Gauging Affectivity in Social Networks

By
Konstantinos Matthaios Brilakis

Supervisor

Dr. I. Paraskakis

Abstract

Numerous platforms throughout the Web offer the ability and promote individuals to express their opinions and other subjective content for various reasons such as product reviews and social network posts. User generated content increase has greatly boosted this phenomenon. And it has significant implications in the market concerning businesses; and interested entities are starting to monitor social network activity to identify preferences and indications of behaviour. Thus, this paper investigates the field of Sentiment Analysis that is concerned with the review of opinionated material and identification of sentiment (positive, negative) in natural language. Initially, a thorough background on this natural language processing application is conducted, examining challenges, the technical foundations, and contemporary techniques, tools, and technologies used today. Subsequently, this paper proposes and describes a machine learning solution for a sentiment analysis engine using known supervised learning models, that is, support vector machine implementations in the GATE environment, a text engineering toolkit. It experiments with linear classifiers and n-gram models to develop a model to analyse Twitter posts and social network content in general, and film reviews as well. Finally, we present a complete software application (Sent-Affect) implemented in Java that integrates the sentiment analysis learning model for user interaction.

"All sentences or passages quoted in this dissertation from other people's work have been specifically acknowledgement by clear cross-referencing to author, work and pages(s).I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this dissertation and the degree examination as a whole.

Name: Konstantinos Matthaios Brilakis

Signed:

Date: 08/06/2015"

Table of Contents

List of Figures

List of Tables

List of Listings

1. Introduction

The analysis and understanding of opinions and sentiments in social networks and generally on the web is an area of study that has drawn much attention in the past years. The advent of Web 2.0 technologies has led to an increased amount of user generated content on the web, particularly in platforms such as blogs, forums, and social networks, in which people express their opinions about products or give various reviews and other subjective comments towards any type of service or product [1], [2], [3]. Hence, different organisations, companies, researchers, and other interested parties have readily available an immense amount of relevant and significant raw data. Opinions are important as they often indicate and may influence behaviour. Therefore, considering the mere size of available information, manual extraction and classification of opinions is quite limited and inefficient, thus, the process needs to be automated in order to process large amounts of data [1], [3], [4].

For this reason, the field of *Sentiment Analysis* is concerned with the analysis of opinions and it is a highly contemporary natural language processing task. This versatile problem has been dealt with lexical and rule based approaches and machine learning techniques, such as supervised learning, ergo, taking advantage of the data availability on the Internet to train models that recognise sentiments and are able to separate positive from negative text [1], [2]. This project is involved with the development of a sentiment analysis engine through supervised learning models that performs analysis of sentiments at the document and sentence level of text sources. The system realised is called Sent-Affect and it is perceived as a complete sentiment analysis application that handles confidently data from Twitter, Facebook and similar platforms.

Moreover, in combination with the increased capabilities of data mining, the understanding of opinionated content can be enhanced and focused on significantly by using advanced algorithms and linguistic analysis and features. Additionally, the important aspect of the mentioned issue relates to the idea of collecting feedback from people. Trends such as, mobile computing have provided people with devices (smartphones, tablets) and platforms, in which they can conveniently express opinions, especially in social networks like Facebook, Twitter, Flickr and discussion forums [1], [2]. Consequently, there is an interest in discovering what exactly people are saying in the millions of Twitter posts. And it is a challenging task indeed due to

obstacles, being; sarcasm, spelling mistakes, use of acronyms and abbreviations, and fake content.

Undoubtedly, one of the main driving forces of this trend of seeking and expressing opinions online is the consumption of products and services. This is because the specific domain is highly subjective and people value real testimonials and experiences. According to [1], users of the Internet constantly search for opinions and they also report that a staggering 81% of Internet users have conducted search on sentiments regarding a product at least one time. This excessive interest in opinionated content can wield a major influence; therefore, shop owners, vendors, and large businesses are focusing their attention on the voices of consumers. As a result, it has been acknowledged that among tweets, Facebook, and Flickr posts dwells vital information that needs to be processed, as otherwise would be wasted.

1.1 Aim and Objectives

This project aims to develop a Sentiment Analysis engine that will be able to classify opinionated content regarding the sentiment's that is expressed (positive, negative). Further, a software system is to be implemented to support and demonstrate the sentiment analysis module.

Therefore, the main objectives of this final year project are the following:

1. Conduct an extensive background research on the field of sentiment analysis and present main topics, issues, and methodologies in a review of literature.

2. Identify and evaluate existing tools and available technologies for performing sentiment analysis.

3. Develop a sentiment analysis engine that will handle the classification of opinions and sentiments to positive, negative, and neutral, in text source inputs.

4. Implement an interface platform to supplement and demonstrate the analysis engine for user interaction.

1.2 Report Overview

This paper is structured in the following manner:

- **Chapter 1**: an introduction that presents the motivation and demand for sentiment analysis and sets the initial aims and objectives of the project.

- **Chapters 2**: include a background research on sentiment analysis covering key definitions, topics, challenges and issues, applications, and techniques used in this field. Additionally, tools and technologies are presented and discussed.

- **Chapter 3**: analysis, which includes a discussion of the requirements of the system, an outline of the expected functionalities to be implemented and a high level presentation of the system's primary components. Also, the project planning and general process is discussed.

- **Chapter 4**: presents the construction of the machine learning models that have been developed to handle the sentiment classification. It explains the data collection process, the annotation phase, and the classifier training and evaluation. Secondly, the software system's design and implementation is discussed as well.

- **Chapter 5**: comprises a discussion chapter that analyses in more depth the machine learning development process. Such a process requires intensive experimentation and various conclusions were inferred during the project.

- **Chapter 6**: the conclusions chapter summarises key points and the project as a whole and at the same time present potential aspirations regarding future work and development.

2. Sentiment Analysis Background

This chapter conducts a review of the literature on the field of sentiment analysis and opinion mining. It discusses in depth several aspects of this multifaceted problem starting from a definition and background information. It discusses how sentiment analysis, a natural language processing application has various challenges like sarcasm, short and ambiguous text, negation, and spamming. The coverage is extended to present the main levels of analysis (word, document, and sentence) and the underlying foundations of the field that are used to deal with this problem. The primary aspects covered include machine learning (supervised, unsupervised models), classifier analysis (naïve Bayes, Support Vector Machines) and also the dictionary approach with lexicons and pre-labelled words. Moreover, the various applications of SA in the market, closing with a summarisation and conclusion that explains the factors and techniques adopted for the completion of this project.

2.1 Preamble

Sentiment analysis (SA) or opinion mining is a category and task of natural language processing (NLP) and text analysis. This field concentrates on the extraction and analysis of emotions, sentiments, and subjective opinions of a particular dataset (text resource) towards various entities [1], [2], [3], and [4]. Sentiment analysis deals with nonfactual data, i.e., information that expresses a sentiment rather than present raw facts that can be proven or disproven. The purpose of this type of text analysis is concerned with providing classification of subjective comments in their most fundamental forms; positive, negative, and neutral [2], [4], and [5]. Moreover, classifications of a higher level, such as emotional states (e.g. happiness, frustration) also exist.

Natural Language Processing (NLP) is applied to readable text data, or to data presented in a mark-up language. A mark-up language (e.g. XML) is used to describe certain attributes of the data; they act as meta-information and they are called annotated or labelled data. As explained later in the discussion of techniques for SA, numerous of them, which are used for sentiment analysis, use an annotated corpus of information to apply *Supervised Learning* methods [4].

According to [2], both the terms sentiment analysis and opinion mining first appeared in 2003; however, research concerning opinions and sentiments had begun earlier. Research in opinion mining is quite contemporary at the moment and it is still under development and experimentation. Nowadays, the terms opinion mining and sentiment analysis are used interchangeably [2], [3] and in this paper they will also be used as synonyms. It is vital to point that sentiment analysis is a relatively new area of study, but nevertheless, it has an extensive research area presently due to numerous applications it can facilitate; particularly in social media and social networks and user focused content (e.g. blogs, forums, twitter, and in other social contexts) [1], [2], [4], and [6]. In the mentioned areas where this field has applications, sentiment analysis targets user generated content; for example, a product review, blog and forum posts, tweets (Twitter), and posts in any platform that expresses an opinion towards some entity [1]. An entity may be a specific product (e.g. a smartphone device), a service, organisations, and even people [2], [3].

An equally significant aspect of sentiment analysis is related to the type of context to which it is applied. In order to effectively classify opinions and derive accurate results, it is imperative to determine to what the opinions stated refer to, namely, what is the target object/entity or the general domain of discussion. Context is especially important when one is determined to undertake the problem of sentiment analysis using machine learning methods. This can be explained by the fact that in numerous domains, the vocabulary used may have other or opposite meanings in different contexts. For instance, consider a domain that has specific notation or wording (e.g. Internet slang and abbreviations in websites like Twitter). For this reason, engines created to conduct sentiment analysis are usually built to target specific domains to increase the correctness of the analysis and thus provide more accurate results. As a consequence, generic methodologies, engines, and algorithms may perform poorly on a specific realm and require specialization [2], [7].

2.2 Opinion Definition

The most important concept in sentiment analysis is the opinion itself. An opinion needs to be interpreted, and for this, a coherent definition is required. Thus, in order to

be precise and unambiguous, an opinion, as reported by [4], can be defined in natural language in the following way:

"*An opinion denotes a personal, subjective belief, judgment, appraisal, or sentiment towards an entity, an idea or another opinion that cannot be proven by facts. It thus constitutes the counterpart to factual information*".

The above definition seems crystal clear; however, a more formal definition is required to define an opinion. Liu [2], [8] defines an opinion as a quintuple.

$$opinion(e_i, a_{ij}, s_{ijkl}, h_k, t_l) \qquad \textbf{(2.1)}$$

Where *e* is the target entity, *a* is the aspect or feature of *e*, *s* is the sentiment expressed on aspect *a*, *h* is the one that expressed the specific opinion on time *t*, i.e., when the opinion was stated in date and time.

Further, the above terms of the 5-tuple, the pentad of *Opinion* (expression 1.1) will be explained in more detail.

- **Entity**: an entity, or sometimes called object, is the target of the opinion giver. Entities can have aspects or features (e.g. phone X is the entity, the battery of that phone under discussion is the aspect). As mentioned earlier, an entity can be a product, service, topic, a person or a business entity (e.g. a brand like Nike) [2], [3].

- **Aspect**: an aspect or feature is a component, an attribute of the entity that sentiments are expressed towards (e.g. a laptop can be the entity and its battery life an aspect). Aspect analysis is quite difficult and contemporary nowadays, due to the fact that a single opinion tends to be divided among attributes of entities and hence this information needs to be saved [2], [3].

- **Sentiment**: the sentiment on the aspect that can be positive, negative, and neutral or defined with a standardised scale of strength e.g. one to five stars in product reviews. The sentiment may also refer to the entity as a whole, in this case, the aspect and entity become the same [2], [3].

- **Opinion holder**: the holder is a legal or natural person that states the opinion [2], [3].

In the above definition of an opinion, the subscripts of the members are used to underline that the members of the pentad must correspond to one another. In other words, the sentiment s_{ijkl} is provided by the holder h_k about the feature/aspect a_j of entity e_i at time t_l. In a situation where they do no match, there exists an error [2].

Furthermore, the defined pentad is effective in terms of covering potential situations of various meanings of a sentiment being expressed, but there are cases where this definition cannot account for; for instance, when opinions have contexts: e.g. "*This bike is too high for a short person.*". This opinion on the particular bike does not concern everyone. According to it, only a short person is expected to be troubled and thus, this cannot be treated as a plain negative sentiment for that bike. Moreover, the definition may sometimes lead to data loss; for instance, "*The oil for this car is very expensive*". In this case, the sentence per se does not mean that the car is expensive, so the aspect of 'price' of the entity car is not in question, rather a negative view of the oil is observed. As a result, the information regarding the sentiment towards the oil is lost, unless one wishes to analyse aspects and features of the oil. Thence, the oil must be treated as a whole new entity; otherwise such data may be missed [2].

The above representation of an opinion has undeniably limitations, notwithstanding, it is acceptable and satisfactory for most applications as it preserves the indispensable information needed for an opinion in a rather simplified and easy to understand form. While expanding the definition could, conceivably, solve some of the above issues, the increased complexity of the definition may lead to issues in the problem solving process and constitute the use of it cumbersome [2].

2.3 Challenges in Sentiment Analysis

The polarity classification of opinions is quite a challenging task, considering the fact that it is formidable even for a human to attempt to interpret opinions. Some of the considerable issues and obstacles in sentiment analysis are: sarcasm, various linguistic factors, object/entity recognition, and fake content/opinion spam, [1], [2], [11], [12], [3]. These difficulties mentioned above will be further discussed in this section.

2.3.1 Sarcasm

Regarding the issue of sarcasm, the problems quickly became evident. The sentiment analysis must be performed by a 'trained' computer program, conduct the text classification, extract sentiments and produce valid, up to a point, results. Sarcasm is an obstacle due to the fact that it is common to use words of one type of sentiment, but the meaning of the whole segment implies the opposite sentiment, namely, usage of sarcasm [1], [3], [12]. The hindrance lingers in the ambiguity of natural language and sarcasm. Also, it is rather burdensome and, candidly, impossible to construct a set of predefined rules to filter opinions because the problem of sentiment extraction cannot be discretized. Furthermore, even humans cannot completely produce certain results in peculiar cases where extreme or specific sarcasm is used. In fact, a recent a case study where two persons were asked to classify text, has shown unanimous agreement in only 82% of the test examples [47]. Specific sarcasm is referred to words or sentences that are derived from particular domains that some people may not understand or jokes and idioms that have specific meanings to specific people.

