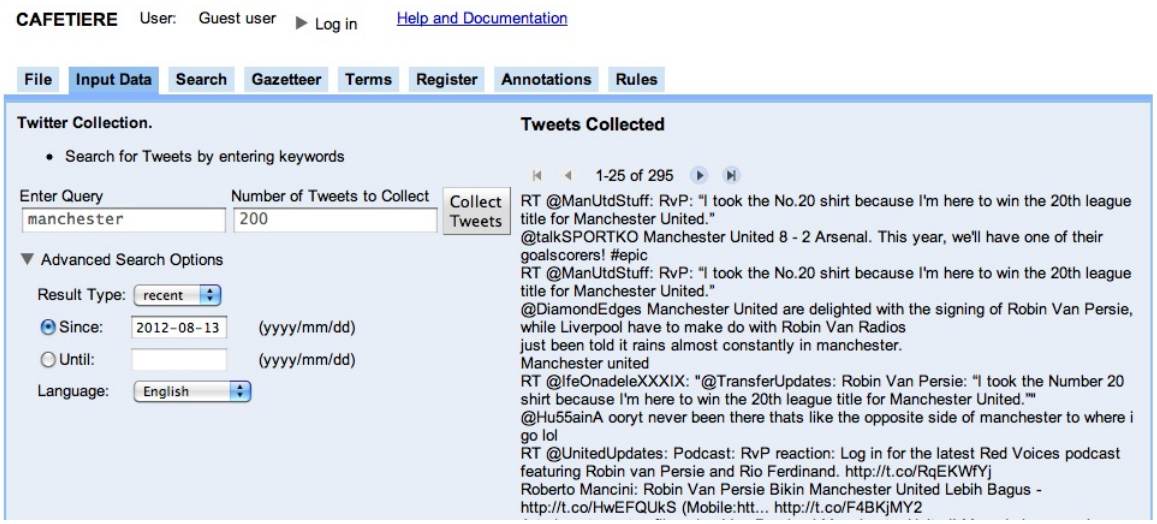

Figure 3.2: *Cafetiere* UI screenshot showing an exmaple of the *Input Data* tab.

Enhancement 1: Applying the MVC pattern

The client-side of this enhancement was implemented following the *MVC* pattern design described section 3.1.1 and illustrated by the figure 3.3.

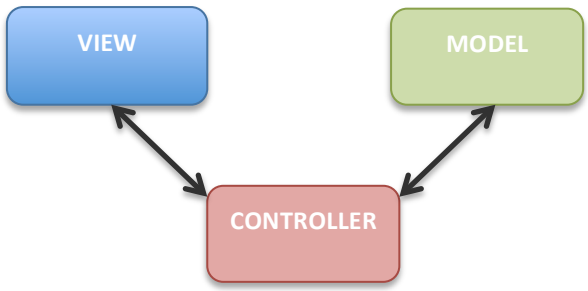

Figure 3.3: *MVC* pattern design applied for the development of *Cafetiere's* enhancement 1.

According to this pattern, the following three components where defined:

- **View**: which consists on the UI modifications described in the previous section.
- **Controller**: this component is in charge of translating user's interactions with the *View* into actions and updates with the *Model*. The *Controller* is responsible for:
  - *Authentication:* the controller retrieves user's credentials, from the *username* and *password* fields, in order to interact with the correct user's database.

  - *Validation:* all the data provided by the user via text fields, list boxes, radio buttons and calendars, is validated by this component when the user presses the button *Collect Button*.
  - *Sending Requests:* once the data has been successfully validated, the controller is in charge of creating and sending the requests to Twitter Servers.
  - *Processing Requests:* the data retrieved from Twitter servers is processed using asynchronous requests, following the *AJAX* paradigm and providing a more dynamic user experience. Thanks to this approach, the UI interface does not freeze while the requests are being sent and processed.

    The controller is also responsible of stopping all the services and resources that are enabled when the user decides to press the *Stop* button.
  - *Accessing the data base: tweets* retrieved from Twitter Servers are sent back to the server and stored within the user's database.

    The data stored is organized following the folder structure described in figure 3.4. Thanks to this schema, each tweet collected can be uniquely identified within the user's database, based on the *keywords* it contains and the *date and time* when the request was sent.
  - *Updating the View:* the controller is responsible of updating the UI, based on changes in the model and the status of the requests sent.

- **Model:** this component consists of an abstract representation of the *tweets* and the meta-data received from Twitter Servers, which is processed and stored within the user's database.

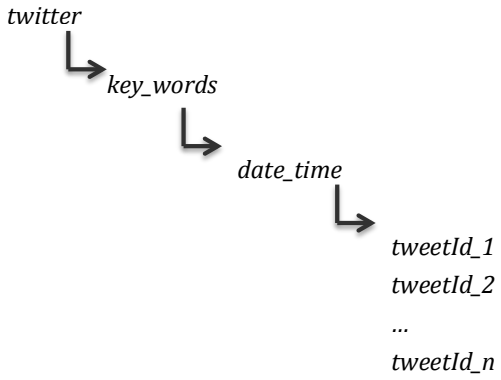

Figure 3.4: User Database folder structure to store *tweets* collected

3.2.2 Enhancement 2: Integrating *Cafetiere* and *SentiWordnet*

Enhancement 2: Requirements

The lexical resource SentiWordNet will be integrated with Cafetiere and the sentiment analysis will be computed using the information provided by this lexicon.

Enhancement 2: Design Scenario

Between the lexical resources covered in section 2.3, *Cafetiere* will be enhanced with *SentiWordNet* due to:

- The number of research papers that have used *SentiWordNet* for sentiment analysis, i.e., the works conducted by [42, 43].
- It consists of *WordNet* synsets that have been automatically annotated with a sentiment score, so the system could be expanded with any of the *Word Sense Disambiguation* approaches that rely on the information provided by this lexicon.

In addition, and based on how the current version of *Cafetiere* was implemented, the following design principles were taken into consideration for the development of this enhancement:

- To ensure its compatibility with the current version of *Cafetiere*, *SentiWordNet* was integrated using the same design principle that was applied for the *Gazeteer* lookup described in section 1.2. As a result, *SentiWordNet* entries are stored within *Cafetiere's* database.

  In addition, and in order to maximize the chances of finding words matching entries within *SentiWordNet*, a stemmed version of this lexicon was also stored in the database. As a result, the sentiment score from inflected words can be retrieved by using their stem or base root form.

- A new *UIMA* component was created to execute a pipeline of text analytical processes for computing sentiment scores. This pipeline breaks the *tweets* into tokens, normalizes them and finally makes use of the information provided by *SentiWordNet* to obtain their sentiment polarity.

Enhancement 2: Implementing the *UIMA* pipeline

This enhancement relies on a new UIMA component that was developed for normalizing and cleaning the text contained within the *tweets* collected, as well as compute their sentiment score based on the information stored in *SentiWordNet*.

Figure 3.5 shows the pipeline of text analytical processes executed by this new component, which are similar to those techniques described in section 2.2. According to this figure, each *tweet* collected is processed as follows:

Figure 3.5: *UIMA* pipeline for sentiment analysis using *SentiWordNet*.

1. **Sentence Segmentation & Tokenization:** Where *tweets* are divided into sentences and each sentence is split into elementary text units or *tokens*.

   Sentence segmentation is done using a pre-existing module from *Cafetiere*. However, the *tokenizer* that was integrated within the system was not accurate enough to identify *twitter syntax elements* (*emoticons*, *twitter topics*, *twitter mentions* or *links*). In fact, it could not tokenize those pieces of text where these elements were attached to their adjacent terms, or when *tweets* were written with no spaces between words, which are writing habits that are commonly found in social network feeds and microblogging feeds [89].

   To overcome with these limitations a new *tokenizer* was developed, so syntactic twitter elements could be identified and split in a more accurate way, i.e., sentences like: *"more information seehttp://www.aghsyl.co"* or *"we should vote yes:-)"* are now tokenized, respectively, as:

   $$\{more,\ information,\ see,\ http://www.aghsyl.co\}$$

   $$\{we,\ should,\ vote,\ yes,\ :-)\}$$

   The output of this step is a list containing the tokens constituting the *tweet* that has been processed.

2. **Normalization:** When analysing text from *social network feeds* or *microblogging feeds*, that typically contains non standard words, like abbreviations or slang-words, *text normalization* becomes an important step for obtaining more accurate results in further processing [89].

   The normalization step developed for this *UIMA* component involves the following methods:

   - *Lower-case:* a lower-case version of each token is saved as meta-data within the *Token* object generated. This information will be used during the *SentiWordNet lookup* step, as entries in the *SentiWordNet* database are stored in a lower-case format.
   - *Stemming:* A stemmed version of each individual token is also saved as meta-data within the *Token* object. This *stem* will be used during the *SentiWordNet lookup* step to maximize the chance of finding entries within the lexicon.
   - *Social Media Abbreviation Expansion:* Abbreviations commonly found in social networks, like *"lol"*, are substituted with their expanded version: *"lots of laugh"*. For this purpose, *Cafetiere*'s database was enriched with the list of the most commonly used chat slang terms for social media collected by [90].

3. **Part of Speech Tagging (PoS Tagging):** Each word is then characterize with the part of speech it belongs to. A pre-existing *PoS Tagging module* that was integrated with *Cafetiere* was used to implement this analysis.

4. **SentiWordNet lookup:** In this step, the sentiment score of a *tweet* is obtained from those tokens matching entries within the *SentiWordNet* lexicon. More specifically:

   (a) Firstly, given the token word and its *Pos*, the process looks for entries within the *SentiWordNet* database matching this criteria. If a match is found, the sentiment score of the token is calculated as the averaged score obtained from all the senses associated with the word. For example, the adjective *unable* has the following 3 senses in *SentiWordNet*:

   - *unable#1:* PosScore = 0 , NegScore=0.75
   - *unable#2:* PosScore = 0, NegScore=0.375

- *unable#3:* PosScore = 0.125, NegScore=0.25

and its final sentiment score is calculated as:

$$SentScore = \frac{\sum PosScore - \sum NegScore}{3} = -0.42$$

which produces a token with a *negative polarity*.

(b) If a match is not found in the previous step, then the process will look for entries in the *stemmed* version of *SentiWordNet*, by using the token's stem and its *PoS tag*. If a match is found, the sentiment score of this token is calculated as the averaged score obtained from all the senses found for this criteria.