The problem of sarcasm shall be demonstrated through a routine example. Consider the following review for restaurant X:

"*This restaurant was great. Now, all future meals shall taste fantastic in comparison*."

The first sentence definitely implies a positive experience; however, upon reading the second sentence, the sarcasm emerges, as the reviewer implies that the food was so awful that their future meals will be great compared to the restaurant food. By the same token, sentence two can also be characterised as positive, if one isolates the words "*taste fantastic*". The latter word, again, is a positive sentiment word. Hence, dealing with such situations is a gargantuan problem in sentiment analysis.

2.3.2 Linguistic Problems

Linguistic issues are also extremely common in extracting opinions. In this category, there are various problems, thus, some will be mentioned briefly. A major concern is poor spelling and grammar. These situations may cause issues in word identification, and thus lead to inconclusive or misinterpreted results. Moreover, sentences that do not provide necessarily a sentiment; interrogative sentences (questions) need consideration. For example, "*Do you guys know if the Xphone is good?*" In this example, positive sentiment words appear, but the sentence per se does not present a positive or negative opinion towards the particular product. On the other hand, not all questions are sentiment-less [2], [11]. Consider the following sentence: "*Does anyone know how to activate the Bluetooth in this lousy laptop?*" In this sentence the author calls the object laptop lousy, which can be defined as a negative sentiment towards the laptop.

2.3.2.1 Comparative Sentences

Furthermore, comparative opinions, where an author is comparing two entities is a difficulty. It is analogous with direct opinions, but the meaning may be distinct and thus have to be dealt with in another way than typical opinions.

A typical comparative opinion has the following generic form: "*The camera of phone X is better that the camera of phone Y*". These types of sentences express a relation between one or more entities pertaining to similarities or differences of the participating entities. Comparisons are shaped by using the comparative or superlative forms of adjectives and adverbs. The formal definition of an opinion as a quintuple in

section 1.3 does not cover comparative sentences and has to be treated and defined differently. Therefore, a comparative sentence can be defined formally in the following way:

$$(O_1, O_2, F, po, h, t) \qquad \textbf{(2.2)}$$

Where O_1 and O_2 are the sets of the entities or objects that are the subject of comparison on their features F they have in common, po is the set in which the preferred entity of h the holder of the opinion. The time when this sentence took place is t [1], [2], and [8].

2.3.2.2 Use of Negation

Negation is another important factor in extracting sentiments of a particular dataset. Negation of words cannot be overlooked because they revert the sentiment of words or even whole sentences. A most fundamental example that illustrates the inversion of a sentiment using negation: "*I like this phone*" and "*I don't like this phone*". The use of the word "not" changes the sentiment from positive to negative; in reality, however, the sentences are not so simple and straightforward. Particularly the use of the word "not" does always invert the meaning of a sentence. For instance, consider the following sentence: "*Not only does this laptop have a very low battery life, but only it is immensely expensive*". This example reveals that the use of "not" will not revert the sentiment of a sentence, which is this case remains negative. Moreover, double negatives have to be considered as well e.g. "*This printer is not that bad after* all". The previous example is a difficult case to interpret because negative only sentiment words appear in the text. Consequently, as shown by the examples, modelling and dealing negation is quite challenging in sentiment analysis, but a vital one, unquestionably, due to its frequent usage in natural language and flexibility [1], [2], and [13].

2.3.2.3 Conditional Sentences

Conditional sentences require special attention too due to the fact that there are many types of such sentences and more importantly, it proves that the sentiment words on their own cannot be used to extract sentiments and separate opinionated from non-opinionated sentences. Sentiment words refer to the set of words that express a sentiment (e.g., good, great, terrible). The following examples demonstrate the different factors and considerations in these types of sentences [1], [2], [14].

Example 1: "*If the phone is nice and has a good camera, I will definitely purchase it*". This is a conditional sentence that contains no opinion whatsoever, even though two positive sentiment words can be found i.e. nice and good. It merely implies that depending on the quality of the camera, the person shall purchase it.

Example 2: "*If you are not happy with your X laptop, you should consider buying this awesome and durable Z laptop*". In this situation, there is no direct opinion on X laptop; however, it is clear that there is a positive sentiment towards Z laptop. This is a delicate sentence and the sentiment towards laptop X may be negative from the owner, the use of "if" does not establish this as a fact. Additionally, by removing the "if" the first part of the sentence, it is evidently negative towards laptop X.

Example 3: "*If you guys want a smartphone with a really good camera, don't buy this phone X*". The previous sentence expresses a negative sentiment for phone X although the sentence is conditional. More formally, the negative sentiment is targeted at aspect $a \rightarrow$ *camera* of entity $e \rightarrow$ *phone X*. In the next example the effect is same but inversed.

Example 4: "*If you guys want a smartphone with a really good camera, buy this phone X*". It is evident that the sentence is a positive view of the camera of the subject phone. It is very similar to example 3 and the slight difference is the negation of the word *buy*.

Conclusively, conditional sentences are a formidable task as demonstrated by the examples given. The question that is raised is whether conditional sentences are that important in order to be handled; this question, effectively, is whether there is a large amount of conditional sentences to excuse an extensive study [14].

2.3.3 Named Entity Recognition (NER)

Named Entity Recognition (NER) nowadays a term that is quite frequently used and talked about in the field of *natural language processing* and its broader purpose is to identify named entities found in textual sources that belong to a number of known classes or categories; for instance, companies, natural persons, locations. Thence, these identified entities might need to be stored and processed. In addition, it constitutes a major assignment of information extraction and the problem is usually divided into two main subtasks: **entity** detection and classification [15], [16].

In the context of Sentiment Analysis, the main objective of NER is to discover to whom opinions and sentiments are expressed towards, i.e., what is the entity under discussion, and this is a challenging problem as it will be shown through examples further.

The advent of blogging, micro-blogging and social media in general has shifted the focus of information extraction, and this relatively new form of content is now under research. In particular, tweet posts and Twitter, the leading platform in micro-blogging according to [15] with almost 300 million registered users, producing approximately 500 million tweets per day. Moreover, research has shown that NER techniques applied in large documents and textual sources have extremely high accuracy (close to 90%), but struggle in short tweets with quite low percentages of 30% to 50%. Some of the main issues in tweet posts that make the process harder are: first, tweets are short, up to 140 characters and are, therefore, difficult to understand. Second, ambiguity in the posts, namely, the common uses of sarcasm, internet slang, uncontrolled capitalisation of letters, use of emoticons (e.g. :/), abbreviations (IMO, 'in my opinion'), and hash tags. Furthermore, ideas and concise information are hard to find in short tweets, unlike large structured documents such as articles, journals, etc. and tweets usually have an unlimited range of topics. As a result, new algorithms, specifically targeted at micro-blogs, are under development to deal with the issues mentioned previously [15], [16].

Named Entity Recognition, is used in Sentiment Analysis, summarisation, and has various applications in: e-Commerce, e-Business, and e-Rulemaking [15].

In opinion mining, there exist problems because in many situations, the subject of discussion may not be easily distinguished e.g. "harry potter 1" may refer to the book or the film, "300" may concern the historical Spartan force at Thermopylae

or the 21st century film. Moreover, the object referred to by nouns and pronouns with the use of "*it*" are difficult to understand to what they actually refer to. NER is also difficult because discussion areas on the Internet do not have a preselected topic that can be identified directly [3]. Further, named entities often have aliases and they could be referred to in numerous ways making the identification cumbersome. Lastly, named entity recognition has been studied in recent years and the common approaches for such systems are rule-based and popular learning techniques: supervised, unsupervised by clustering, and weakly supervised learning [2], [16].

2.3.4 Opinion Spam

Fake content or opinion spamming is not only an NLP problem, but a data mining one as well. Social media and networks have offered anonymity to users, which is used as a safety net by people to operate among the general public without having to worry themselves about the repercussions of their actions. Hence, it is natural that this anonymity is not always used with the most honest of intentions [1], [2], [3], [11]. The, so-called, opinion spammers are individuals that post fake reviews of products, services, and organisations with the purpose of promoting or slander a particular entity. The opinion spam problem is of great importance as the opinions of the public should be a trustworthy and sincere. Moreover, spamming may also refer to duplicate posts that appear in many sources [2], [3]. Several review platforms attempt to deal with this issue by allowing users to review reviews and state whether this review was helpful or not, and even flag a review that is misleading and misinformative.

Opinion spam or generalised spamming can be divided into to three main categories; fake reviews, non-reviews, and general reviews without a target item. In the next page, these topics will be covered to a greater extent.

1. **Fake reviews**: as described earlier, fake reviews are fraudulent meaning that the author's opinion is not genuine and it is not concluded by their experience and usage of the targeted product or service. Fake reviews have dishonest motives to promote or discredit some item. Fake reviews are laborious or even impossible to identify in some extreme situations [1], [2], and [3].

2. **Non-Reviews**: non reviews are not reviews; they may be advertisements or some kind of text that provides no opinion to any product or service (e.g. questions) [2].

3. **General reviews without a target item**: These types of reviews are general and do not concern a particular product, they may be rants and raves about the brand (e.g., "*I never buy Apple products, they are the worse*"). These "reviews" may not be fake, but they are not targeting the specific entity in question and can be quite stereotypical, biased and do not assist other consumers and are consequently regarded as spam [2].

2.4 Sentiment Analysis in Practice

2.4.1 Levels of Analysis

Sentiment analysis is researched in four main levels; **word level, document level**, **sentence level**, and **aspect level** (see figure 2.1 [3]) [2], [17].

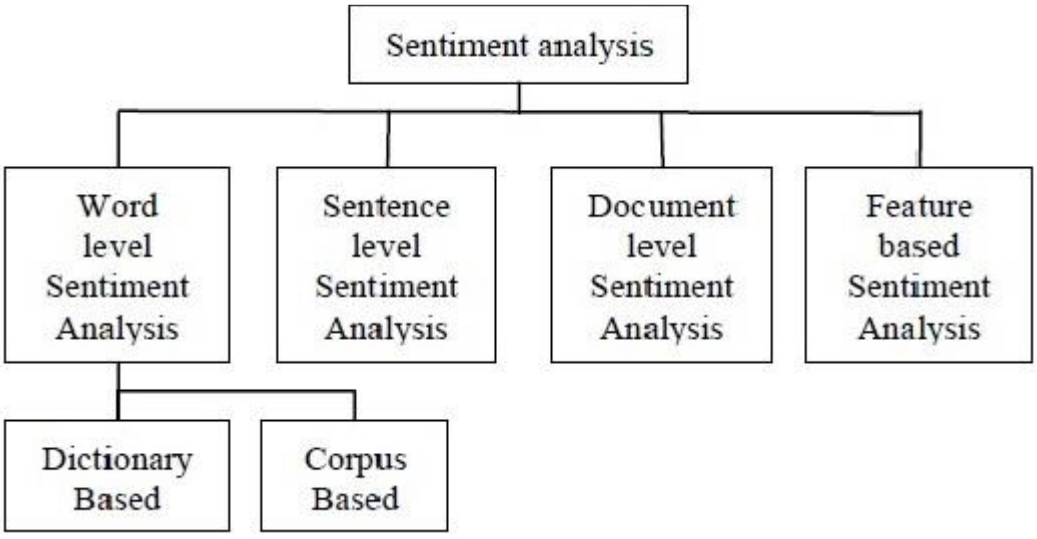


Figure 2.1. Sentiment analysis levels (excerpted by [3])

Document level analysis refers to the classification of a document as a whole, pertaining to whether it expresses a positive or negative sentiment towards an entity. The assumption the following type of analysis makes is that the document expresses an opinion for a single object only. As a result, it is not applicable for documents that relate to more than one entity. For example, reviews written for a specific movie can be dealt with document level classification because the target entity of the reviewer is undoubtedly the movie per se. On the other hand, it is somewhat dangerous to use document level analysis as mixed opinions about various entities appear often in text sources. For this approach, standard machine learning classifiers have been used such as, naïve Bayes, support vector machines, and maximum entropy [2], [3], [17]. All three of these machine-learning classifiers will be presented and discussed more thoroughly in section 5.2.1.

An example of simple document level analysis: consider a review page in a popular movie website such as the Internet Movie Database (IMDB). The whole text of the review is extracted and the purpose is to identify whether the review is positive or negative. In this case, the sentences and words are examined but with the assumption that they refer to the entity (the film). Hence, a naïve approach would be to examine the sentiment words that appear in the text and assign points e.g. add points for a positive word and subtract for a negative word (note that words may have different point values depending on its strength). Finally, check if the number is positive or negative and respectively state what the whole review was interpreted as.

Unquestionably, **sentence level** analysis is necessary because sources may present various opinions for many entities. As the name suggests, this form of analysis focuses on the classification of individual sentences in a corpus in order to distinguish positive and negative sentences or ones that do not express any opinion (factual sentences). Further, the targeted entities under consideration must be known in this domain as well, but it is very flexible and useful to extract sentiments in conditional and comparative sentences mentioned in section 4.2, and analysis of different features of entities. [2], [3], [17].

Additionally, some examples on sentiment analysis using sentence and document analysis will be provided. The following figures have been generated by the Stanford NLP online live demo that predicts sentiments of film reviews based on their new model for sentiment analysis. Consider the following text opinion regarding an automobile.

"*This car is terrible, the design is good, but the engine makes noises all the time, it is extremely annoying. Not recommended!!*"

The model used provides a sentence structure and outputs a visual representation of it in a tree (see figures 5.2 and 5.3) where each word and sentence are divided into parts and sentiments are assigned to each node and by examining all nodes the final result is output and passed on to the initial node.