For example, the verb forms *"fishing", "fished"* or *"fishes"* do not exist within *SentiWordNet*. However, their stem *"fish"* has the following 2 senses associated within the *stemmed SentiWordNet* database:

- *fish#1:* PosScore = 0, NegScore=0.25
- *fish#2:* PosScore = 0 , NegScore=0

so the token sentiment score is calculated as:

$$SentScore = \frac{\sum PosScore - \sum NegScore}{1} = -0.25$$

These approaches are commonly applied for obtaining the sentiment polarity of piece of text of document [42, 43].

5. **Sentiment Calculation:** Several works have been focused on calculating the sentiment polarity of sentences [7, 20], while others consider the document as a whole entity [9].

   Due to the limitation of a maximum of 140 characters that can be contained within a *tweet*, the enhancement here developed computes the final sentiment of a message post, by calculating the average score obtained between all the tokens carrying sentiment, without considering individual sentences. This approach has been applied with the aim of capturing more contextual *sentiment* information from the short piece text contained within *tweets*.

3.2.3 Enhancement 3: Word Sense Disambiguation and *SentiWordNet*

Enhancement 3: Requirements

Word Sense Disambiguation techniques will be combined with SentiWordNet to improve Cafetiere's sentiment analysis capabilities

Enhancement 3: Design Scenario

When sentiment analysis is conducted by using lexical resources like *SentiWordNet*, where words can have different *senses* with different sentiment scores associated, the use of WSD techniques before a lexicon lookup can help to improve the sentiment calculation. The reason behind this fact is that the correct word sense is disambiguated based on the context where words are used, so its precise sentiment score can be retrieved from the lexicon.

In section 2.5.2 was described that although *Supervised* WSD methods provide better results than *Knowledge-based* approaches, these are commonly applied because they rely on knowledge resources like *WordNet* and their extensions, which are widely used within the research community. Due to this fact and in order to implement a disambiguation technique that could expand the integration of *SentiWordNet* from *enhancement 2*, the following *Knowledge-based disambiguation approaches* where added to enrich *Cafetiere's* capabilities for sentiment analysis:

- **Similarity Measures: Adapted Lesk and JiangConrath**:

  According to [58], both *Adapted Lesk* and *JiangConrath* performed better than any of the measures described in section 2.5.2. This work also remarked that researchers have traditionally focused efforts on solving the disambiguation problem of only nouns, *"assuming that results would generalize to the network of hierarchies of other parts of speech"*.

  However [91, 57, 58] showed that there is a difference in how *Knowledge-based* approaches perform when they are applied over different parts of speech. In fact, [57] found that *Adapted Lesk* performed better when disambiguating nouns and verbs, while *JiangConrath* becomes more suitable for the disambiguation of adjectives [58]. Due to this fact and the difference in the properties hold by nouns, verbs and adjectives, [91] suggests that the disambiguation of different part of speech should be treated as separate problems .

Bearing in mind this difference in performance, this new enhancement was developed by combining *Adapted Lesk*, for the disambiguation of *nouns* and *verbs*, together with *JiangConrath* for the disambiguation of *adjectives*. Adverbs will not be disambiguated because similarity measures over *WordNet* do not support this type of speech, as there is an absence of semantic relatedness between adverbs in *WordNet*.

- **Structural Approaches: The SSI Algorithm.**

  The work conducted by [69] demonstrated that the performance of the *SSI algorithm* is comparable with the current state of the art of WSD techniques. Its main difference with the *similarity measures* approaches is that it takes into consideration more contextual information for the disambiguation of words. For this reason, and in order to compare its performance with the previous WSD technique implemented, this structural approach was also integrated within *Cafetiere*.

Following the design principle applied for the development of *Enhancement 2*, two different *UIMA* components were implemented in order to integrate the WSD approaches previously mentioned. Figure 3.6 shows a generalization of the component created for each of these approaches.

Finally, *Cafetiere's* UI interface was modified to give users the option to select which WSD approach to apply when conducting sentiment analysis.

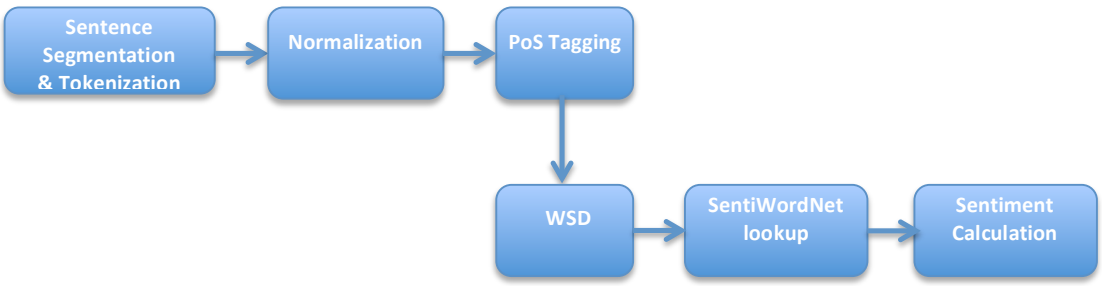

Figure 3.6: *UIMA* pipeline for sentiment analysis using *Word Sense Disambiguation* combined with *SentiWordNet*.

Enhancement 3: Implementing an UIMA component with Adapted Lesk and Jiang-Conrath

The UIMA component developed for this enhancement is quite similar to the one implemented for *Enhancement 2* . A simple inspection of figures 3.6 and 3.5 shows that

both components only differ in the presence of an additional WSD step, as well as how the *SentiWordNet lookup* uses the disambiguation obtained to retrieve sentiment scores.

In this section only the steps that differ with *Enhancement 2* are described, which involved:

- **WSD with Adapted Lesk and Jiang-Conrath:**

  This disambiguation technique was implemented by using the open source *Java* API *WordNet::Similarity* developed by [92]. This API measures the semantic distance between pairs of words sharing the same part of speech, by choosing any of the similarity measures described in section 2.5.2, thus supporting both *Adapted Lesk* and *Jiang-Conrath*.

  As a result, and considering the design principles described in the previous section, the WSD step was developed as follows:

  1. Firstly, a list of pending words to disambiguate, *P*, and a list of disambiguated words, *D*, are created. In this first step both *P* and *D* are empty, as there are still no words to be disambiguated.
  2. Then, for each word token generated from a *tweet*, the process looks for entries in *SentiWordNet* matching the word token and its part of speech associated. If a match is found, the word and is part of speech are both saved within *P*, i.e., for the *tweet*:

     If #YesToAV could just use this cartoon to explain their argument, the campaign would be over much quicker http://j.mp/evdX9a

     the list *P* will contain the following word tokens found in *SentiWordNet*:

     P={*just#adv, use#v, cartoon#n, explain#v, argument#n, campaign#n, be#v, much#adj* }
  3. If no matches are found, then the process will look for entries in the *stemmed* version of *SentiWordNet*, by using the token's stem and its associated part of speech.

     If a match is found, the process will then combine the data from *SentiWordNet* and its *stemmed* version, to obtain the list of all the different expanded words from *SentiWordNet* that derived from the token stem, i.e, following

the previous example, the adjective *quicker* was not found within *SentiWordNet* but its stem *quick* was found in the *stemmed version* of the lexicon.

The list of expanded words retrieved, together with their part of speech associated, are saved within *P*. So, in the context of the previous example, *P* will become:

$$P=\{\textit{just\#adv, use\#v, cartoon\#n, explain\#v, argument\#n, campaign\#n, be\#v, much\#adj, quick\#a}\ \}$$

4. In this step the words in *P* are disambiguated using:
   - *Adapted Lesk* to calculate the semantic similarity between all combinations of pairs of noun-noun and verb-verb in *P*.
   - *Jiang-Conrath* to calculate the semantic similarity between all combinations of pairs of adjectives in *P*.

   For each token word in the *tweet*, the process will use the API to calculate the semantic similarity between this word and all the other words sharing the same part of speech, i.e, for the verb *use*, the following pairs will be computed:
   - *use-be*
   - *use-explain*

   For each pair computed, the API retrieves the pair of word senses with the highest similarity obtained, i.e, when calculating the similarity between the verbs *use* and *be*, the API retrieves: *use#v#3,be#v#10* as the pair of senses with the highest similarity computed (0.251), between all the senses of *use* and all the senses of *be* and where:
   - *use#v#3* represents the word *use*, with the part of speech *verb* (v) and the sense number 3.
   - *be#v#10* represents the word *be*, with the part of speech *verb* (v) and the sense number 10.

   Based on this approach, the similarity obtained between *use* and *explain* was: *use#v#1,explain#v#1* with a score of 0.096.

   From all the measures calculated between a word and the other words sharing the same part of speech, the sense chosen for the disambiguation will

be the sense from all the sense pairs generated with the highest similarity measured, i.e, in the previous example the verb *use* is disambiguated as *use#v#3*, as it obtained the highest score of 0.251, between all the sense-pairs retrieved.

Once the word has been disambiguated, the word, its part of speech and its sense number are stored within the list *D*.

However, it is important to remark that those pairs of words that contain no semantic relatedness within *WordNet* will produce a similarity score of zero. If an element of *P* has a maximum similarity of zero, when is measured against all the other words from *P* sharing the same part of speech, it means that the word could not be disambiguated based on the information provided by *WordNet*, i.e., in the example previously considered, the adverb *just* has a similarity of zero as no semantic relatedness exists for adverbs within WordNet.

In this case, only the word and its part of speech are stored within the list *D*.

The list *D* of disambiguated words obtained for the example is:

$$\{just\#r,\ use\#v\#3,\ cartoon\#n\#1,\ explain\#v\#1,\ argument\#n\#2,\ campaign\#n\#2,\ be\#v\#10,\ much\#a\#1,\ quick\#a\#3\}$$

- **SentiWordNet lookup:** In this final step each element stored in *D* will be used to compute the sentiment of the *tweet* as follows:
  - If the word stored has a sense number associated, then it was successfully disambiguated and its correct sentiment score will be retrieved from *SentiWordNet* based on the word, its part of speech and its sense number.
  - If the word stored has no sense number associated, then the word could not be disambiguated. Its sentiment score will be calculated as the average score obtained from all the senses associated with the word and its part of speech.