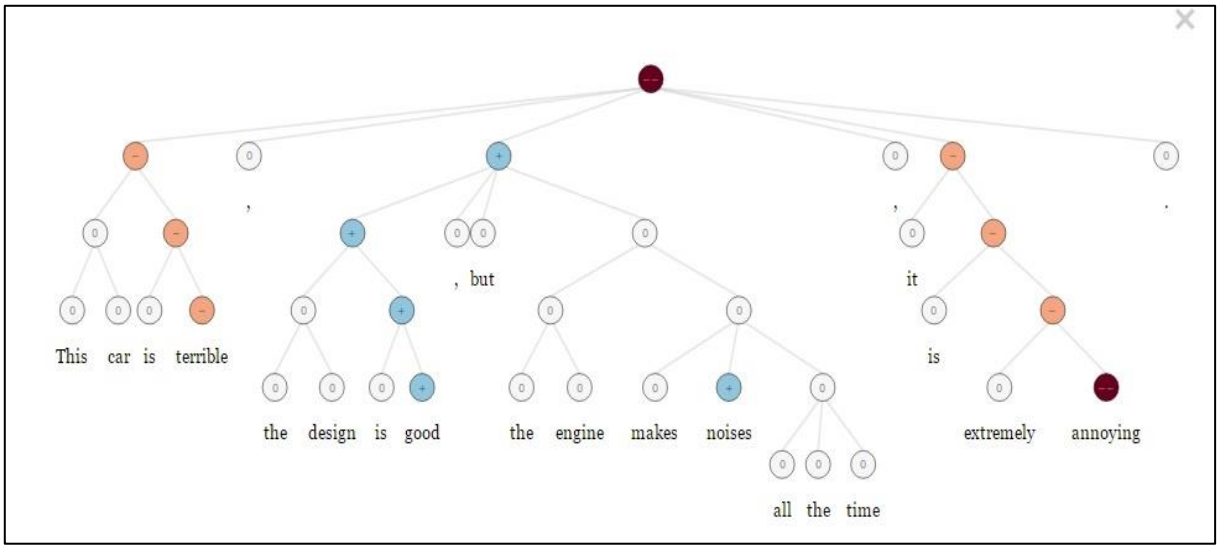


Figure 2.2. Sentence structure in a tree (1) (excerpted by [43])

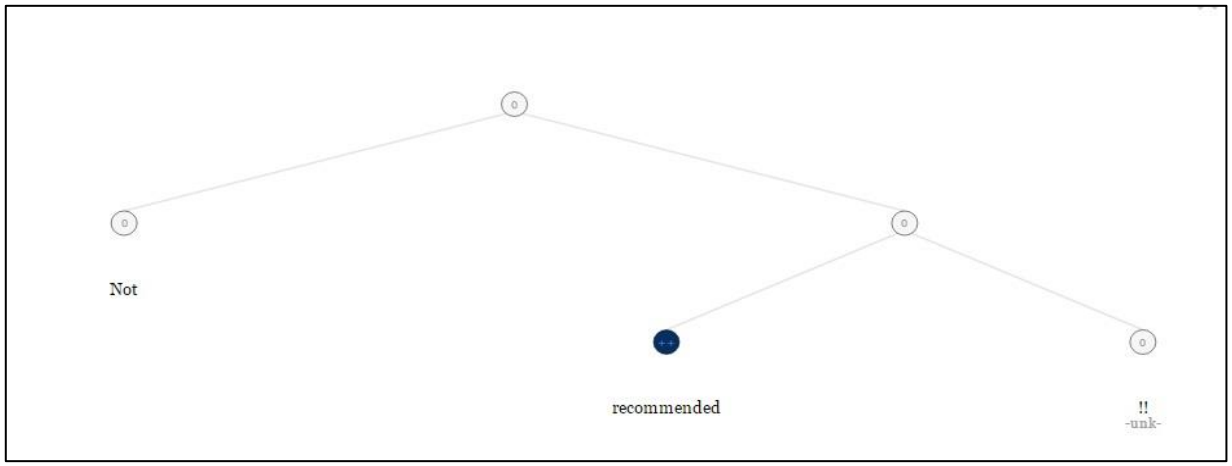


Figure 2.3. Sentence structure in a tree (2) (excerpted by [43])

In the above figures it is observed that every sentence (and word) is examined and has a particular value (++, +, --, -). The sentiment strength is measured by the word polarity and evidently, in figure 1, the predictions were correct apart from labelling the word 'noises' as a positive experience, even though it is not. However, in figure 2, after inspection it can be seen that the classification was tricked by the simplest form of negation. It falsely labelled the part of the review: 'not recommended' as ++, i.e., highly positive. However, it disregards the fact that there is a 'not' before the word that inverses the meaning, thus, the upper node should have negated the result.

Aspect level analysis is somewhat more complex as its function is to detect and classify sentiments expressed in a document but simultaneously inspect to what aspects or features each of them refers to. This is important as in numerous situations, such as reviews and discussions in forums, people express opinions for specific attributes and aspects of entities; e.g. comments about a phone's battery and simultaneously different comments for the same phone's Wi-Fi signal. This is handled in the opinion definition in section 1.3 and it is extremely important as in most cases aspects of entities are the targets of opinions. The notion of aspects has already been indirectly introduced, but nonetheless, some new examples will be presented next [2], [3], [17].

Example 1: "*The laptop is ok, the screen quality is great, but the battery sucks*". In this example there are two aspects of a laptop discussed, that is, the screen quality and the battery. These constitute the aspects of the entity laptop and the sentiment expressed is different for each of them.

Example 2: "*Although the hotel was good, the staff was quite rude to us sometimes*". Figure 2 shows an aspect analysis of a mobile phone and Figure 3 an aspect comparison between two mobile phones.

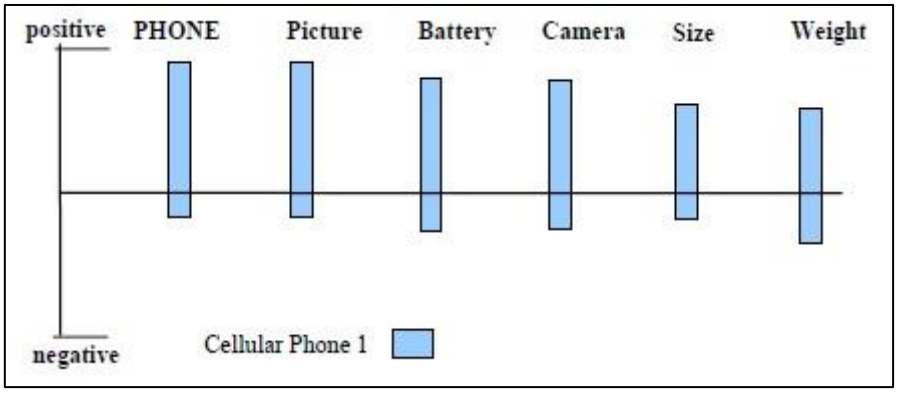


Figure 2.4. Mobile phone aspect sentiments (excerpted by [8])

In the previous image the aspects/features of the cellular phone are displayed and towards which sentiment they are favoured. In the case of figure 2, the majority of sentiments expressed are positive about all features of the phone.

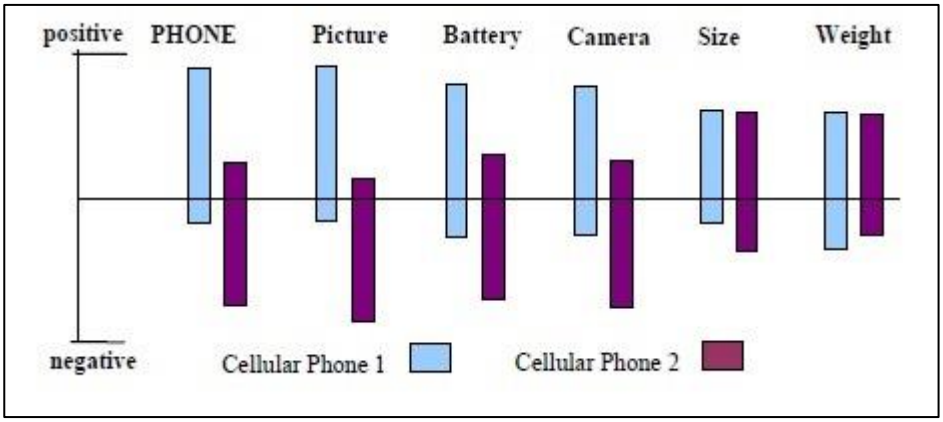


Figure 2.5. Mobile phone aspect sentiments – comparison (excerpted by [8])

In the above figure, the results of a comparison of features between two cellular phones are presented. Each phone has a different colour on the graph and the weights of the sentiments are compared and it is observed that phone 1 is stated to have better features than phone 2 besides the weight feature.

Word level analysis is related with examining and determining the sentiment and strength of each word. Word level sentiment analysis is the most fundamental; it is used in all levels of analysis, and is convenient as it deals with individual words and their polarity which is usually standardised depending on the domain certainly. For example, words (good, nice) can be defined as positive sentiment words of strength +2. This can be achieved by using dictionaries where words are mapped to sentiments or corpus based methods as shall be examined later [3].

2.4.2 Sentiment Analysis Techniques

Hitherto, the current techniques and methods used for sentiment analysis have not been presented. Therefore, this section will discuss the two main types and subtypes of methods that perform sentiment classification. The two main and broad categories that can tackle the problem of sentiment analysis are: **Machine Learning** and a **Lexicon based** approach. In the following subsections, several supervised learning classifiers will be analysed and further, a short description of some lexicon approaches used. See figure 5.6 for an overview diagram of different methods used for opinion mining [2], [17], [18], and [19].

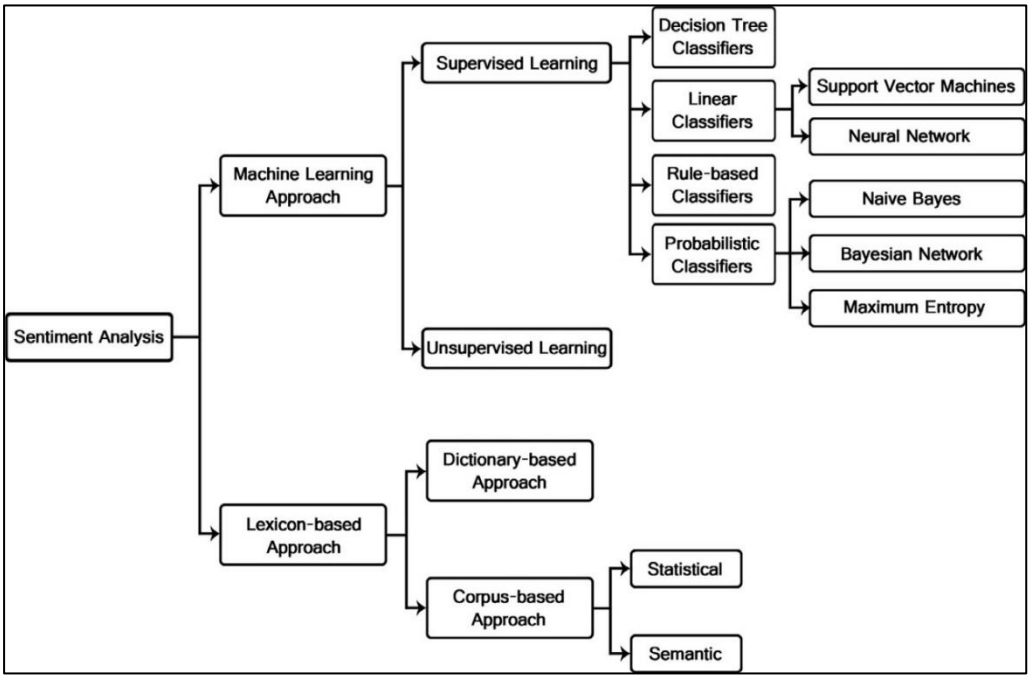


Figure 2.6. Sentiment analysis techniques and approaches (excerpted by [19])

2.4.2.1 Machine Learning in Sentiment Analysis

In this part of the paper, the three most common supervised learning classifiers used will be presented and examined, including the details on their foundations. Generally, the machine learning (ML) approach in sentiment analysis relies on its mighty algorithms that treat the problem as a standard text classification issue [17], [19], and [20]. Before we examine the classifiers, a small introduction on the general machine learning concept will be delivered. To start with, machine learning techniques are branched into two main categories:

1. Supervised Learning
2. Unsupervised Learning

The term *Supervised* is used to state that correct examples are fed to the algorithm and are fully labelled to guide the training process. Specifically, examples with correct classifications are used to gather valuable data and then based on the training data, perform future operations [4]. [5], [19]. Maintaining such datasets is also important in the usage of probabilistic classifiers and statistical methods that rely, by definition, on prior distributions. This modus operandi is called supervised learning [4], [19], and [20]. Moreover, in this domain, the sample dataset is used to train the supervised algorithm before it is tasked to solve real examples. The rise of the, now popular, supervised techniques is a consequence of the availability of a great deal of data and other relevant sources. The next ML methods are supervised models: [4], [5], [9], [19], and [20].

- Naïve Bayes
- Maximum Entropy
- Support Vector Machines (SVM)
- *K* - nearest neighbours (k-NN)
- Decision Trees

On the other hand, in unsupervised learning there are officially no training data provided to the model, ergo, no *a priori* knowledge and training. However, in reality, many unsupervised learning techniques tend to start with at least a small set of annotated data as a starting point, and are thus called semi-supervised. Semi-supervised learning start with the so-called *seeds* that are later used to expand the knowledge base. Purely unsupervised learning methods are performed by clustering [4], [15], and [17].

Subsequently, we will examine classification. Classification is performed with use of classifier algorithms and its objective is to determine and interpret the records of a dataset. Mathematically a classifier is defined as a function f that binds vectors $x \in X$, where X is the feature space, in order to extract labels $y \in \{1, \dots, C\}$; these are the classes. In other words, a classifier is a tool that given a set of items (the data) it appoints them in one out of N classes [17], [22], and [24]. In the case of sentiment analysis, the labels usually consist of positive and negative. Furthermore, there are various classifier types: probabilistic classifiers (e.g. Naïve Bayes, Max Entropy), linear classifiers (e.g. Support Vector Machines), and decision tree and rule based classifiers. The statistical classification is usually the parent of most of these mathematical models [17], [22], and [23].

The classification problem is formally expressed the following way: there is a set of features $M = \{f_1, f_2, \dots, f_i\}$ that can be discovered in a document d. Thereafter, we denote with $num_i(d)$ the occurrences of f_i in document d. Afterwards a vector can be used, for this reason, each document d is placed in a vector such that: $\overrightarrow{d} := (num_1(d), num_2(d), \dots, num_l(d))$. Then, the model is used for computing the class label and mapping it according to the methodology practiced [4], [17], [19], and [22]. To elaborate further, assigning a label to an instance is the primary purpose of a classifier. For example, a specific instance (in machine learning terms) is a Facebook status. Then, the classifier is used to assign a label to that FB post from the following set: $Labels = \{positive, negative, neutral\}$. This particular example describes a multi-label classification, in contrast to the standard binomial or binary classification that focuses on two classes.

Naïve Bayes

The Naïve Bayes (NB) classifier is one the most intelligible and frequently used classifiers. It is a probabilistic classifier and has to be trained (supervised learning). Moreover, it belongs to the larger field of Bayesian probabilities and statistics and its simplicity distinguishes it from other methods. The NB classifier is used for text classification and has applications in spam detection in e-mails, language recognition, and sentiment extraction [22], [23]. The NB classifier, based on the dispersion of words in a text source, calculates the prior probability of a feature being labelled to a particular class. The training data provided compose the basis for this posterior probability [23], [24]. The location of any word in the document does not concern the NB model. Further, it uses the Bayes Theorem (5.1) to make predictions to which label a set of features will be assigned to. The general form of the Bayes Theorem is the following [22], [23], [24], [25].

$$P(C|F) = {P(F|C) \times P(C)}\Big/{P(F)} \qquad \textbf{(2.3)}$$

In the above formula, $P(C|F)$ is the probability of features *F* being assigned to class *C*, this is in fact what must be calculated in order to select the highest probability for the classification. $P(F|C)$, is the probability of having features *F* given class C. And $P(C)$ is the probability of class *C* instances, namely, the frequency of class *C* in the given knowledge base. The NB model makes a daring assumption, that every feature is independent of the class label, or formally: the features are *conditionally* independent. This assumption is made consciously even though features are not always independent with each other; however, the process is simplified with this assumption [22], [23], [24], [25]. Therefore, given this naïve assumption, expression (5.1) can be rewritten:

$$P(C|F) = \frac{P(C) \times \prod_{i=1}^{n} P(F_n|C)}{P(F)} \quad \textbf{(2.4)}$$

In expression (5.1), $P(F|C)$ becomes the product of the probabilities of each feature *f* given class C due to the fact that they are assumed to be independent, hence the probabilities are multiplied.

To conclude on the Naïve Bayes classifier, there are two main advantages that describe it. First, it is simple and fast to use i.e. in order to train the classifier there is no need for an enormous training data collection. Second, according to [26] it is quite efficient compared to more sophisticated solutions, having an approximate time complexity of $O(Np)$ in Big-O notation; where *N* is the sum of training examples provided and *p* the number of features. In addition, it is not a mandatory requirement to possess high-end hardware in terms of both CPU and memory power in order to perform classification using a Naïve Bayes classifier [23], [24], [25].

Lastly, there is another approach using what is known as *Bayesian Network*s. They are also probabilistic and are used to create graphical models to represent a knowledge base and each node in the graph is a variable. Bayesians Networks (BNs) are modelled with the use of graph theory, probability theory, and statistics. In contrast with the NB classifier, the exact opposite assumption is made regarding variable dependencies, that is, all features are dependent to each other. However, due to its high computational complexity it is not used as frequently as the NB model [19], [27], [28].

Maximum Entropy

The Maximum Entropy (MaxEnt or ME) classifier is another probabilistic, but discriminative classifier that is in the exponential models' class. It is used to tackle various natural language processing issues such as text classification, sentiment analysis, and language detection [19], [22], [29]. Unlike the Naïve Bayes, MaxEnt does not assume feature independence, but the computational complexity of MaxEnt is quite higher than the NB. The fundamentals of the MaxEnt classifier rely on the

Principle of Maximum Entropy, which in short, looks at probability distributions of the available models and the one that seems to suit the current state better is assumed to be the one with the largest entropy and is subsequently chosen. Furthermore, distributions must fulfil certain constraints [19], [22], [29] [30], and [31]. According to [22] and [30], MaxEnt can occasionally surpass the Naïve Bayes classifier in performance in some situations.

For the Maximum Entropy classifier, if the reader recalls the definition of classification given in section 5.2; very briefly, there is a set of features that compose a category (class) in a document and all documents are represented by a vector by their feature counts. We can look at features as words in documents. Subsequently, the MaxEnt classifier has to compute the probability that a document *d* belongs to a class *c*. This probability can be calculated like this [22], [30]:

$$P(c|d) = \frac{1}{Z(d)} \exp([\sum_{i} l_{i,c} F_{i,c}(d,c)]) \qquad \textbf{(2.5)}$$

In expression 5.3, $P(c|d)$ is the probability that a class is seen in a document. The function *Z(d)* is a merely for normalisation. Next, $l_{i,c}$ represent feature weight parameters. This set is used to increase the entropy of the distributions. Further - $F_{i,c}$, is a function *feature f to class c*. It is defined in the following way:

$$F_{i,c}(d,c') = \begin{cases} 1, & num_i(d) > 0 \ \wedge \mathrm{c} = \mathrm{c}' \\ 0, & else \end{cases} \qquad \textbf{(2.6)}$$

The above binary function outputs 1 if a document that belongs to a category contains a given feature *f*, and 0 otherwise [22], [30].

The MaxEnt approach is useful and robust in situations where there is a lack of data on posterior distributions and there is simply not enough data to make assumptions (like the Naïve Bayes). Moreover, when it is known beforehand that

features are not conditionally independent, MaxEnt is a suitable choice as its principle is to make fewer assumptions as possible. While, its computational complexity and training time are both greater than the NB, it is still competitive and gives consistent results [19], [22], [29], [30].

Support Vector Machines

Support Vector Machines (SVM) is a supervised learning method for text classification and pattern matching and the classification conducted using SVMs is non-probabilistic linear. The fundamental idea behind it is using decision planes to mark boundaries in order to separate objects that belong to different classes. For example, consider a set of blue objects on one side and a set of red objects on the other side, and they are separated with a straight line (linear). See figure 5.7 for an illustration [19], [22], [32], [33].

As reported by [22], Support Vector Machines are extremely successful in standard text classification issues, beating Naïve Bayes on many occasions. On a more technical note, the purpose of the training procedure in SVM is to discover a hyper plane, let vector $\vec{h}$ depict it; that divides all the document vectors into different classes. Consequently, we can define the correct class of document d_j as: $c_j \in \{-1, 1\}$, (1 for positive and -1 for negative). Therefore, a training dataset consists of a set of *K* input vectors x_1 ... x_K and all of them have a class *c*, as shown earlier (-1 or 1). Moreover, the dataset is split linearly, that is, they can be separated with their respective classes using a single hyper plane [4], [19], [22] [33], [34]. Ergo, the vector is derived like this:

$$\vec{h} := \sum_j \left(a_j \, c_j \, \vec{d}_j\right), \;\; a_j \geq 0 \qquad \textbf{(2.7)}$$

The document vectors $\vec{d}$ are called support vectors because they participate in forming $\vec{h}$. Subsequently, the classification of test cases simply befalls on examining on which side of $\vec{h}$'s hyper plane they are found.

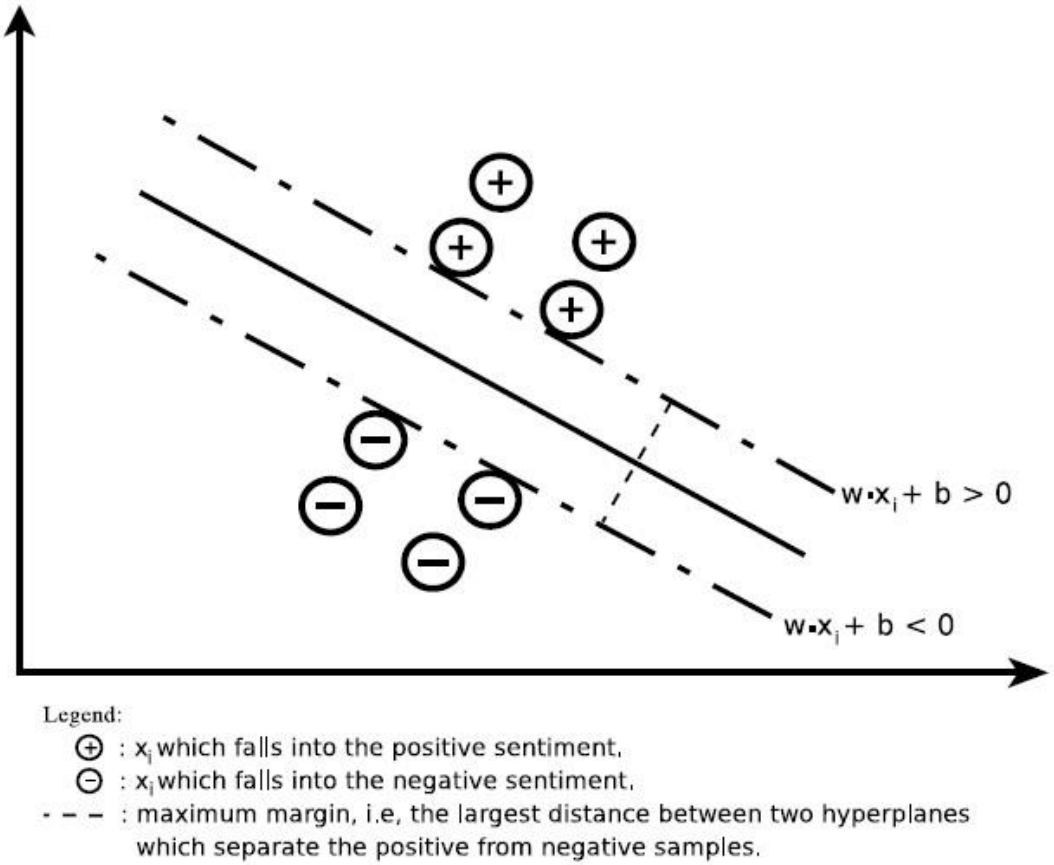


Figure 2.7. SVM classification in Sentiment Analysis (excerpted by [21])

Moreover, in relation to the vector machines, the SVM approach has been highly effective in domains such as, image processing, face recognition, text classification, sentiment analysis, and spam filtering. Traditionally, SVM uses feature vectors that are formed from a slew on input features. Additionally, in contrast to feature vectors, many variations of SVM learning can utilize certain Kernel functions (similarity functions), as such: [32], [33], and [34].

$$K(X_i, X_j) = \varphi(X_i) \cdot \varphi(X_j) \qquad \textbf{(2.8)}$$

In compliance to this Kernel function (5.6), there are four commonly used types of kernels (linear, polynomial, RBF, sigmoid) and they are the following:

- Linear: $K(X_i, X_j) = [X_i \cdot X_j]$. **(2.9)**
- Polynomial: $K(X_i, X_j) = (\gamma X_i \cdot X_j + C)^d$. **(2.10)**
- Radial Basis Function: $K(X_i, X_j) = \exp[-\gamma|X_i - X_j|^2]$. **(2.11)**
- Sigmoid: $\tanh K(X_i, X_j) = (\gamma X_i \cdot X_j + C)$. **(2.12)**

Where C, γ, and d are Kernel parameters. Lastly, it is important to mention that each of these kernel functions have a different complexity and greatly differ in execution time. For instance, as it will be verified later, using the polynomial function (either quadratic or cubic), the classification of a set of documents is quite slower than the linear.

2.4.2.2 Lexicon Approach

Furthermore, there are various specialised Lexicon based approaches that use an opinion lexicon to detect opinion words that express a positive sentiment or a negative one. It is mostly utilised with either dictionary oriented methods or corpus based, this type of approach is known to utilise a knowledge base (e.g. WordNet) i.e., they possess a large database of words and have set predefined sentiment strengths on these words. Subsequently, these sources, like WordNet and SentiWordNet, are used to identify synonyms and antonyms of words found in text to expand the range. Further, machine learning and lexicon techniques are not mutually exclusive, that is, lexicons can be used in classification using machine learning [3], [18], [19], and [35]. Additionally, various rule based systems may also be employed as a solution, for instance, architectures based on logical connectives, which are heavily depended on the logical implication, namely, "if – then" rules [18], [19].