Finally, the total sentiment score of the *tweet* will be computed following the same method applied in *Enhancement 2*: by calculating the average score between all the tokens carrying sentiment.

Enhancement 3: Implementing UIMA component with the SSI Algorithm

The difference between the UIMA component here implemented and the previous one is that it relies on *SSI algorithm* for the disambiguation step, instead of applying the *Adapted Lesk and Jiang-Conrath* method. Due to this reason, only the WSD step illustrated in figure 3.6 is described in this section.

The initial approach was to try to replicate the *SSI algorithm* based on the steps given in [66], which required the creation of a database that could contain all the semantic relations used by the algorithm. In fact, several days of work were dedicated to find and collect the data that defines these relations.

However, due to constraints in the time available to conduct this dissertation, and the fact that the algorithm became available through the *BabelNet API* [68] in July 2012, it was decided not to put more effort in trying to reproduce the *SSI algorithm*. Instead, it was agreed that a better approach will be to focus on finding the way to combine this algorithm with additional knowledge-based resources, that could improve the performance of the WSD step.

As described in section 2.5.2, *BabelNet API* implements the *SSI algorithm* in combination with two graph-based knowledge resources: *BabelNet* and *WordNet++*.

For the purpose of this dissertation, where all the enhancements developed for *Cafetiere* aim to improve its sentiment analysis capabilities over *tweets* written in English, the WSD step was developed using the SSI in combination with *WordNet++*, as it consists of a smaller subset of *BabelNet* containing only English words.

The SSI algorithm takes as an input the list of all the word tokens from a *tweet*, their associate part of speech and the semantic relations from *WordNet++* to produce a list of disambiguated words. Based on this principle, the WSD step here developed executes the following set of steps:

1. Firstly, a graph with all the pending words to disambiguate, *G*, and a list of disambiguated words, *D*, are created. In this first step both the graph and the list are empty, as there are still no words to be disambiguated.

2. Then, for each word token generated from a *tweet*, the process looks for entries in *SentiWordNet* matching the word token and its part of speech associated. If a match is found, the word is added to the graph as a child node *c* of the previous token processed *p*. Figure 3.7 ilustrates this process.

3. If no matches are found, then the process will look for entries in the *stemmed*

Figure 3.7: Example of the graph created to compute all the possible sentences for the SSI algorithm. p represents a parent node and c its child node.

version of *SentiWordNet*, by using the token's stem and its associated part of speech.

If a match is found, the process will then combine the data from *SentiWordNet* and its *stemmed* version to obtain the list of all the different expanded words from *SentiWordNet* that derived from the token stem, i.e, following the previous example, the adjective *quicker* was not found within *SentiWordNet* but its stem *quick* was found in the *stemmed version* of the lexicon.

In this case, the different expanded versions retrieved are added as child nodes $C = \{c_1, c_2, ..., c_n\}$ of the previous word token processed (node p).

Then the next token processed, *ch*, will be appended as child node of each of the nodes $c_i \in C$ previously created. This process will be repeated until all the word tokens from the *tweet* are processed. Figure 3.8 illustrates this concept.

4. Thanks to the graph generated, it is then very easy to compute all the possible sentences that can be created based on the expanded versions of words associated with the tokens stemmed. As a result, each path in the graph from the root node to a leaf, represents one of this possible sentences.

   This step then processes all the possible paths and applies the SSI algorithm for each sentence created. The sentence obtaining the highest score of semantic relatedness is considered to be the most probable sentence within the context where it is used, and the disambiguation produced by the algorithm for this path is the one chosen.

   Finally, the list of disambiguated words, with their part of speech and its sense associated are then added to D.

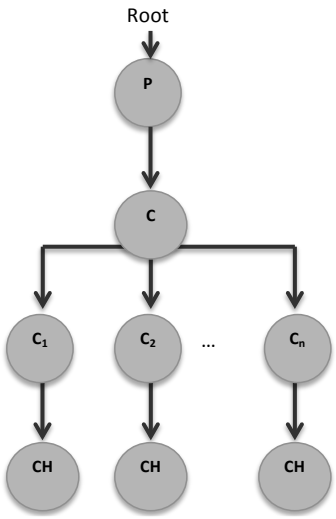

Figure 3.8: Example of the graph created to compute all the possible sentences for the SSI algorithm when expanded versions of stems are added.

However, it is important to remark that there might be some words that could not be disambiguated by the SSI algorithm if no semantic relatedness could be extracted from *WordNet++*. In this case, only the word and its part of speech are stored within the list *D*.

Finally, the *SentiWordNet lookup* step will be executed following the same method applied in *Enhancement 3*.

3.2.4 Enhancement 4: Sentiment Analysis with Supervised Classification

Enhancement 4: Requirements

Cafetiere will be enhanced with machine learning capabilities for sentiment analysis, in order to build robust classifiers for subjective detection and polarity evaluation. This will allow exploiting machine-learning algorithms that make use of text features and Twitter syntax features for the detection and evaluation of sentiment, similar to the approaches described in [17, 19].

Enhancement 4: Design Scenario

The aim of this enhancement is to build a two step classifier for subjectivity detection and polarity evaluation, similar to the approaches described in section 2.6. For this

purpose, the *UIMA* component described in figure 3.9 was developed.

Mallet [93] and *Weka* [94] are the two machine-learning *Java* APIS that were considered for building the classifiers. Both packages are widely used within the research community and they mainly differ in the amount of documentation available.

However, considering the time constraints to conduct this dissertation, the two classifiers built for this enhancement were implemented using *Weka*, as this package is significantly better documented than *Mallet*.

Each classifier was developed with the set algorithms for text classification described in section 2.6.3, with the aim of comparing their performance and select the one that provides the most efficient sentiment detection.

The classifiers were trained using 24,856 *tweets* from the *#Yes2AV* topic and collected during the *AV referendum*[5]. These *tweets* were manually annotated by three different human subjects with scores measuring the subjectivity and polarity of their content.

According to figure 3.9, the three first steps of the *UIMA* component here implemented are similar to those created for *Enhancement 2 and 3*, and it only differs in the presence of the *Feature selection* step and the two *Classification steps*. As a result, only detailed description of this new processes is provided in the following sections.

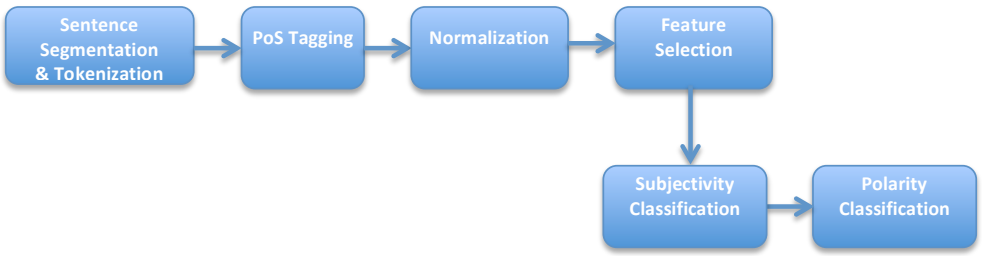

Figure 3.9: *UIMA* component implemented for sentiment analysis using a two step classifier.

Enhancement 4: Feature analysis selection

Feature selection becomes a critical step when building robust classifiers for sentiment analysis. For the purpose of this dissertation, *tweets* will be represented based on the set of features proposed by [17], as they produced a robust performance when are applied for subjectivity detection and sentiment polarity evaluation tasks.

[5] *Alternative Vote (AV) referendum*. A referendum held in the UK during May 2011 to change the method of electing MPs.

However, a modification will be introduced to the set of features proposed, with the aim of capturing more contextual information when considering the sentiment expressed by a word and its immediate adjacent neighbours.

The set of features used can be grouped into three different categories:

- **Meta-features:** each word in a *tweet* is mapped to its part of speech, prior subjectivity and prior polarity. The prior subjectivity and polarity of a word is obtained from the *MPQA subjectivity lexicon*[6] [7]. Also, the prior polarity of a word is switched from positive to negative, or vice-versa, if the word is preceded by a negative expression like *"didn't", "must not", "never", etc.*.

- **Meta-feature Modification:** The modification introduced in this dissertation differs from the approach described by [17] in that it aims to capture more contextual information, by adding the prior polarity of the intermediate adjacent words to the prior polarity of the word being processed.

- **Twitter Syntax Features:** each word in the *tweet* is also mapped to its correspondent *tweet syntax feature* if it matches a *retweet*[7]*, hashtag, reply, link, punctuation* (exclamations and question marks), *emoticons* and *upper cases* (if the word starts with upper case).

Finally, the frequency of each feature in a *tweet* is divided by the number of the words in the *tweet*.

The previous set of features will be used for building the two classifiers here proposed for subjectivity detection and sentiment polarity evaluation.

Enhancement 4: Implementing the Subjectivity Classifier

The aim of this first classifier is to predict whether a *tweet* expresses a subjective or an objective idea. For this purpose, a classifier was built using the collection of *tweets* from the *AV referendum* as training data.

After inspecting these collection of *tweets*, it was revealed some level of disagreement between the subjectivity annotations given by the three different human subjects. As a result, and in order to improve the quality of the classification, it was decided to

[6] *The MPQA subjectivity lexicon*, available at: http://www.cs.pitt.edu/mpqa/

[7] A *retweet* is a re-posting of someone else's *tweet* by prepending *RT* at the beginning to indicate that is re-posting someone else's content.

remove those *tweets* that did not contain annotations that had been agreed between the three human subjects involved.

The training data obtained after the disagreement removal was reduced to 21,166 *tweets*, where 6,767 (31.97%) are labelled as *objective* and 14,399 (68.02%) are labelled as *subjective*. This result revealed that the training samples contained a higher distribution of subjective content.

In order to train and test the classifier with a set of samples that could provide a good representation of the data collected, a 10 fold cross validation technique was applied, where each fold built contained the same proportions of subjective and objective *tweets* as the complete data set. This 10 fold cross validation technique has become the standard method for training and evaluating classifiers when the data samples are no equally distributed [95].

Finally, the classifier was built using three different algorithms: *SVM, MaxEnt and Naive Bayes*, in order to compare their performance and then select the most robust.

Enhancement 4: Implementing the Polarity Classifier

The next step was to build a second classifier to predict if the sentiment polarity expressed in a subjective *tweet* is positive or negative.

Following a similar approach to the one applied when building the subjectivity classifier, a disagreement removal step was implemented, producing a total of 14,399 *tweets* with subjective content, where 12,649 (87.84%) are labelled as *positive* and 1750 (12.15%) as *negative*.

The training data shows that a higher proportion of samples have a positive polarity. This result was expected, as the topic *#Yes2AV* reflects a positive intention of vote towards the *Av referendum*. Nevertheless, a 10 fold cross validation technique was applied for training and testing the classifier, where each fold contained the same proportions of positive and negative *tweets* as the overall data set.

Finally, the classifier was build using the same three different algorithms that were considered for the subjectivity classifier: *SVM, MaxEnt and Naive Bayes*, with the aim of comparing their performance and select the most robust.

Chapter 4

System Evaluation

4.1 General Considerations

The series of *Cafetiere's* enhancements developed in this dissertation were evaluated according to the following principles:

1. **Software Testing:** a series of test cases were created to verify that, the list of requirements defined in section 1.3 are satisfied by each enhancement implemented, as well as to detect the presence of software bugs within the code developed.

2. **Classification Evaluation:** *Cafetiere* sentiment analysis *enhancements 2-4* were evaluated using the 21,166 *tweets* collected during the *AV referendum*. This collection, that has been manually annotated with scores representing the subjectivity and polarity of its content, constitutes the *Gold Standard* for the analysis conducted in this dissertation.

   Table 4.1 shows the distribution of *sentiment* across the *tweets* contained within this data set. According to this table, the proportion of *tweets* expressing a *positive* sentiment is much higher than those carrying a *negative or neutral* opinion. This result was expected, as this *Gold Standard* contains *tweets* talking about the topic *#Yes2AV*, which expresses a positive opinion towards the *AV referendum*.

   To analyse the performance obtained by *enhancements 2-4*, each classification method was evaluated in terms of *precision, recall* and *F-measure*, over a representative test data set taken from the *Gold Standard* samples. This approach

constitutes the traditional evaluation method that is applied in Information Retrieval and Machine-learning classification problems, when data samples are not equally distributed between the classes [71, 95].

3. **User Interface Evaluation:** the series of *Cafetiere's* UI modifications introduced during the development of this dissertation, were evaluated according to the principles of *functionality, reliability, usability, efficiency, maintainability* and *portability* defined in [88].

Table 4.1: Distribution of *sentiment* in the collection of *tweets* used as *Gold Standard*.

	Number of *tweets*	Percentage
Positive	12649	59.76%
Negative	1750	8.20%
Neutral	6767	31.97%

4.2 Software Evaluation

A one-person *Agile Methodology* was applied for the development of this dissertation. This method consisted of an iterative process where a set of tests cases were written for evaluating the developments, and also, to verify that the list of requirements defined in section 1.3 were satisfied. Then, and depending on the results and the bugs found, the code was refactored and new test cases were written until the list of requirements were completely satisfied.

4.3 *Cafetiere's* Enhancements Evaluation

4.3.1 Enhancement 1 Evaluation: Collecting *tweets*

The implementation of this enhancement mainly involved the development of a UI, as well as the software required in both *Cafetiere's* client and server sides to allow the collection of *tweets* directly from the system.

The enhancement was tested to verify that all its requirements were satisfied, as well as to check its correct behaviour when:

- *Validating user input:* the data that is introduced by a user via *Cafetiere's* UI to customize the queries that are sent, is validated in order to check if the requests generated satisfy the list of requirements established by *Twitter Search API* (see section 2.10.1).

- *Reacting against errors produced while sending requests against Twitter Servers and when processing the data received:* A state machine was implemented to coordinate the interaction between the client and the server when sending and processing requests. Thanks to this approach, both the client and the server are aware of their state at any time, so they can react and recover from error messages, or connectivity problems, that could happen between *Cafetiere's* Web client and server, or between *Cafetiere* and Twitter Servers.

  The behaviour of this state machine was tested by creating a series of test scenarios that reproduced error messages and connectivity losses.

- *Database Integration: tweets* collected are saved within *Cafetiere* user's database. A series of test scenarios were created to verify that no data was stored if error messages or incorrect data was received from Twitter Servers, as well as to check that the database was never left in an inconsistent state that could stop its further use.

4.3.2 Enhancement 2 Evaluation: Sentiment Analysis using *SentiWordNet*

When the *Gold Standard* data set was evaluated using this enhancement, a total of 10,303 (48.67%) *tweets* where correctly classified with an error rate of 51.32%.

A closer look to the confusion matrix obtained (see table 4.2), reveals that the use of *SentiWordNet* has a tendency to classify *tweets* as *positive*, thus increasing the error rate when detecting *negative* or *neutral* sentiments. These results also correlate with the values of *precision*, *recall* and *F-measure* obtained for each of the three classes here considered. Table 4.6 shows how the *positive* class has a very high *F-measure* (over 0.7), while both the *negative* and *neutral* classes have a very low classification performance with F-measure values below 0.07.

The results here shown demonstrate the low performance obtained when this approach is applied to conduct sentiment analysis over *tweets*.

Table 4.2: Confusion Matrix obtained when using *SentiWordNet* and no WSD technique for the sentiment analysis of *#YesToAV* tweets.

	Positive	Negative	Neutral
Positive	9464	2606	579
Negative	1171	518	61
Neutral	4298	2148	321

Table 4.3: Precision, Recall and F-Measure obtained in when using *SentiWordNet* and no WSD technique for the sentiment analysis of *#YesToAV* tweets.

	Precision	Recall	F-measure
Positive	0.748	0.756	0.752
Negative	0.296	0.034	0.061
Neutral	0.047	0.022	0.023

The main problem with this technique, is that the sentiment obtained from *SentiWordNet* is an averaged value between the sentiment scores retrieved from all the word senses composing a *tweet*. As no WSD is being applied, the system is not able to calculate the precise sentiment score for the correct word sense, thus increasing the chance of computing the wrong sentiment carried.

In order to remove the biased towards *positive*, the *Gold Standard* data set was processed again using a modified version of *SentiWordNet* that did not contain an entry for the word *"yes"*. The idea behind this approach was to avoid the system to consider the positive sentiment score of this word, which is likely to be present in every *tweet* due to the *#YesToAV* topic that composes the *Gold Standard* data set.

However, the results obtained with this modification were worse than in the previous scenario, mainly because the word *"yes"* is present in every *tweet* within a *Twitter topic* element that is preceded by a *hashtag* (*#YesToAV*). *Twitter topics* are correctly identify by the tokenizer implemented and will not match any entry within the lexicon due to the *hasgtah* character. As a result, removing the word *"yes"* from the lexicon alters the positive sentiment score from those *tweets* that contain this word out of the *Twitter topic* element, thus computing the wrong sentiment and increasing the error rate.

Another problem with *SentiWordNet* is that it has been automatically annotated from *WordNet* synsets, so some words have an incorrect sentiment score assigned,

i.e., words like *"glad"* or *"won"* have both negative sentiment scores associated, although the definitions and examples of use provided by *SentiWordNet* are sentences that clearly express a positive opinion. As a result, *tweets* like:

- *At Scottish launch of YesToAV campaign. The wind was not helping, but glad the banner eventually stayed down... http://fb.me/HXnbPQ4l*
- *Hope our UK referendum goes YestToAV. Tories have only ever won 2 real majorities here. Is why they're fighting AV so hard.*

are examples of instances containing a clear *positive* opinion towards the topic of matter, but were classified as *negative* due to the wrong sentiment scores obtained from *SentiWordNet*.

Also, one of the disadvantages of this approach is that it relies on a singe lexicon. Combining multiple lexicons, together with domain specific terms, could help to produce a more accurate sentiment evaluation as in [96].

Finally, the performance obtained could have been improved by expanding the context of the analysis and considering not just single words, but also the presence of negation forms or sarcastic comments. These are expressions typically found in political posts and can change the whole polarity of the opinion expressed. Also, considering the presence of *emoticons*, exclamations marks or words written in upper case, which are also indicators of sentiment, could have helped to improve the results obtained.

However, due to constraints in the time available to conduct this dissertation, only the effect of introducing a WSD technique, as a prior step to the lexicon lookup, has been considered for the evaluation of *enhancement 3* described in the following section.

The analysis of negation forms, the sentiment carried by adjacent words, as well as as the presence of *emoticons*, upper cases and exclamation marks have been only considered for the supervised sentiment classification developed for *enhancement 4* and evaluated in section 4.3.5.

4.3.3 Enhancement 3 Evaluation: *SentiWordNet* and WSD

The evaluation here conducted describes the results obtained when the *Gold Standard* data set was processed with *SentiWordNet* and the two WSD techniques implemented for *enhancement 3*.

Tables 4.5 and 4.6 show the confusion matrix and evaluation measures produced when *Adapted Lesk* and *Jiang & Conrath* were combined for the disambiguation step.

According to these tables, the performance obtained is very low and, as it happened with *enhancement 2*, there is a tendency to wrongly classify *tweets* towards a positive sentiment score.

When combining the *SSI Algorithm* with *WordNet++*, the performance obtained was slightly better. This result was expected, as this approach uses more contextual information to conduct the disambiguation, i.e., the following *tweet*:

If #YesToAV could just use this cartoon to explain their argument, the campaign would be over much quicker http://j.mp/evdX9a

is disambiguated by each technique as follows:

1. *Adapted Lesk* and *Jiang & Conrath*:

   [just:adverb, use#3:verb, cartoon#1:noun, explain#1:verb, argument#2:noun, campaign#2:noun, be#10:verb, much#1:adjective, quick#3:adjective]

2. *SSI algorithm* and *WordNet++*:

   [just:adverb use#1:verb, cartoon#1:noun, explain#1:verb, argument#7:noun, campaign#2:noun, be#1:verb, much#1:adjective, quick#5:adjective]

The previous disambiguations obtained reveals that each approach produced different results for the words *use, argument, be* and *quick*. In fact, and by analysing the word sense definition provided by *SentiWordNet* (see table 4.4), it is clear that the disambiguation obtained when combining the *SSI algorithm* with *WordNet++* is more accurate for the context here considered.