Dictionary Based

Dictionary based approaches rely on a list of words that have a predefined sentiment polarity strength. This list is typically small and acts as a seed in order to match the words with synonyms and antonyms from online sources such as SentWordNet 3.0 [35], WordNet (a lexical source for the English language). These online dictionaries have a gargantuan amount of related words and as a result, they can be extracted and placed in positive and negative word lists. While it is quite facile to use such an online dictionary, there is a major limitation in the dictionary base approach. The polarity of words is general and cannot be tweaked to suit a particular context or domain. As discussed earlier, context in specific domains is important and the same vocabulary may have a completely different meaning in different contexts [1] [2], [3], [19], and [35]. For this problem, it has been considered to combine such an approach with a rule based system to identify the domain under discussion, as explained by [19].

Corpus Based

Corpus based approaches depend on statistical or language syntax methods and it is typical to start with a manually selected list of seed words (positive, negative) and then, this list is used to discover more sentiment words from a big corpus. In this approach some linguistic rules are used to identify the orientation of word pairs when they are combined with logical connectives (and, but, or). For example when two adjectives are combined with an AND they usually have the same orientation (e.g., "*Good and reliable*"). Even though orientations are helpful, again, the same limitation of the Dictionary Based method holds, i.e., that orientation differs in various contexts and domains [2], [3], and [19].

2.5 SA Tools and Technologies

There exist a vast numbers of tools, applications, and APIs that conduct sentiment analysis on a given dataset. The main concern is textual analysis only. Some of the most popular tools and technologies that are used for sentiment analysis are: Gate [36], an open source software for NLP, developed by the University of Sheffield, including sentiment analysis. Tools from the company Lexalytics, specifically, Semantria [37] and Salience, with both providing sentiment analysis services through APIs and Excel. Further, RapidMiner [38] is a software platform for machine learning, and sentiment analysis among other features, Weka [39] from the University of Waikato provides machine learning algorithms to perform classification on text and data mining in Java. Additionally, there are various APIs such as LingPipe [40], the Google Prediction API [41], Python NLTK [42], Stanford NLP [43] RepuState [44], Alchemy API [45], and Bitext API [46], which all provide the capabilities of conducting SA among other textual analysis applications; some to more extent than others. Next, a presentation for few of the aforementioned sentiment analysis tools is included.

2.5.1 GATE

General Architecture for Text Engineering (GATE) is a natural language processing tool that can tackle almost any text engineering problem. It is open source and provides numerous functionalities to users. GATE is highly configurable through its integrated development environment (IDE), Gate Developer. Users are able to load a document, in raw text or already marked and apply various text processing operations such as: sentence splitting, POS-tagging, Named Entity Recognition and others. There are also numerous plugins for GATE and their development is driven by the general NLP community.

Users can customize and set a pipeline of operations to be performed on their documents. GATE can be configured to perform sentiment analysis using machine learning plugins. Regarding engines, it offers Naïve Bayes Weka, the Perceptron Algorithm with Uneven Margins (PAUM) and Support Vector Machine from the

LibSVM implementation that are fully configurable. For the SVM, one can set linear or polynomial kernels for example. In figure 8, the interface of GATE Developer is displayed, with a product review been loaded from Pang/Lee dataset [22] as raw text from a standard text file [36].

Additionally, it has interactions with other APIs and tools like the LingPipe API pipeline, the Alchemy and similar tools intended for text engineering and, of course, sentiment analysis. Furthermore, GATE offers a rich API (GATE Embedded); which is for programmers who either wish to develop plugins for the application of GATE, or use GATE features in an application of their own. A usual approach would be to create programmatically the processing and language (corpus) resources and pass documents to them and receive feedback.

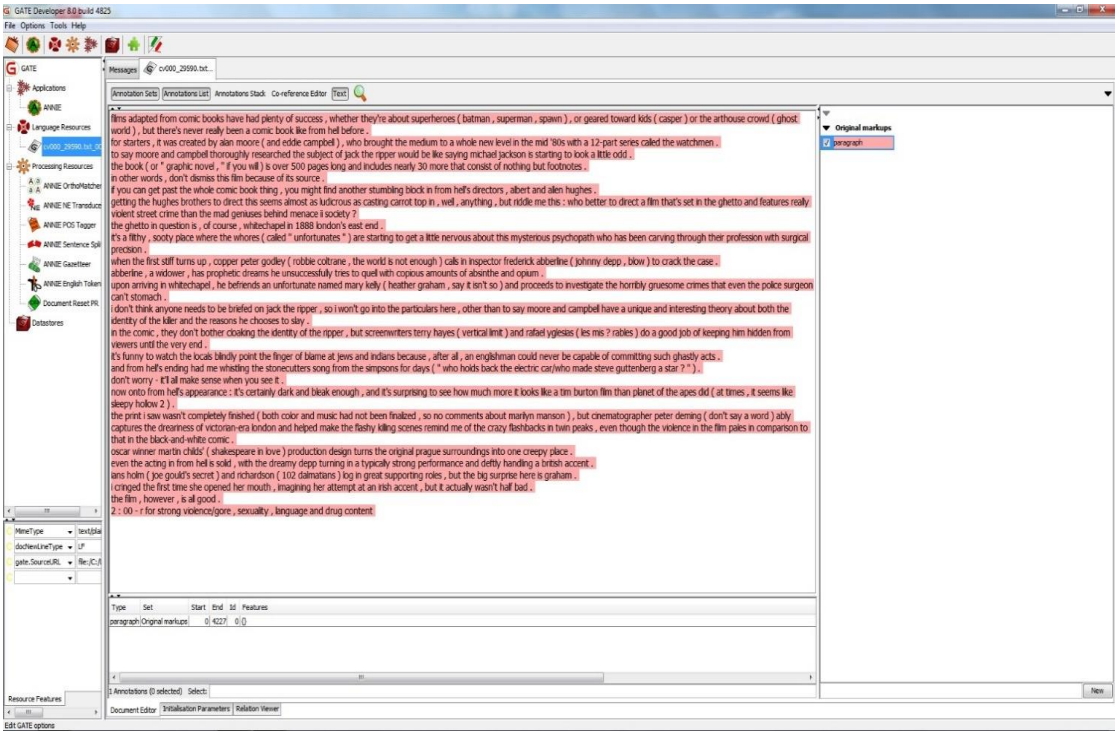


Figure 2.8. GATE Interface - Loaded Review (excerpted by [36])

Here are some text processing resources that GATE Developer offers:

- POS Tagger
- Sentence Splitter

- Morphological Analyzer
- English Tokenizer
- Batch Learning (ML)
- NE Transducer

2.5.2 Rapid Miner

Rapid Miner is a software platform that provides predictive analytics, data mining, and more through a studio. It can be used for sentiment analysis, risk modelling, quality, and others. Rapid Miner offers a studio with an environment that is quite user friendly, given the fact that no low level technical knowledge (programming) is required to use it. For sentiment analysis, it offers a set of predefined classifiers (Naïve Bayes, Support Vector Machine (linear), etc.) and users can load their documents or use other resources. With RapidMiner, one can train a selected classifier with a training dataset and then apply it and analyse its performance. [38].

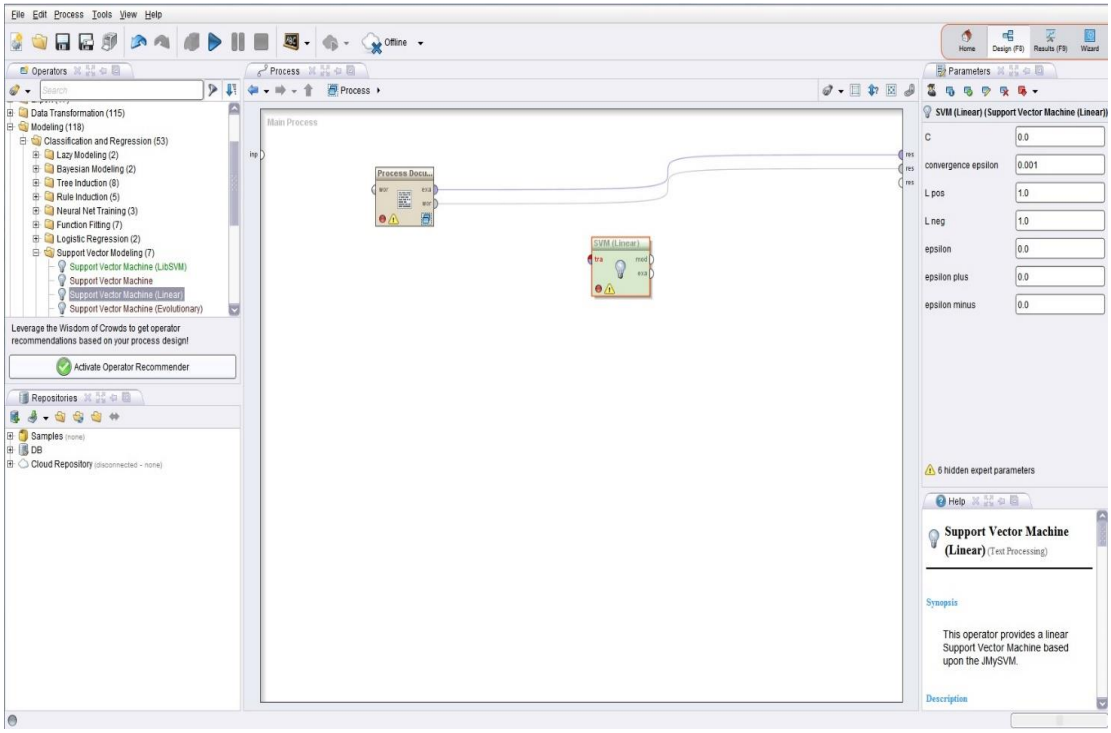


Figure 2.9. Rapid Miner Studio Interface (excerpted by [38])

2.5.3 Other Technologies

Sentiment Analysis as a web service is available and offered by many companies as briefly stated in the introduction of this section. We are interested in these APIs as regards their underlying technique of performing sentiment analysis and effectiveness on the problem. The company of Alchemy offers a web service API for various text analytics. Some of their available functionality include text extraction, named entity recognition, face detection, sentiment analysis, and others. Their sentiment analysis API can operate on accessible web pages or HTML content and raw text. In addition, clients can pinpoint specific named entities or phrases that they wish to extract sentiment from. Directed sentiment analysis is really important and useful, since data may contain various sentiments regarding different entities such as companies or people. Their API can be called with standard HTTP methods and the responses are JSON or XML. According to the Alchemy website, they are following a machine learning approach and in extension supervised learning models.

Another major player in sentiment analysis is Bitext. A text analytics company with APIs for sentiment analysis, entity extraction, text categorization, and topic extraction. Bitext has developed an extremely accurate system according to their website and their method is called deep linguistic analysis. They are working with more advanced computational linguistic features and concepts that allow very fine-grained analysis. Their API was tested in a trial version and their sentiment analysis module handles tough linguistic challenges like negation. Furthermore, Bitext API categorises the sentiments on a given text according to the entities or concepts involved and provide a sentiment score for each separately. It is definitely not a pure machine learning solution, but rather, is enhanced with an in depth linguistic feature extraction to handle natural language difficulties.

2.6 Applications of Sentiment Analysis

A substantial amount of research has been conducted on the field of sentiment analysis with the goal of discovering approaches to acquire subjective and unbiased feedback from available sources [1], [2], and [4]. The rise of web applications with Web 2.0 technologies that facilitate the expression of opinions, appraisals, and emotions to the public have prioritised and advanced further the study of sentiment analysis [1], [2], [3], [5]. This has happened because Web 2.0 proliferation and personalisation have led to a colossal increase of user driven content.

In the introduction, the example of sentiment analysis in product and service evaluation was used to acquaint the reader with the concepts. Nonetheless, the applications of sentiment analysis extend the domain of review based requirements. In the following section, there will be a short presentation of some popular applications of opinion mining.

2.6.1 Review Oriented Platforms

Websites such as review sites, blogs, social networks and media, and forums are domains to which sentiment analysis can be applied to extract subjective material about businesses, products, and services. Information regarding products, namely, reviews and other forms of opinion, targeted at a particular product are crucial to businesses [1], [2], [3], [5]. And this is, in fact, what businesses that provide services or sell products use opinion mining for [3], [5].

Platforms of this nature exist; one could consider particular sources that constantly extract and classify opinions about any type of product; a great example is epinions.com. Moreover, there are many movie review websites e.g. rotten tomatoes, or multi domain review platforms (TV shows, video games, films) e.g. metacritic. These platforms offer the ability to users to post what they think about something, including, at the same time, opinions and analysis from trusted sources and known critics. It is important to highlight that such platforms have attracted a great number of users and they compose a reference for other consumers because they provide valuable feedback.

This offers power to customers and hence, review sites have captivated an enormous fan base and people that are dedicated to assist fellow consumers through detailed and honest reviews. On the other hand, the anonymity of the Internet leaves, naturally, various potential exploits of this fact and the relevant issues will be examined further in the report. Finally, reviews are not necessarily restricted to goods and services. The concept can be expanded to candidates of public office or government policy issues, for instance [1], [2], [5].

2.6.2 Technology Component

Opinion mining can enable various other technological applications in other systems as a foundation or extension.

Recommendation applications for example, where negatively stamped products will not be recommended. These types of applications would usually recommend various items based on the feedback of users found online and not solely on the discretized review form of the application itself. Such applications exist, thus the sentiment analysis component would merely aim to significantly enhance them.

Further, question answering in which highly opinionated answers are expected is another field where opinion mining can prove beneficial. It can be helpful as people tend to value answers that contain perspectives of different people and thus advise other people in a better manner according to [1], [2], and [3].

2.6.3 Political Science and Other Domains

The influence of opinions in politics is evidently of great importance and extracting opinions of voters in countries is something to consider. Political parties would use it in order to make various predictions and understand the voters' reasoning behind their choices; such information would be invaluable to them. Hence, it would enable them to make adjustments to attract more voters [1], [3], [9].

Furthermore, other projects attempt to classify and produce a clear summary of the positions and opinions of politicians and government figures, with the purpose

of providing concise and reliable sources for voters to have access. This would also enhance the quality of information [2].

Moreover, electronic rulemaking (eRulemaking) is an application of opinion mining. Electronic rulemaking refers to the use of technology by agencies to assist them in decision making pertaining to some policy, law or other regulation, the government wishes to issue. In this platform citizens would express their opinions and they would be analysed in order to produce some important conclusions about the policy in question. The benefit of the aforementioned concept is quick feedback from active citizens [2].

Additionally, in social media and social networks, opinions about public figures and social phenomena appear constantly and are worth being investigated [1], [3] [9], [10]. By the same token, sentiment analysis can be applied in education. Students could input feedback concerning units, lectures, etc. through a mobile application, for example, and it can be processed to identify the general sentiments of the students. And lastly, applications of sentiment analysis can also be found in psychology, and sociology [3], [4].

2.7 Summary

The area of sentiment analysis/opinion mining is contemporary indeed nowadays and it is a task that requires insight and a solid base on background issues and its underlying foundation. The analysis of various techniques and methodologies is important for the understanding and knowledge required for an attempt at creating such a system. The challenges that are faced have to be considered; however it has been widely accepted that there is no perfect solution for sentiment classification, but highly accurate systems can be developed with the available technologies and theoretical foundations.

It has been explained that SA, as a natural language processing subtask, can be tackled with machine learning techniques that transform and treat the problem as a usual classification problem. And this approach is adopted for the sentiment analysis system this project aims to develop. More specifically, supervised learning models are to be used and this was selected due to the abundancy of datasets that are publicly available and in extension they are more reliable. Furthermore, a significant amount

of research reviewed in this chapter worked with supervised models and lexical methods. Simultaneously, there are plentiful machine learning, NLP frameworks and text analytic tools available that can be used to develop learning models. The majority is merely APIs; web services and SDKs, but there are complete tools that provide an extraordinary environment for such applications.

Hence, the chosen tool for developing and integrating the learning model for a sentiment analysis application is GATE. This tool provides exceptional support and documentation for performing machine learning, other crucial NLP tasks, and text engineering that are sufficient and necessary for a sentiment analysis application. Furthermore, it offers a rich API (GATE Embedded) implemented in Java that can be used to integrate, access, and program resources and applications created through the GUI application of GATE (Developer). As a consequence, the process to be followed using GATE is extremely efficient, because the prerequisites for the model (training, testing, evaluating) can be handled first in the GATE application and afterwards, they can be integrated programmatically through its Java API when the development of the learning models is completed and they meet some quality expectations pertaining to accuracy.

3. Sent-Affect Requirements and Analysis

In this section of the report, the analysis of the expected sentiment analysis system is presented. First, the primary functional and non-functional requirements shall be explained and discussed. Second, the process of creating the sentiment analysis engine will be described in detail. Third, the overall system architecture regarding its components and resources is showcased. Finally, the plan of the project is presented by describing the steps of the solving process.

3.1 Requirements

The sentiment analysis system to be developed should handle the classification of opinionated content and extract sentiment labels from textual data. The context of the application will specifically handle posts from social networking sites – Twitter and Facebook for example, and also, film reviews. The purpose is to classify the input, in text format and determine its polarity (positive, negative, and neutral). The focus of the project will rest in the development of this engine. In greater detail, users of the system should be able to input their data in bulks preferably, because we are interested in analysing amounts of data that humans simply do not have the time to do. Another preferable feature would be the analysis of a webpage directly from the URL, given that it is publicly accessible. Afterwards, the data will be processed by the classifier in order to produce results and hence conclusions regarding the input.

From the system user's perspective, the functional requirements are quite straightforward and facile to comprehend. This is because users need only input data and examine the results; however the application will provide more support along with standard save/load functionalities and so forth. Consequently, the functional requirements of the system are: (See figure 3.1 for use case diagram).

- Read documents
- Read webpage
- Classify inputs (positive, negative, etc.)
- Produce Summarisation of Results

Regarding non-functional requirements, a point of focus is definitely quality, which in this case, refers to the accuracy of classifications. Although perfect classification is unquestionably impossible, a standard of quality has to be set. And in order to accomplish this, experimentation with various classifiers and tools should be conducted to determine what performs better for our circumstances. The other non-functional requirements include:

- Quality (as discussed above)
- Configuration Management
- Interoperability (potential integration in other systems)
- Portability

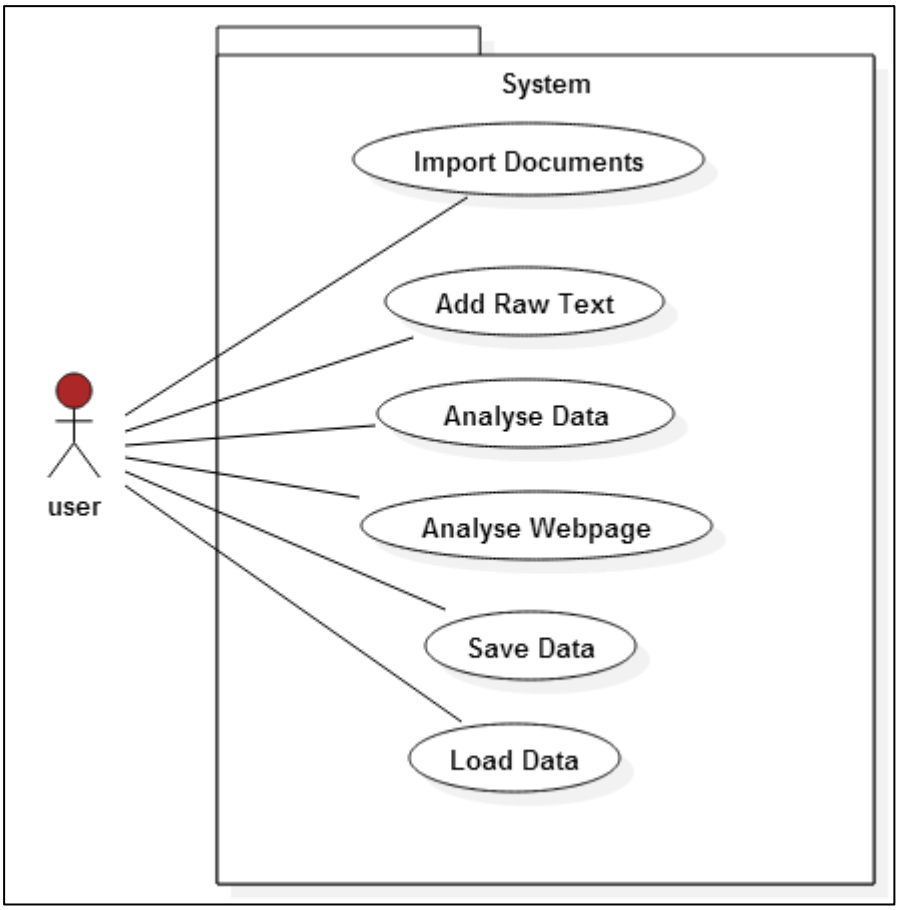


Figure 3.1. Use case diagram

3.2 Building a Sentiment Analysis Model

The standard process of building a model to perform sentiment analysis includes a number of steps and requires the use of a tool (GATE) and available data. The chosen method of tackling the problem of sentiment analysis is through machine learning; specifically supervised learning. Therefore, the general procedure to create a supervised learning model is presented below (see figure 3.2 for diagram).

1. **Collect and Label Data**: It is necessary to obtain a training dataset for the sentiment analysis model. This is also related with the domain of one's application. Subsequently, the training examples should be annotated/labelled with a classification that perfectly describes the test instance according to the understanding of a human. The number of training examples should be at least in the range of thousands and should be clear and well defined examples. There are various sentiment analysis training datasets publicly available for this reason, but they require some annotation depending on the tool used.

2. **Train Model**: Considering that there is a fully annotated corpus of information, we can train a model with our dataset. For this, the classifier is fed with the labelled data and creates a statistical learning model. It is imperative to experiment with different classifier configurations in order to determine which performs better in the given circumstances. The choice of the classifier depends on the availability in the tool to be used and in its evaluation process. With GATE, the experimentation will use the SVM and a variation, the PAUM algorithm. Additionally, there are some considerations as regards the settings of the classifier (e.g. kernel, margins, cost degree, etc.); such concerns have to be researched and undergo through an experimentation process.

3. **Apply Model**: After the completion of the model training, it can be put to the test by providing inputs and observe whether the labels appointed to it by the model are correct and exactly how correct they are. It is vital to point that before a learning model is deployed, it should be evaluated extensively. There are effective methods of evaluation, namely, k-fold validation, which stretches the classifier's capabilities and provides a generalised result on its accuracy.

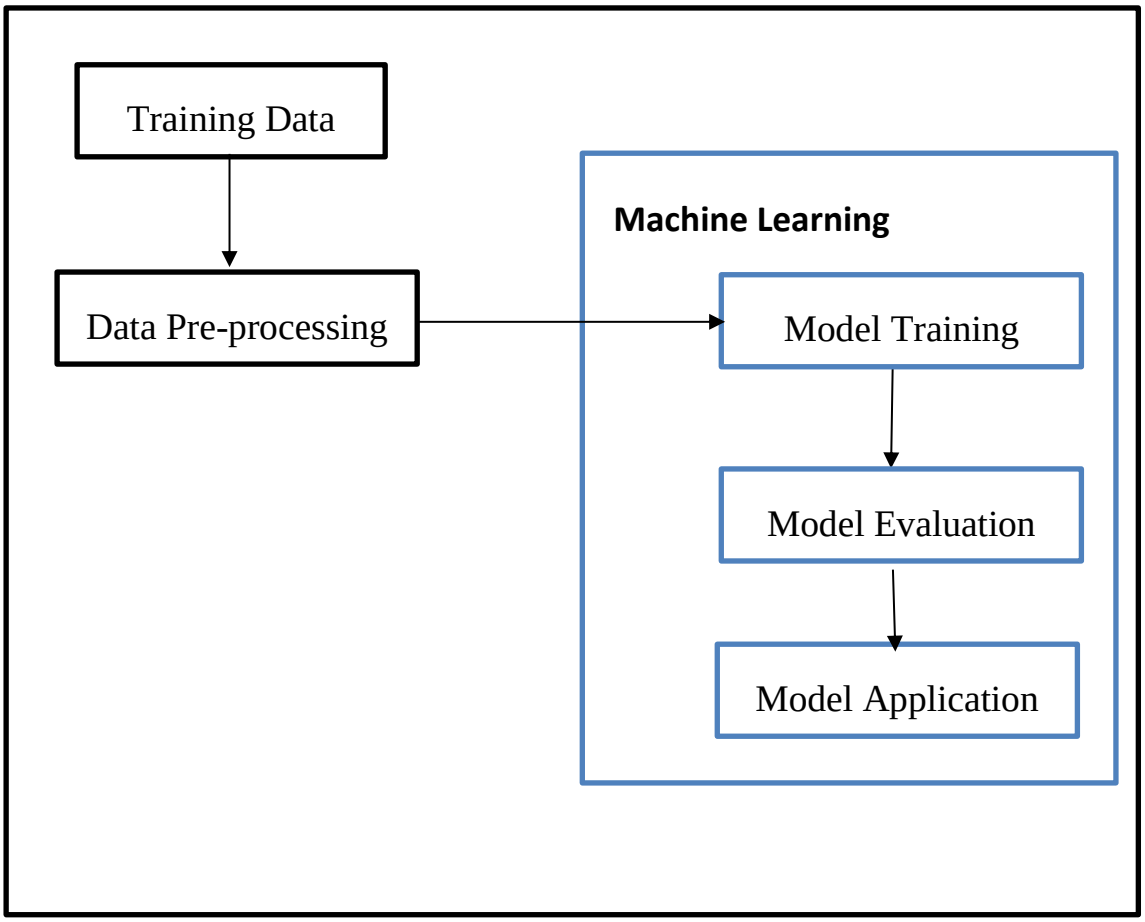


Figure 3.2. Creation of a Supervised Learning Model

This procedure is not specifically designed for sentiment analysis problems, but rather it describes the general process through which a ML developer is expected to follow in order to create a supervised learning model. Classification is a huge task and it is solved with ML techniques and algorithms. The machine learning approach for this sentiment analysis system does not differ greatly from any other classification tasks regarding the general process, albeit there are huge issues and considerations during training for SA models.

As a consequence, the practical development of this project consists of two distinct stages: the machine learning model development as described, and the software system implementation that will be integrated with the model. Thus, the first part of the development is to configure, setup GATE and start the experiments regarding machine learning and using text engineering tools. Relevant data should also be researched and found, otherwise there can be no supervised model.

Subsequently, a first valid model should be produced through the GATE interface application and then begin working with the GATE API that to integrate the developed models into other systems. After the integration part is understood, the user interface of the expected system should be designed and implemented in addition with the underlying business logic. Thereupon, the complete integration can take place, therefore, modular design should be a priority. The machine learning encapsulation library should have only GATE API dependencies and should be easily integrated into other systems. The development process is incremental-iterative. This is useful because the requirements are clear and immobile. The overall architectural design of the system was presented and then the parts of the project are split into separate iterations. The content of the iterations is presented next as well.