WSD	Sense	POS	Definition
SSI Algorithm & WordNet++	use#1	verb	put into service; make work or employ for a particular purpose or for its inherent or natural purpose
	be#1	verb	have the quality of being
	quick#5	adj	performed with little or no delay
Adapted Lesk & Jiang and Conrath	use#3	verb	use up, consume fully
	be#10	verb	spend or use time
	quick#3	adj	moving quickly and lightly

Table 4.4: Sense Disambiguation obtained when using: *Adapted Lesk* with *Jiang & Conrath* and *SSI Algorithm* with *WordNet++*.

However, neither of the two techniques implemented were able to provide a disambiguation for the adverb *just*. In addition, they failed to obtain the correct sense for the noun *argument*, which in this context should be: *argument #3: "a discussion in which reasons are advanced for and against some proposition or proposal"* and instead they provided:

- *SSI Algorithm* and *WordNet++: argument#7*: *"a course of reasoning aimed at demonstrating a truth or falsehood; the methodical process of logical reasoning"*.
- *Jiang & Conrath: argument#2*: *"a contentious speech act; a dispute where there is strong disagreement"*.

However, although there is an improvement in the disambiguation obtained with the *SSI algorithm*, the performance in the final sentiment classification is still low, with F-Measures below 0.1 for the *negative* and *neutral* class. Furthermore, the tendency towards the *positive sentiment* class is still present in the system. Tables 4.7 and 4.8 show the confusion matrix and the low evaluation measures obtained for this approach.

Table 4.5: Confusion Matrix obtained when using *SentiWordNet* and WSD techniques (*Adapted Lesk* and *Jiang & Conrath*) for the sentiment analysis of *#YesToAV tweets*

	Positive	Negative	Neutral
Positive	8062	3077	1510
Negative	986	569	195
Neutral	3554	2176	1037

Table 4.6: Precision, Recall and F-Measure obtained in when using *SentiWordNet*and WSD techniques (*Adapted Lesk* and *Jiang & Conrath*) for the sentiment analysis of *#YesToAV tweets*.

	Precision	Recall	F-measure
Positive	0.637	0.669	0.652
Negative	0.325	0.038	0.068
Neutral	0.153	0.075	0.1

The evaluation here conducted also showed an unexpected result: the classification error obtained with the two different WSD approaches is higher than the one obtained in *enhancement 2*, where no disambiguation technique was implemented. The reason

Table 4.7: Confusion Matrix obtained when using *SentiWordNet* and WSD techniques (*SSI Algorithm* and *WordNet++*) for the sentiment analysis of *#YesToAV tweets*.

	Positive	Negative	Neutral
Positive	8389	2512	1748
Negative	1043	497	210
Neutral	3632	2104	1031

Table 4.8: Precision, Recall and F-Measure obtained in when using *SentiWordNet* and WSD techniques (*SSI Algorithm* and *WordNet++*) for the sentiment analysis of *#YesToAV tweets*.

	Precision	Recall	F-measure
Positive	0.663	0.686	0.674
Negative	0.284	0.032	0.057
Neutral	0.152	0.076	0.101

behind this behaviour is related with the short length of *tweets*, which in many cases does not provide enough context to conduct a more accurate disambiguation of words. This effect was seen in the previous example, when no disambiguation was obtained for the adverb *just* and an incorrect sense was given to the noun *argument*.

Enhancement 2 provides slightly better results simply because the sentiment score it computes is based on the averaged score obtained from all word senses, thus on average, it tends to obtain more sentiment information than using the precise word-sense disambiguated. Nevertheless, it is expected that using a more accurate WSD approach should improve the results obtained in *enhancement 2*.

Table 4.9 resumes the difference of performance obtained for *enhancements 2* and *3*, when estimating the sentiment score from the *tweets* contained in the *Gold Standard* considered for this dissertation.

Table 4.9: Comparison between *enhancements 2* and *3* for the sentiment analysis of *#Yes2AV tweets*

Enhancement	Correctly Classified	Error Rate
2	48.67%	51.32%
3 with *SSI Algorithm* and *WordNet++*	46.85%	53.14%
3 with *Adapted Lesk* and *Jiang & Conrath*	45.67%	54.32%

The approaches here evaluated could be combined with the set of improvements

proposed for *enhancement 2* in section 4.3.3, together with a more robust WSD approach, like some of the supervised methods described in section 2.5.2, in order to provide a more accurate sentiment classification.

4.3.4 Final considerations about Enhancements 2 and 3

The Effect of the PoS Tagger.

When evaluating the results obtained from *enhancements 2* and *3*, it was noticed that the PoS tagger integrated with *Cafetiere* is not optimized to deal with the presence of *Twitter syntax* elements (*emoticons, hashtags, links, retweets and user mentions*). These elements, which do not correspond with any of the syntactic forms that can appear in a sentence, simply denote the topic of the *tweet* and provide additional metadata about the subject of matter.

The main problem when using this PoS tagger, is that it processes those elements as normal words in a sentence and gives them a tag (*noun, verb, pronoun, etc..*) that does not represent their nature. This effect changes the whole syntactic structure of the sentence, altering the disambiguation obtained and the sentiment scores retrieved from the lexicon lookup step. As a result, the impact of this wrongly tagged entities is to produce a wrong sentiment score for the *tweets* processed, thus increasing the final error rate of the classification, i.e., in the following *tweet*:

@PopeJimXXIII vote yes to spite the big supporters on the no campaign and give environmental politics a chance #av,

where the PoS tagger has identified *@PopeJimXXIII* and *#av* as nouns, while has wrongly tagged *supporters* as an adjective and *politics* as a verb.

It would be then interesting to modify the PoS tagger with rules that could allow the correct recognition of these twitter syntax elements, and then, analyse its effects on the final sentiment obtained in *enhancements 2* and *3*. Due to constraints in the time available to conduct this dissertation, this modification was not considered for the current implementation, but it would be an interesting approach to apply in further developments.

Computational Cost

The current version of *Cafetiere's* database does not allow concurrent access, as a result, the computational performance obtained with *enhancements 2* and *3* is not very

efficient. However, this limitation could be easily overcome by upgrading the data base to a more advanced version that could allow concurrency.

Moreover, the WSD techniques implemented in *enhancement 3* have shown a different computational cost depending on the approach applied. For instance, *Adapted Lesk* tends to be very quick when computing similarities between words, while *Jiang &Conrath* becomes less efficient, as its computation can be highly heuristic and some similarities can take very long to calculate. On the other hand, the *SSI algorithm* is the fastest technique here evaluated, but the computational cost derived from the path calculation can be quite high.

Nevertheless, *enhancements 2* and *3* have been designed with the idea of facilitating their parallelization, so they can be easily split into concurrent tasks and improve *Cafetiere's* performance.

4.3.5 Enhancement 4 Evaluation: Sentiment Analysis using Supervised Classification

This section is dedicated to the evaluate the two step-classifier that was implemented as part of *enhancement 4*.

Subjectivity Classifier

To illustrate which of the proposed features are more relevant for this classification, the top-10 features, in terms of information gain, obtained from the training data set are: *determiner*, *positive polarity*, *upper case*, *hashtag*, *noun*, *link*, *strong subjectivity*,*reply*, *punctuation* and *verb*.

This analysis reveals that the presence of twitter syntax features are more relevant for subjectivity detection than any other features here considered.

Table 4.10 shows the results obtained for each of the algorithms applied to build the classifier. The best performance was given by *NaiveBayes*, with an error rate of *29.59%*, followed by *MaxEnt* and *SVM*.

Table 4.10: Error rates obtained for Subjectivity Classification

Algorithm	Error Rate
Naive Bayes	29.69 %
MaxEnt	30.51%
SVM	32%

A closer look to the *Naive Bayes* classifier built, reveals its tendency to wrongly classify *tweets* towards the *subjective* class. Table 4.11 shows this effect, where the *subjective* class has a higher *precision* and *recall* than the *subjective* one.

The training data set used does not contain an equal proportion of *subjective* (68.02%) and *objective tweets* (31.97%). This imbalance seems to overfitting the classifier towards the *subjectivity* class.

A possible solution to increase the precision and recall of the *objective* class, would be to train the classifier with a more balanced data set, in which the proportion of *subjective/objective tweets* will be equally distributed. This possibility was not considered due constraints in the time available to conduct this dissertation, as it would have required a significant time to collect and manually label a new training data set.

Table 4.11: Precision, Recall and F-Measure obtained for Subjectivity Classification (Naive Bayes)

Class	Precision	Recall	F-Measure
Subjective	0.774	0.796	0.785
Objective	0.538	0.507	0.522

Polarity Classifier

When building the polarity classifier, it was discovered that the meta-feature modification introduced in this dissertation (see section 3.2.4), which aimed to capture more sentiment information from adjacent words, was not as suitable for polarity classification as it was for subjectivity detection. In fact, the classifier gave a better performance when the adjacent polarity was not considered. For this reason, it was finally discarded as a feature for this type of classification.

The top-10 relevant features for polarity classification are: *determiners, hashtag, noun, upper case, link, positive polarity, verb, reply, negative polarity* and *strong subjective*. Based on this fact, *prior polarities* have more relevance for polarity detection than for subjectivity classification.

Table 4.12, shows the error rates obtained for each of the algorithms here considered. According to this table, both *SVM* and *MaxEnt* gave lower error rates than *Naive Bayes*. However, when calculating their *precision* and *recall* (Table 4.13), it revealed that both were wrongly classifying almost every *tweet* that contained a *negative polarity* into the *positive* class. Due to this fact, it was decided to use *Naive Bayes* to build

the polarity classifier, as its levels of precision and recall are more balanced for both classes.

The distribution of *tweets* belonging to each class in the training data set is more unbalanced here than it was for the *subjectivity* classifier. In fact, 87.84% of the *tweets* are labelled as *positive*, while only a 12.15% are *negative*. This difference seems to be overfitting the model created towards the *positive* class.

In order to improve the performance obtained, it would be recommended to train the classifier using a new data set that could contain a balanced distribution of *positive* and *negative tweets*. This improvement was not considered here due to constraints in the time available to conduct this dissertation, as it would have required a significant time to collect and manually label a new training data set.