Major parts of the project development process:

1. Machine learning model development
    - Data pre-processing
    - Training
    - Testing
    - Produce ML model

2. Programmatic model implementation (depends on 1)
    - Utilise GATE API to create and run applications programmatically

3. User Interface design and implementation
    - Implement required business logic
    - Develop the complete user interface in support of the engine

4. Final integration
    - Integrate parts 2 and 3 for the final release

3.3 System Architecture

Concerning the main components of the expected system; since, the classifier will be trained, the model as a whole belongs to *supervised learning*, as seen in section 2.4.2. Which type of classifier depends on the choice of the tool to be utilised and the experimentation as regards the effectiveness of that classifier. GATE offers SVM implementations. The software application will be written in Java providing facile integration with the NLP framework. The overall architecture of the system can be seen in the below figure (3.3).

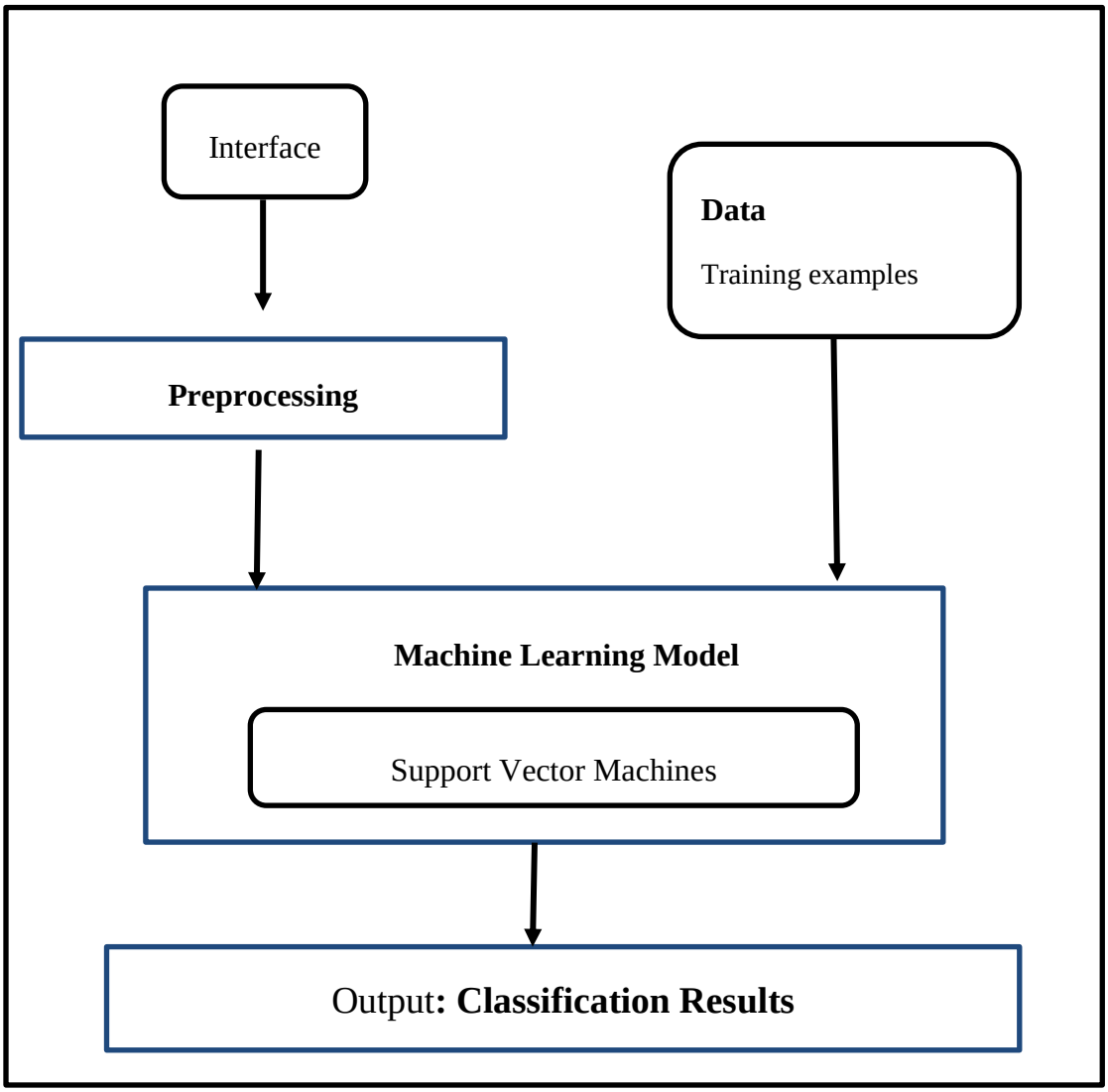


Figure 3.3. System components and architecture

In the above figure, in a high level of abstraction, the main components of the system can be observed. Below, each component is further discussed.

- User Interface: the UI should be quite simple and handy. The main revolves around one task, so the UI should reflect that. Nevertheless, an interface is imperative and is the medium through which text sources are input to the system.

- Pre-processing stage: the pre-processing stage is for structuring the input into a format GATE can understand (XML annotation schema). Also, sentiment analysis is performed on the Web and input can be HTML pages for instance, hence, the mark-up must be eradicated before attempting to classify the data.

- The data aspect of the system refer to the fundamental classifier training data. The training dataset for the learning model are a vital part of a system that uses machine learning techniques. In fact, a ML model is as good as its training data. An objective of this project is to create a corpora of film reviews, and of posts and comments from social networks plus general context language.

- The model with its respective classifier (referred to as Machine Learning in the above diagram) is the primary component of the system. This machine learning model is responsible for assigning labels of sentiments (positive, negative, etc.) to inputs. This part of the system requires the most effort, as building an effective learning model *(i)* needs high quality training data, and (*ii*) proper classifier selection and other important parameters like the n-gram model, and feature concatenation. These have to be extensively tested to achieve a promising result.

- The outputs of the system are aimed at providing a statistical view on data in order to process, potentially, large samples and be able to examine generalised results allowing the drawing of conclusions, which is the ultimate goal and principle of the particular field.

4. Sent-Affect Design and Implementation

In the present chapter, the learning models developed will be presented in a structured manner that follows the steps of creating a sentiment analysis model elaborated in section 3.2. During the lifetime of this project, two distinct learning models were developed, namely, a film review model and a twitter model. Both models will be analysed, regarding processes like training, testing and generally the parameters considered. Following, this chapter is arranged to analyse the machine learning development thoroughly, and at the same time discuss and present the software application created in support of the SA engine.

Regarding the ML development, a presentation about the acquisition of assembled data to train and test the two machine learning models is conducted. Moreover, the annotation process, i.e., formal annotation (in XML) that was necessary for the engineering tool chosen to understand and interpret data. And the complete development procedure that was followed to train and evaluate such a supervised learning model.

Concerning the software component of the project, the Java application developed is presented in terms of general system design and implementation details and decisions.

4.1 Data Collection

This section covers and presents data collection process for both the film review and twitter model intended for sentiment analysis. Further, sources and details about the datasets gathered are discussed.

i. Film Review Model

The core training dataset for this model was assembled from the available data compiled and published by Pang/Lee [22] for film reviews. The set consists of two

thousand manually annotated reviews of various films. These reviews are split into positive and negative. See table 8.1 for the exact dataset numbers.

Type	Positive	Negative	Total
Film review	1000	1000	2000

Table 4.1. Training Data - Film Review Model

In addition to the Pang/Lee dataset, a significant amount of film reviews were amassed with the purpose of testing the model and evaluating its performance. The dataset, made available by [48], contains 50,000 film reviews (half for training and half for testing). However, only 4,000 of those were used for testing purposes. The test sets were split into two groups; 1,000 and 3,000; see tables 4.2 and 4.3 for the test sample details. This was concluded in contemplation of stretching the range and ability of the classifiers in different sets. Consequently, we are able to provide a more generalised result pertaining to its accuracy and overall performance.

Type	Positive	Negative	Total
Film review	500	500	1000

Table 4.2. Test Data I - Film Review Model

Type	Positive	Negative	Total
Film review	1500	1500	3000

Table 4.3. Test Data II - Film Review Model

Evidently, this model should deal with two labels (binary classification), that is, positive and negative. Ergo, it is assumed that a review per se, is either positive or negative and nothing in between. Providing a neutral class would be quite difficult due to the fact that neutral reviews can be cumbersome to find and somewhat ambiguous to define.

In other words, the assumption is that a film review expresses some sentiment or opinion about a film. On the other hand, neutrality can be explained as a mixture of balanced positive and negative traits. However, such fine-grained analysis is exceedingly complicated for this task and may also affect the classification of the majority of reviews that are one-sided. As follows, no effort to understand such reviews was appointed. Next, a sample positive review taken from the test sets [48] dataset:

"*This film isn't just about a school shooting, in fact its never even seen. But that just adds to the power this film has. Its about people and how they deal with tragedy. I know it was shown to the students who survived the Columbine shooting and it provided a sense of closure for a lot of them. The acting is superb. All three main actors (Busy Phillips, Erika Christensen and Victor Garber) are excellent in their roles...I highly recommend this film to anyone. Its one of those films that makes you talk about it after you see it. It provokes discussion of not only school shootings but of human emotions and reactions to all forms of tragedy. It is a tear-jerker but it is well worth it and one i will watch time and time again*"

ii. Twitter Model

Sentiment analysis for social media and social networking sites requires a plethora of training data to support the learning algorithm because posts in such platforms contain noise and can be extremely ambiguous [49]. Short texts with an informal character present difficulties in sentiment classification and such samples are quite common in social networking platforms. All of these mentioned issues and considerations have been discussed in the background coverage of the topic.

Regarding the collected data, it originates from Twitter, YouTube, myspace, and custom-built subjective sentences that express opinion. The majority of the data are real uploaded comments or posts. Furthermore, generally in social media models, it is important to consider the neutral class. Numerous posts can be described as non-subjective, that is, they do not express any emotion. For instance, questions or statement of facts. This is quite common on Twitter; accounts belonging to news channels for instance usually post neutral statements or a major headline and proceed

afterwards to encourage the reader to visit the website for the full story. In contrast to the film review model, a similar assumption about a de facto sentiment cannot be consciously made. As a logical consequence, the neutral class should be taken into consideration as a label in the final delivery of results, but that does not imply that it should be a separate class in the learning process of the model. Subsequently, we shall review the sources of the data collected.

First, from Twitter, the STS-Gold dataset by [49] contains 2034 tweets about various topics from which 632 are positive and 1402 are negative. It is considered a set of decent quality, but on the other hand it is imbalanced, i.e., negative instances outnumber the positive ones substantially (2.22 times more negative instances). Moreover, SentiStrength [50], [51], [52] provides a manually annotated twitter dataset; however, this dataset is annotated with sentiment scales of 1-5 for both positive and negative. Hence, the instances translated to discrete label classification problems (positive, negative, and neutral). The scheme used, as proposed by [49], is that an instance: (*i*) is neutral if the positive value equals the negative value, (*ii*) is positive if the positive value is 1.5 times greater than the negative one, and (*iii*) negative otherwise. After this processing, the dataset results in 1953 neutral, 1258 positive, and 1038 negative tweets. SentiStrength also has similarly labelled datasets from YouTube and myspace that were transformed into discrete labelling. Table 4.4 below summarises the collected data for the Twitter model.

Source	**Positive**	**Negative**	**Neutral**
Twitter	1884	2438	1954
YouTube	1469	963	975
Myspace	635	197	207
Custom	3995	3091	0
Total	7983	6689	3136

Table 4.4. Data collected for the twitter model

It is vital to point that table 4.4 displays the total amount of data collected through research, but does signify that all of it was used for the training and testing of the machine learning model. Overall, the instances found in the majority of these datasets were quite ambiguous, short, and even misclassified (subjectively) in several cases.

The compiled data, as a collective dataset, is also quite imbalanced having more positive than negative examples and too few neutral ones. For these reasons it cannot be used as a training dataset itself. As a result, it is necessary to start with a dataset as a basis and continue thereafter. The STS-Gold dataset is the starting point for this model, as it is well regarded in some research on twitter dataset evaluations and contains examples of good quality [49]. Even though it is unequal, as mentioned previously. Afterwards, the STS-Gold was expanded to include more instances, particularly of positive label in order to balance the two classes to avoid any bias towards a class. A model for each dataset was developed and they are compared in section 5. The resulting dataset, i.e., STS-Gold Balanced, consists of 1229 positive and 1411 negative posts from Twitter.

Dataset	**Positive**	**Negative**	**Total**
STS-Gold	632	1402	2034
STS-Gold Balanced	1229	1411	2632

Table 4.5. Training data for twitter model

The first thing one would notice is the absence of neutral examples, and they would be right to do so. So, regarding the neutral class, the classification task will not include directly the neutral class as a label for the classifier. This means that we are still discussing another binary classification problem of positive or negative evaluations. Classification using neutral examples can cause issues and miss-classifications, especially when there are no reliable neutral examples available in order to balance the learning process. In short, the data shortage and its low quality did not allow for the insertion of a new label in the learning phase. Therefore, the problem will be handled with confidence thresholds, which is covered at a later section of this paper.

Further, in social networking sites such as Twitter, which is the model's target base, the use of emoticons (facial expression representations using symbols like ':D' and ':p') is significantly common nowadays and the majority of which express definitively emotion or sentiment. Thus, the ability to understand smiley faces and other emoticons would be desirable in this system because it may improve

performance in multiple occasions [55]. Also, it is quite facile to produce labelled test samples with emoticons. Hence, it was decided to include emoticon data in the training sets to train the model to classify confidently various emoticons that may be encountered in Internet posts. Table 8.6 summarises the emoticons used in the samples.