Table 4.12: Comparison of error rates obtained for Polarity Classification

Algorithm	Error Rate
SVM	11.97%
MaxEnt	11.98%
Naive Bayes	29.73%

Table 4.13: Precision, Recall and F-Measure obtained for Polarity Classification

Algorithm Comparison				
Algorithm	Class	Precision	Recall	F-Measure
Naive Bayes	*positive*	0.926	0.72	0.81
	negative	0.219	0.576	0.317
MaxEnt	*positive*	0.882	0.998	0.936
	negative	0.5	0.017	0.034
SVM	*positive*	0.88	1	0.936
	negative	0	0	0

4.3.6 Final considerations about Enhancement 4

The evaluation here described shows how implementing a two-step classifier, for subjectivity detection and polarity evaluation, is a more suitable approach for sentiment analysis in the *Twitter* context, than it is the used of the lexicon approaches implemented in *enhancement 2* and *3*.

In fact, table 4.14 shows that the error rate obtained when processing the *Gold Standard* with the two step-classifier here implemented was 36.7%, which is lower than those achieved in *enhancements 2* and *3*.

Table 4.14: Comparison between *enhancements 2-4* for the sentiment analysis of *#Yes2AV tweets*

Enhancement	Correctly Classified	Error Rate
4	63.21%	36.78%
2	48.67%	51.32%
3 with *SSI Algorithm* and *WordNet++*	46.85%	53.14%
3 with *Adapted Lesk* and *Jiang & Conrath*	45.67%	54.32%

Although there is a significant improvement in the performance achieved with this enhancement, the results obtained are considered low for text classification standards, which normally require precisions and recalls above 0.7. However, it is expected that a new training data set, with a more balanced distribution of each class, will provide a better performance to *Cafetiere's* sentiment analysis capabilities.

It is also important to remark that this enhancement is also affected by the effects of the PoS tagger currently embedded with *Cafetiere*, which assigns incorrect PoS tags to words when twitter syntax elements are present in a *tweet*. In this particular case, this effect alters the feature representation of *tweets*, by producing an incorrect frequency count of the parts of speech contained, thus affecting the final classification.

Due to constraints in the time available to conduct this dissertation, the effects of modifying the PoS tagger to correctly identify the presence of twitter syntax features, was not measured. However, it would be interesting to analyse its effects in the performance of the two-step classifier here implemented.

Finally , as both the polarity and subjectivity classifier are built using a *Naive Bayes* algorithm, the time it takes to train and classify instances is very quick, thus making this enhancement a suitable improvement for *Cafetiere's* sentiment analysis capabilities.

4.3.7 UI Evaluation

The modifications in *Cafetiere's* User Interface were evaluated according to the following principles [88]:

- **Functionality:** each UI modification implemented provides the required functionality to satisfy its:
  - *Suitability:* as it allows the execution of functions that satisfy the list of requirements defined in section 1.3.

  - *Accuracy:* by providing the expected results according to the requirements.
  - *Interoperability:* as it enables an interface for the integration of Twitter Servers with *Cafetiere's* sentiment analysis and text analysis capabilities, as well as with *Cafetiere's* database.
  - *Security:* as it relies on the use of user's credentials to grant access to the database.
  - *Compliance:* as it has been designed and implemented according to MVC pattern for Web Development.

- **Reliability:** the UI modifications implemented are reliable as they satisfy:
  - *Maturity:* the UI has been implemented and verify to avoid failures.
  - *Fault Tolerance:* thanks to the correct decoupling between components, that is achieved with the MVC design pattern, the UI can inform the user about failures that occurred in the server or in the connection, without affecting the functionality of the whole system.
  - *Recoverability:* based on the user interaction, once an error has been produced, the system can successfully recover to its previous state.

- **Usability:** the UI has been designed in a way that it becomes easy for the user to understand, learn, operate and control the system.
  - *Understandability:* the UI has been implemented in the simplest possible way, so it is very easy for the user to understand what is required to do in order to execute *Cafetiere's* functions.
  - *Learnability:* it is possible for the user to learn how to use the UI, as it has been designed to be intuitive enough to allow a self-learning process.
  - *Operability:* due to its simplicity, the user can easily operate with the UI implemented.
  - *Attractiveness:* its attractiveness is sufficient for the purpose of this dissertation, fitting with the other *Cafetiere's* UI components.
  - *Compliance:* the UI modifications have been implemented following appropriate standards, using the set of widgets provided by the *GWT* development environment.

- **Efficiency:** the performance obtained is correct in terms of resources used, thanks to the MVC design pattern implemented.
  - *Time behaviour:* Thanks to AJAX methods, the UI becomes very responsive for any task it executes.
  - *Resource utilization:* the resources needed are accessed using asynchronous HTTP requests between the client and the server, giving a high response to the UI.
  - *Compliance:* it adequates to the MVC and AJAX web development standards.
- **Maintainability:** correcting, improving or adapting the software implemented is possible thanks to the decoupling between components provided by the MVC pattern, as well as the encapsulation of methods and classes that was implemented.
  - *Analysability:* defaults can be diagnosed via a series of test cases, as each functionality provided is properly encapsulated.
  - *Changeability:* thanks to the MVC pattern implemented, modifications can be done without requiring major changes in its current design.
  - *Stability:* methods and classes are correctly encapsulated to avoid side effects when executing other software components.
  - *Testability:* test cases can be defined to test and validate the UI implemented.
  - *Compliance:* it follows the principles of the MVC design pattern, allowing a correct maintainability of the components implemented.
- **Portability:** the software developed for *Cafetiere's* UI modifications can be transferred to other environments that make use of the *GWT* and implement the MVC pattern.
  - *Adaptability:* adaptability between non *GWT* platforms is not possible, due to the pre define widgets that are used.
  - *Installability:* it can be easily installed in any Java Web environment, as the GWT development toolkit allows the creation of *.war* files that can be successfully loaded as a Java Based Web Application.

- *Coexistence:* classes and methods have been correctly encapsulated so the UI can coexists with other independent software.
- *Replaceability:* upgrading the UI is easy, as it is completely decoupled from other software components.
- *Compliance:* the UI has been designed according to the design principles of the MVC pattern and Java Based Web Application developments.

Chapter 5

Conclusions and Future Work

5.1 Dissertation Summary

The project here described aimed to enhance the Information Extraction Engine *Cafetiere* with a set of features that enabled the automatic collection of Twitter feeds, as well as provided the system with more robust capabilities to conduct sentiment analysis over the *tweets* retrieved.

In order to get an understanding of what was required for developing the set of enhancements here proposed, this dissertation firstly analysed the different approaches to allow the automatic collection of *tweets*, followed by a summary of the current state of the art on sentiment analysis research, as well as the challenges involved. In addition, the dissertation described how the recent *social networking phenomenon* generated around Twitter, has helped to bring popularity to this field.

The first feature implemented enabled the automatic collection of Twitter feeds by using the *Twitter Search API*, which allows real time searches of *tweets* by sending customized queries via HTTP requests against Twitter Servers. This approach is suitable for dynamic web environments like *Cafetiere*, where multiple users can simultaneously access the system and send queries against Twitter Servers. These requests are customized and sent directly from the user's web client, which polls the *firehose* until the number of *tweets* requested are finally retrieved. The tweets collected from each request are sent back to *Cafetiere's* server, where are saved within the user's database. Thanks to this approach, the implementation overcomes with the rate limits imposed by the *Search API*, as they are measured against each user's IP address.

This feature was implemented according to the MVC design pattern and AJAX

principles for web development, allowing the decoupling between the components involved and providing a very responsive user experience. *Java* was the programming language chosen for the development of each component, combined with the *GWT* framework for managing asynchronous requests between the client and the server. Finally, this first feature implemented was evaluated via a series of test cases that guaranteed its correct behaviour.

Cafetiere's sentiment analysis capabilities where enhanced with sentiment lexicons and supervised text classification algorithms, which are the two most commonly applied approaches to conduct this type of analysis. For this purpose, a series of pipelines of text analytic processes where developed using the *Apache UIMA* framework. Also, each approach implemented was evaluated using 21,166 *tweets* that were collected during the *AV Referendum*, and had been manually annotated with the polarity orientation they express. This collection of *tweets* constituted the *Gold Standard* data set for this dissertation.

Each *UIMA* pipeline implemented executes a series of NLP techniques before the sentiment analysis step, with the aim of cleaning and structuring the text contained within the *tweets* analysed. These NLP methods mainly divide each *tweet* into elementary text units or tokens, as well as provide additional meta-data, such as their part of speech or the word root form. For this purpose, and to overcome with the limitations imposed by the tokenizer integrated with *Cafetiere*, a new tokenizer was implemented to correctly identify and tokenize Twitter syntax elements (emoticons, twitter topics, twitter mentions or links).

The first sentiment analysis enhancement was developed by defining a UIMA component integrating *SentiWordNet*, a widely used lexicon that is composed by *WordNet* synsets that have been automatically annotated with a sentiment score. This lexicon was stored within *Cafetiere's* database, and it was used to calculate the sentiment orientation of a *tweet*, whether if it is *positive, negative* or *neutral*, based on the average sentiment score obtained from all the senses associated with the word tokens.

This first sentiment analysis approach was evaluated using the *Gold Standard* data set defined for this dissertation. This evaluation revealed the low performance it provides, with an error rate of 51.32% and a tendency to wrongly classify *tweets* towards the positive *class*. The main weakness of this approach is the way in which the sentiment score is obtained from *SentiWordNet*, which is calculated as the average value between the scores obtained from all the senses associated with the words in a *tweet*. As no sense disambiguation is executed, the system is not able to obtain the precise

sentiment score based on the context where the words have been used, thus computing a wrong final sentiment score.

To improve the results previously obtained, two additional UIMA components where implemented with a disambiguation step before the sentiment lookup. For this purpose, two knowledge-based WSD approaches where considered, with the aim of evaluate and compare their performance when applied for sentiment analysis purposes. One approach developed was based on similarity measures, using *Adapted Lesk and Jiang & Conrath*, and the other one made use of structural approaches with the *SSI algorithm*.

This two disambiguation techniques were chosen because they can disambiguate *WordNet* synsets and they have been successfully applied for different disambiguation tasks in many research papers, as well as they are available via *Java* APIs that facilitated their integration with the UIMA components developed.

The performance obtained with the SSI algorithm was slightly better than using similarity measures, mainly because the algorithm makes use of more contextual information to conduct the disambiguation. However, in both cases the error rates obtained were higher than the one achieved when no disambiguation technique was implemented. This results highlighted the constraint imposed by the short length of *tweets*, which do not provide enough context to conduct an accurate disambiguation of word senses.

The final sentiment analysis approach implemented relied on the use of supervised algorithms to build a two step classifier, where firstly the subjective content of *tweets* is detected and then its sentiment polarity is evaluated. *Tweets* were represented using a combination of text syntax features and specific twitter syntax features, that aimed to capture the contextual polarity provided by neighbouring words. Each classifier was implemented using three different algorithms provided by *Weka Java* API, to evaluate and compare they performance. Finally, the classifiers were trained with the collection of *tweets* from the *Gold Standard* here considered.

For both classifiers, the best performance was achieved when using the *Naive Bayes* algorithm. In fact, it was detected that the polarity classifier provided better results when it was built without considering the contextual sentiment from neighbouring words, and it was finally built with a slightly modified set of features. Nevertheless, the final sentiment classification obtained with this method outperformed any of the approaches previously considered.

Despite of obtaining a significant improvement when using supervised classification methods, the results obtained are still low for traditional text classification standards, as the models generated were overfitted towards the subjective class, in the subjectivity classifier, and towards the positive class, in the polarity classifier. In each case, the overfitting corresponds to those classes from the training data set with the highest frequency of samples. This results revealed the low quality of the training data set that was available to conduct this dissertation, which did not contain an equal distribution of *subjective/objective tweets*, as well as of *positive/negative tweets*. As a result, this imbalance degraded the quality of the classifiers implemented. However, it is expected to obtain a better performance if the quality of the training data set is improved.

Overall, this dissertation enhanced the information extraction engine *Cafetiere* with a set of features that improved its sentiment analysis capabilities over Twitter feeds, as well as demonstrated that, within the Twitter context, a two step supervised classification method is a more robust approach than the use of sentiment lexicons, which are constraint by the lack of context carried by the short number of words composing *tweets*.

5.2 System Limitations

The current version of the database integrated with *Cafetiere* does not allow concurrency, so it can only be accessed by one thread at a time. This limitation slows the performance obtained, as none of the sentiment analysis approaches here implemented can be currently parallelized. The database then needs to be upgraded to a better version to remove this limitation.

The total number of past *tweets* that can be retrieved is limited to the restrictions imposed by Twitter's server, as the *Search API* does not allow the collection of *tweets* that are more than 5 days old. For this reason, this limitation should be considered if the sentiment analysis needs to be measured over large periods of time.

An important point to mention is that the sentiment analysis approaches implemented for *enhancements 2* and *3*, which relied in the use of sentiment lexicons, calculate the sentiment orientation of a *tweet* based on the information contained within single words, without considering the contextual sentiment expressed by neighbouring terms. This clearly constraints the performance obtained. The analysis of negation forms, the sentiment carried by adjacent words, as well as the presence of other indicators of sentiment, such as *emoticons*, upper cases and exclamation marks, have only

been considered for *enhancement 4*. Due to constraints in the time available to conduct this dissertation, *enhancements 2* and *3* could not be improved with a wider analysis that could have taken into account more contextual information. For this reason conclusions about their effect could not be estimated.

It was further observed that the PoS tagger currently integrated with *Cafetiere* is not optimize to identify the presence of *Twitter syntax* elements. As a result, the tagger is processing those elements as normal words in a sentence, assigning them a PoS tag that does not represent their functionality. The presence of this wrongly classified entities sometimes affects how other words in a *tweet* are tagged, thus affecting the final sentiment score obtained from *SentiWordNet* in *enhancements 2* and *3*, producing a wrong disambiguation in *enhancement 3* and generating the wrong set of features in *enhancement 4*. Due to constraints in the time available to conduct this dissertation, a modified version of the PoS tagger that could take into consideration the presence of *Twitter syntax* elements was not implemented, and the conclusions about its effect could not be estimated.

Finally, the results obtained for the supervised classification methods implemented in *enhancement 4*, have shown a tendency to wrongly classify tweets towards those classes with a higher distribution in the training data set. Due to time constraints, a cost sensitive learning, that could penalize a wrong classification of the less frequent classes, was not applied and the conclusions about its effect could not be estimated.

5.3 Future Work

Several enhancements could be developed in order to improve the current system limitations. The sentiment analysis conducted with *SentiWordNet* should ideally be expanded to consider more contextual information from adjacent words, as well as extend the lexicon with domain specific terms. In addition, better disambiguation techniques could be applied, in order to overcome with the context constraints imposed by the short number of words contained within a *tweet*.

The PoS tagger should also be modified to correctly identify the presence of twitter syntax elements, as well as analyse its effect over the final sentiment classification obtained for each approach here implemented.

Based on the better performance provided by supervised sentiment classification methods, future work should also be done to improve their accuracy, by creating a more robust and balanced training data set, as well as measuring the effects of using

different set of features. In addition, *Cafetiere* UI could be modified to allow users to provide feedback about the classification conducted. This information could be use to train the classifiers previously developed, in order to improve their performance for sentiment detection.

Sarcasm and irony can affect the final sentiment analysis computed, as their presence changes the polarity of the idea expressed to its opposite. *Cafetiere* should be enhanced with capabilities that could take into consideration the presence of sarcastic and ironic comments. In addition, the detection of specific moods or feeling could be also implemented.

The enhancements developed for this dissertation have been designed to allow its further parallelization, once the database will be upgraded to a version supporting concurrency. Future work should be done in this direction and improve *Cafetiere's* performance when multiple users are using its sentiment analysis capabilities.

Finally the use of the *Streaming API* should be also considered when measuring the evolution of sentiment over large periods of time. This modification will require a deep analysis on how to combine persistent connections with having multiple users that are simultaneously accessing the system, as well as study the way to store and maintain the huge number of *tweets* that are retrieved when using this API.

Bibliography

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval. Now Publishers Inc*, vol. 2, no. 1-2, pp. 1–135, 2008.

[2] J. Wiebe, "Tracking point of view in narrative," *Computational Linguistics. MIT Press Journals*, vol. 20, no. 2, pp. 233–287, 1994.

[3] J. Wiebe, R. Bruce, and T. O'Hara, "Development and use of a gold-standard data set for subjectivity classifications," in *Proc. of the 37th Annual Meeting of The Association for Computational Linguistics on Computational Linguistics*, doi: 10.3115/1034678.1034721, 1999.

[4] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proc. of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, pp. 79–86, 2002.

[5] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classication of product reviews," in *Proc. of the 12th International Conference on World Wide Web*, pp. 519–528, 2003.

[6] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proc. of the 42nd Annual Meeting on Association for Computational Linguistics*, pp. 271–278, 2004.

[7] T. Wilson, J. Wiebe, and P. Hoffman, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proc. of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 347–354, 2005.

[8] R. Mihalcea, C. Banea, and J. Wiebe, "Learning multilingual subjective language

via cross-lingual projections," in *Proc. of the Association for Computational Linguistics*, p. 976983, 2007.

[9] P. Turney, "Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews," in *Proc. of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 417–424, 2002.

[10] S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G. Reis, and J. Reynar, "Building a sentiment summarizer for local service reviews," in *WWW Workshop on Natural Language Processing Challenges in the Information Explosion Era*, 2008.

[11] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,*, pp. 168–177, 2004.

[12] J. Bollen and M. Huina, "Twitter mood as a stock market predictor," *IEEE Computer*, vol. 44, no. 10, pp. 91–94, 2011.

[13] A. Nicholas and D. Shamma, "Characterizing debate performance via aggregated twitter sentiment," in *Proc. the 28th International Conference on Human Factors in Computing Systems*, pp. 1195–1198, 2010.

[14] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith, "From tweets to polls: Linking text sentiment to public opinion time series," in *Proc. of the International AAAI Conference on Weblogs and Social Media*, pp. 122–129, 2010.

[15] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment," in *4th International AAAI Conference on Weblogs and Social Media*, 2010.

[16] J. B. William, J. McNaught, A. Vasilakopoulos, K. Zervanou, B. Theodoulidis, and F. Rinaldi, "Cafetiere: Conceptual annotations for facts, events, terms, individual entities and relations," *Parmenides Technical Report*, no. TR-U4.3.1, 2005.

[17] L. Barbosa and J. Feng, "Robust sentiment detection on twitter from biased and noisy data," in *Proc. of the 23rd International Conference on Computational Linguistics*, pp. 36–34, 2010.

[18] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3 .0: An enhanced lexical resource for sentiment analysis and opinion mining," in *Proc. of the 7th Conference on International Language Resources and Evaluation*, pp. 2200–2204, 2008.

[19] D. Davidov, O. Tsur, and A. Rappoport, "Enhanced sentiment learning using twitter hashtags and smileys," in *Proc. of the 23rd International Conference on Computational Linguistics*, 2010.

[20] H. Yu and V. Hatzivassiloglou, "Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences," in *Proc. of Empirical Methods for Natural Language Processing*, doi:10.3115/1119355.1119372, 2003.

[21] S. Kim and E. Hovy, "Identifying and analysing judgment opinions," in *Proc. of Human Language Technology and the North Ametican Chapter of the Association for Computational Linguistics*, doi:10.3115/1220835.1220861, 2006.

[22] H. Takamura, T. Inui, and M. Okumura, "Latent variable models for semantic orientations of phrases," in *Proc. of The European Chapter for Computational Linguistics*, pp. 201–208, 2006.

[23] A. Andreevskaia and S. Bergler, "Mining wordnet for fuzzy sentiment: Sentiment tag extraction from wordnet glosses," in *Proc. of the European Chapter of the Association for Computational Linguistics*, pp. 209–216, 2006.

[24] A. Esuli and F. Sebastiani, "Determining term subjectivity and term orientation for opinion mining.," in *Proc. of the European Chapter of the Association of Computational Linguistics*, pp. 193–200, 2006.

[25] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. of the 43th Annual Meeting of the Association for Computational Linguistics*, pp. 115–124, 2005.

[26] C. O. Alm, D. Roth, and R. Sproat, "Emotions from text: Machine learning for text-based emotion prediction," in *Proc. of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, pp. 579–586, 2005.

[27] R. Mihalcea and C. Strapparava, "Learning to laugh (automatically): Computational models for humor recognition," *Journal of Computational Intelligence*, 2006.

[28] G. Mishne and M. de Rijke, "Capturing global mood levels using blog posts," in *AAAI Symposium on Computational Approaches to Analysing Weblogs*, pp. 145–152, 2006.

[29] C. D. Manning, P. Raghavan, and H. Schtze, *An Introduction to Information Retrieval*, ch. 2.2 Determining the vocabulary of terms, pp. 22–32. Cambridge University Press, 2009.

[30] *Oxford Dictionary*. Oxford University Press, 2008.

[31] M. Porter, "An algorithm for suffix stripping," *Program 14*, pp. 130–137, 1980.

[32] S. Kim and E. Hovy, "Determining the sentiment of opinions," in *Proc. of the 20th international conference on Computational Linguistics*, pp. 1367–1373, 2004.

[33] P. J. Stone, D. C. Dunphy, M. S. Smith, and D. M. Ogilvie, *The General Inquirer: A Computer Approach to Content Analysis*, vol. 8. MIT Press, 1966.

[34] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," *Language Resources and Evaluation*, vol. 39, no. 2-3, pp. 165–210, 2005.

[35] P. Stone, "General inquirer home page." `http://http://www.wjh.harvard.edu/~inquirer/`, 2002.

[36] H. D. Lasswell and N. J. Z., *The Lasswell Value Dictionary*, vol. 1-3. New Heaven: Yale University, 1968.

[37] P. C. Tetlock, "Giving content to investor sentiment: The role of media in the stock market," *The Journal of Finance*, vol. 62, no. 3, pp. 1139–1168, 2007.

[38] P. C. Tetlock, M. Saar-Tsechansky, and M. Macskassy, "More than words: Quantifying language to measure firms' fundamentals," *The Journal of Finance*, vol. 63, no. 3, pp. 1437–1467, 2008.

[39] E. Riloff and J. Wiebe, "Learning extraction patterns for subjective expressions," in *Proc. of the 2003 conference on Empirical Methods in Natural Language Processing*, pp. 105–112, 2003.

[40] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[41] R. Navigli, "Word sense disambiguation: A survey," *ACM Computing Surveys*, vol. 41, no. 2, pp. 1–69, 2009.

[42] K. Denecke, "Are sentiwordnet scores suited for multi-domain sentiment classification?," in *4th International Conference on Digital Information Management*, pp. 1–6, 2009.

[43] T. Thura, J. Na, C. Khoo, and S. Shakthikumar, "Sentiment anlysis of movie reviews on discussion boards using a linguistic approach," in *Proc. of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion*, pp. 81–84, 2009.

[44] C. Potts, "Sentiment symposium tutorial. stanford linguistics." `http://sentiment.christopherpotts.net/lexicons.html`, 2011.

[45] P. M. Roget, *Rogets International Thesaurus*. Cromwell, 1st ed., 1911.

[46] P. Hanks, *Collins English Dictionary*. Harper Collins, 2002.

[47] C. Soanes and A. Stevenson, *Oxford Dictionary of English*. Oxford University Press, 2003.

[48] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal Human-Computer Studies*, no. 5-6, 1995.

[49] J. Clear, "The british national corpus," *The Digital Word: Text-Based Computing in the Humanities*, pp. 163–187, 1993.

[50] E. Charniak, D. Blaheta, N. GE, K. Hall, J. Hale, and M. Johnson, *BLLIP 1987-89 WSJ Corpus Release 1*. Linguistic Data Consortium, 2000.

[51] G. A. Miller, C. Leacock, R. Tengi, and R. T. Bunker, "A semantic concordance," in *Proc. of the ARPA Workshop on Human Language Technology*, pp. 303–308, 1993.

[52] T. Chklovski and R. Mihalcea, "Building a sense tagged corpus with open mind word expert," in *Proc. of ACL 2002 Workshop on WSD: Recent Successes and Future Directions*, pp. 116–222, 2002.

[53] T. Brants and A. Franz, *Web 1t 5-gram, ver. 1, ldc2006t13*. Linguistic Data Consortium, 2006.

[54] B. Magnini and G. Cavagli'A, "Integrating subject field codes into wordnet," in *Proc.of the 2nd Conference on Language Resources and Evaluation*, pp. 1413–1418, 2000.

[55] D. Jurafsky and J. Martin, *Speech and Language Processing*. Prentice Hall, 2000.

[56] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone.," in *Proc. of the 5th International Conference on Design of Communication*, pp. 24–26, 1986.

[57] S. Banerjee and T. Pedersen, "Extended gloss overlaps as a measure of semantic relatedness.," in *Proc. of the 18th International Joint Conference on Artificial Intelligence*, pp. 805–810, 2003.

[58] S. Patwardhan and T. Banerjee, S.and Pedersen, "Using measures of semantic relatedness for word sense disambiguation," in *Proc. of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 241–257, 2003.

[59] A. Budanitsky and G. Hirst, "Evaluating wordnet-based measures of lexical semantic relatedness.," *Computational Linguistics.*, vol. 32, pp. 13–47, 2006.

[60] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE Trans. Syst. Man Cybernet*, vol. 19, pp. 17–30, 1989.

[61] C. Leacock and M. Chodorow, "Combining local context and wordnet similarity for word sense identification," in *WordNet: An electronic Lexical Database*, pp. 265–283, MIT Press, 1998.

[62] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy.," in *Proc. of the 14th International Joint Conference on Artificial Intelligence*, pp. 448–453, 1995.

[63] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," in *Proc. of the 10th International Conference on Research in Computational Linguistics*, pp. 19–33, 1997.

[64] R. Mihalcea, P. Tarau, and E. Figa, "Pagerank on semantic networks, with application to word sense disambiguation," in *Proc. of the 20th International Conference on Computational Linguistics*, pp. 1126–1132, 2004.

[65] S. Brin and M. Page, "Anatomy of a large-scale hypertextual web search engine," in *Proc. of the 7th Conference on World Wide Web*, pp. 107–117, 1998.

[66] R. Navigli and P. Velardi, "Structural semantic interconnections: A knowledge-based approach to word sense disambiguation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1075–1086, 2005.

[67] H. T. Ng and H. B. Lee, "Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach," in *Proc. of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 40–47, 1996.

[68] R. Navigli and S. P. Ponzetto, "Multilingual wsd with just a few lines of code: The babelnet api," in *Proc. of the 50th Annual Meeting of the Association for Computational Linguistics*, pp. 8–14, 2012.

[69] R. Navigli and S. P. Ponzetto, "Babelnet: Building a very large multilingual semantic network," in *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 216–225, 2010.

[70] S. P. Ponzetto and R. Navigli, "Knowledge-rich word sense disambiguation rivaling supervised systems," in *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1522–1531, 2010.

[71] C. D. Manning, P. Raghavan, and H. Schtze, *An Introduction to Information Retrieval*. Cambridge University Press, 2009.

[72] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *Proc. of the AAAI-98 Workshop on Learning for Text Categorization*, pp. 41–48, 1998.

[73] K. Frantzi, S. Ananiadou, and H. Mima, "Automatic recognition of multi-word terms," *International Journal of Digital Libraries*, vol. 3, no. 2, pp. 117–132, 2000.

[74] V. Hatzivassiloglou and J. Wiebe, "Effects on adjective orientation and gradability on sentence subjectivity," in *Proc. of the 18th Conference on Computational Linguistics*, pp. 299–305, 2000.

[75] S. Das and M. Chen, "Yahoo! for amazon: Extracting market sentiment from stock message board," in *Proc. of the 8th Asia Pacific Finance Association Annual Conference*, 2001.

[76] D. Jurafsky and C. Manning, "Stanford NLP course. Lecture:." `https://www.coursera.org/course/nlp`. [Online; accessed 12-May-2012].

[77] T. Joachmis, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. of the 10th European Conference on Machine Learning*, pp. 137–142, 1998.

[78] T. Blog, "Twitter turns six." `http://blog.twitter.com/2012/03/twitter-turns-six.html`, 2012.

[79] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?," in *Proc. of the 19th international conference on World Wide Web*, pp. 591–600, 2010.

[80] B. Carter, "'arab spring' on the hudson: Social media's the same the world over." `http://www.wired.com/epicenter/2011/10/arab-spring-on-the-hudson/all/1`, 2011.

[81] R. Gonzalez-Ibaez, S. Muresan, and N. Wacholder, "Identifying sarcasm in twitter: A closer look.," in *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 581–586, 2011.

[82] D. Davidov, O. Tsur, and A. Rappoport, "Semi-supervised recognition of sarcastic sentences in twitter and amazon," in *Proc. of the 14th Conference on Computational Natural Language Learning*, pp. 107–116, 2010.

[83] Twitter, "Twitter developers." `https://dev.twitter.com/`, 2012. [Online; accessed 12-August-2012].

[84] L. Richardson and S. Ruby, *RESTful Web Services. Web Services for the real world*. O'Reilly Media, 2007.

[85] J. J. Garrett, "Ajax: A new approach to web applications." `http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications`. [Online; accessed 11-August-2012].

[86] S. Burbeck, "Applications programming in smalltalk-80: How to use model-view-controller." `http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html`. [Online; accessed 11-August-2012].

[87] K. Beck and et., "The agile manifesto." `http://http://agilemanifesto.org/`. [Online; accessed 11-August-2012].

[88] M. King, "General principles of user-oriented evaluation," *Evaluation of Text and Speech Systems*, vol. 37, pp. 125–161, 2007.

[89] P. Han, B. Cook and T. Baldwin, "Automatically constructing a normalisation dictionary for microblogs," in *Proc. of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 421–432, 2012.

[90] ChatSlang.com, "Social media slang." `http://www.chatslang.com/terms/social_media/`, 2012. [Online; accessed 19-August-2012].

[91] P. Resnik and D. Mona, "Measuring verb similarity," in *Proc. of the 22nd Annual Meeting of the Cognitive Science Society*, pp. 399–404, 2000.

[92] H. David, "Java wordnet::similarity." `http://www.sussex.ac.uk/Users/drh21/`, 2012. [Online; accessed 21-August-2012].

[93] A. C. McCallum, "Mallet: A machine learning for language toolkit." `http://mallet.cs.umass.edu`, 2002. [Online; accessed 23-August-2012].

[94] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, 2009.

[95] I. H. Witten, F. Eibe, and M. A. Hall, *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[96] S. M. Kim and E. Hovy, "Automatic detection of opinion bearing words and sentences," in *Proc. of the International Joint Conference on Natural Language Processing*, pp. 61–66, 2005.