Emoticon	Polarity
:), :D, :p, :-), =), ;), c:	Positive
:(, :-(, =(, :'(, :@, :c, ='(	Negative

Table 4.6. Emoticon data

The above emoticons were integrated into the STS-Gold Balanced dataset as an external feature. A number of new text documents were created with these emoticons mixed with text and some were placed as a new instance itself. This proved ideal as the classifier was quite successful on all known emoticons presented to it.

4.2 Data Annotation

The data annotation section refers to the computerised annotation that is utilised for the purpose of presenting the semantics and the attributes of the data to the NLP toolkit (GATE). This pre-processing stage is necessary before the data can be used. The text instances amassed are labelled as either positive, negative, or neutral, but this is not sufficient for the machine learning framework to read and understand what exactly this means. GATE uses an annotation system for text documents, where selected text can be assigned a custom annotation, along with a set of features that describe the information one wants to depict. In this way, documents can be inserted into GATE and manually annotated from the GUI application [54]. In the next figure 4.1, a text document has been uploaded in GATE and the annotation details are visible; this is how GATE defines a document.

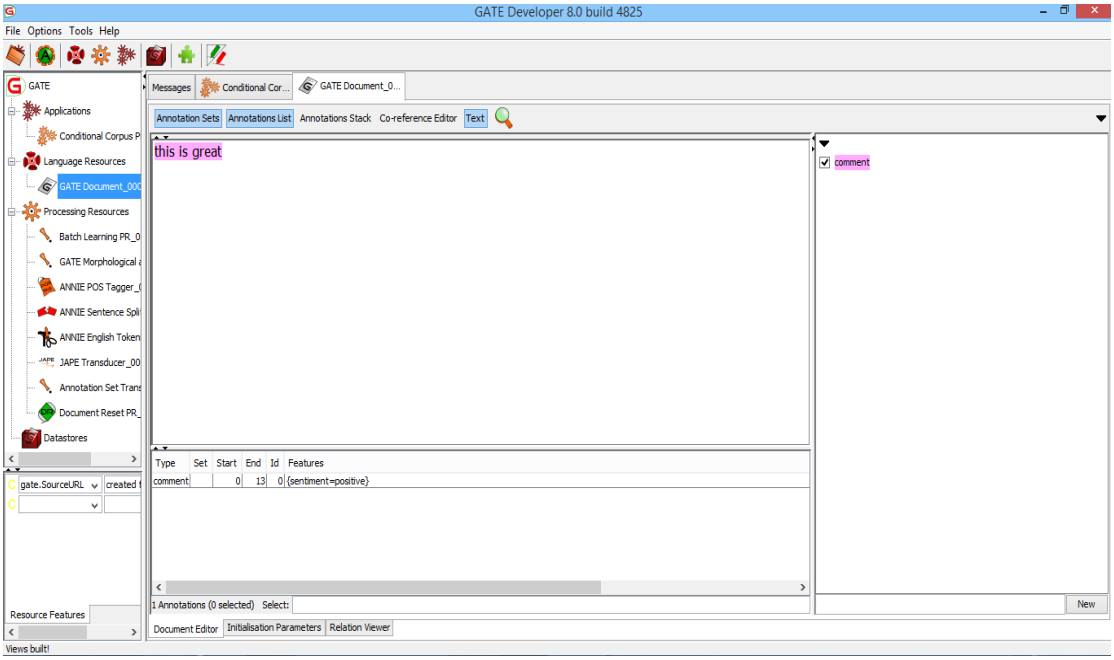


Figure 4.1. Data annotation in GATE

In the above image, there is an open document named *test1* which contains the text *"this is great"*. A manual annotation type 'comment' was created and the sentence was marked as such. The comment represents the instance for which one is interested in certain features; the feature in the case of sentiment analysis is the sentiment and its respective value. Below the white space of the document in the figure, the list of annotations appear. The particular instance is labelled correctly as positive by the GATE user (under features: sentiment = positive) and that annotation corresponds to the text marked in pink only.

This type of annotation system is used by GATE and it is quite efficient and convenient because a document can have multiple annotation sets and distinct classes of instances, therefore, it can be configured to search for specific annotation types. In this project, every instance is of type *comment* and the focal point is the value of the sentiment feature. Moreover, this relation and specification between features and annotations of a GATE document is represented by an XML schema that defines all annotations found in a given document, its features, and locations in the document. Thereafter, any document can be saved in the XML format by GATE. The following XML code shows the annotation of the above example document.

```
<?xml version="1.0" encoding="UTF-8"?>
<GateDocument version="3">
  <GateDocumentFeatures>
    <Feature>
      <Name className="java.lang.String">gate.SourceURL</Name>
      <Value className="java.lang.String" />
    </Feature>
    <Feature>
      <Name className="java.lang.String">MimeType</Name>
      <Value className="java.lang.String">text/plain</Value>
    </Feature>
    <Feature>
      <Name className="java.lang.String">docNewLineType</Name>
      <Value className="java.lang.String">LF</Value>
    </Feature>
  </GateDocumentFeatures>
  <TextWithNodes>
    <Node id="0" />
      this is great
    <Node id="13" />
  </TextWithNodes>
  <AnnotationSet Name="Key">
    <Annotation EndNode="13" Id="1" StartNode="0" Type="comment">
      <Feature>
        <Name className="java.lang.String">sentiment</Name>
        <Value className="java.lang.String">positive</Value>
      </Feature>
    </Annotation>
  </AnnotationSet>
</GateDocument>
```

Listing 4.1. XML schema for GATE document

Besides the GATE document features that are not very relevant, the description is represented further with the actual text, followed by the annotation set, the annotation and its feature of sentiment that has a value of positive. The annotation set can contain multiple annotations, as elaborated earlier. The node values indicate the position in text in terms of character position in the string. These are the types of instances the learning algorithm will be reading and analysing. In a simplistic explanation, the classifier is focusing on the words and associates them with the corresponding label and builds a statistical model around them. Additionally, the attributes are fully compatible with the Java language, for instance, some of the attributes contained in a class are java.lang.String.

Following, the task at hand is to annotate accordingly thousands of text documents to be used for the training and testing of the machine learning model.

Consequently, manual annotation with GATE would be an awfully lengthy and cumbersome process. Hence, the annotation should be automated, not only for the pre-processing stage, but also for the application phase, when the model is used on real data. Because even if the sentiment of the document is not known, the text should be structured in the depicted manner by the schema. For this reason, a small Java application was developed to perform this XML annotation on the collected data. The requirements of this small library were to be able to read a document (txt, CSV, etc.) that would contain the data as text, annotate them with the mentioned XML format and write it in a new xml file. This would result in a usable GATE document that can be understood by the framework. The core functions of this library are presented and discussed in the next page. The language of implementation is Java, the environment used is JDK 1.8, and the XML processing was done using the org.w3c.dom library of the standard edition in the general Java API.

```
public final class XMLCreatorGATE {
    public static Document annotateTextForGATE(String content, String
featureName, String value) {
        Document doc = null;
        try {
            DocumentBuilderFactory docFactory = DocumentBuilderFactory
                    .newInstance();
            DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
            doc = docBuilder.newDocument();
              ...
              }
        return doc;
}
```

```
public static void writeDocToFile(Document doc, File file) {
        try {
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(doc);
            StreamResult result = new StreamResult(file);
            transformer.transform(source, result);
        } catch (TransformerException tfe) {
            tfe.printStackTrace();
        }
}
```

```
  public static String readFile(String path, Charset encoding)
                                    throws IOException {
          byte[] encoded = Files.readAllBytes(Paths.get(path));
          return new String(encoded, encoding);
  }
```

Listing 4.2. Data annotation library in Java

The annotate method builds the appropriate XML schema for any text content inputs, along with a feature and its value, for example {(sentiment, positive)}. The whole body of the method is not shown due to its length, but this is the main function used for annotation. It creates the XML nodes according to the schema shown earlier to transform the document into a GATE document. After the given document is annotated, it should be written to a file as an xml. Thus, the second function uses the transformer to write the DOM source into a file and directory so that we can use the GATE document. The last function is a utility that reads a file using Java SE7 libraries that allow for concise and readable code.

Having the core functions, we can now access the files of the datasets and perform the annotation. Some are in txt files others in CSV, but there is no significant difference, CSV files are just text files with special delimiters. In this way, over 5,000 instances were annotated and a large portion of them eventually used for the learning and testing stages. The wrapper method of the annotation of twitter posts and such is presented next.

```
public class DatasetAnnotationApp {
    public static final String FEATURE = "sentiment";
    public static void main(String... args) {
        DatasetReaderCSV tweetReader = new DatasetReaderCSV();
        List<Post> tweets = tweetReader.readPosts(";",
"C:\\Users\\superhands\\Downloads\\trainingandtestdata\\testdata.manual.2009
.06.14.csv");
        int n = 0;
        for (Post twt : tweets) {
            String xmlFileName = "subj " + twt.getPolarity().toString() + n;
            File destinationFolder = new File(
                    "C:\\Users\\superhands\\Desktop\\Machine Learning
Models\\Social Media Model\\sentences\\"
                            + xmlFileName + ".xml" );
            Document xmlFile =
XMLCreatorGATE.annotateTextForGATE(twt.getContent(),
                                    FEATURE, twt.getPolarity().toString());
            XMLCreatorGATE.writeDocToFile(xmlFile, destinationFolder);
            n++;
```

```
            }
        }
}
```

Listing 4.3. Dataset annotation application

The dataset reader CSV is another part of this library and for more details on that class and the rest of the library see appendix A. Through that class the instances are read and placed in a list of posts that contain the text and the polarity. Subsequently, each post is annotated with the annotation library shown earlier and then written to a directory. As a result, in a matter of nanoseconds all instances are GATE documents that can be used in the application.

A similar procedure was followed for the 2,000 training film reviews and 4,000 for testing. The results were produced efficiently and the automation assisted the process substantially. As this point, data have been collected for both models and have been successfully annotated and processed for the tool. The next step is to perform training and evaluation.

4.3 Model Training

Training a machine learning model for sentiment analysis and NLP tasks in general requires multiple processing resources to apply on the text. In GATE, this series of operations is called a pipeline, i.e., the documents are passed through the pipeline of various processing resources and operations. GATE has ML features with its Batch Learning PR (learning plugin) and it is the newer implementation that replaced the old Machine Learning PR. The former and newer version is designed to target text classification, named entity recognition and provides LibSVM support, including the fast and powerful PAUM algorithm that works with margins in the feature space [54], [56]. There is also an interface for Weka support with naïve Bayes classifier, decision trees, and kNN, but they were not used.

The first requirement to start using the learning features of GATE is to create the machine learning configuration, which is a file that contains a specification of the vital settings and options the algorithm is to follow. We will not cover all of the settings because they are many, but some important ones that are connected to the process that is described in this paper are explained. The ML config is an XML file that assigns attributes and values to considerations regarding the learning such as the algorithm.

```
<?xml version="1.0"?>
<ML-CONFIG>
  <VERBOSITY level="1"/>
  <SURROUND value="false"/>
  <PARAMETER name="thresholdProbabilityClassification"
         value="0.6"/>
  <multiClassification2Binary method="one-vs-others"/>
  <EVALUATION method="kfold"
          runs="10"
          ratio="0.66" />
  <ENGINE nickname="SVM"
      implementationName="SVMLibSvmJava"
      options=" -s 0 -t 1 -c 1 "/>
  <DATASET>
    <INSTANCE-TYPE>comment</INSTANCE-TYPE>
    <NGRAM>
      <NAME>ngram</NAME>
      <NUMBER>1</NUMBER>
      <CONSNUM>1</CONSNUM>
      <CONS-1>
        <TYPE>Token</TYPE>
        <FEATURE>root</FEATURE>
      </CONS-1>
    </NGRAM>
    <ATTRIBUTE>
      <NAME>Class</NAME>
      <SEMTYPE>NOMINAL</SEMTYPE>
      <TYPE>comment</TYPE>
      <FEATURE>sentiment</FEATURE>
      <POSITION>0</POSITION>
      <CLASS/>
    </ATTRIBUTE>
  </DATASET>
</ML-CONFIG>
```

Listing 4.4. ML configuration file

Listing 4.4 shows the ML configuration file used in the twitter learning model. The extremely relevant and important options are explained next.

- *ThresdholdProbabilityClassification*: the classifier produces ultimately a probability value for the confidence that a particular instance belongs to a given class. The threshold allows the ML developer to set a minimum probability for the classifier to assign a label. In this case, the value is set to 0.6, this means that the classifier will assign a label to an instance if and only if it is at least 60% confident on the assignment.

- *Evaluation*: is a means of stretching and evaluating the ML model and it is a feature of GATE. The method of doing this is called cross validation k-fold where k is the number of folds. This process splits the dataset into k parts and then uses training data to train and apply on the smaller test data k times. The k partitions are of equal size and once every fold is evaluated, the results are averaged from each fold to provide an overview on the classifier performance.

- *Engine*: the engine element of the configuration file specifies the ML algorithm to use for the learning sessions. The implementation name is the algorithm and it is followed by some options. In this file, the classifier chosen is a Support Vector Machine implementation. The options of the SVM are: -s 0 states that the classification is binary, -t 1 is the kernel function and 1 signifies the polynomial kernel: $K(X_i, X_j) = (\gamma X_i \cdot X_j + C)^d$ and the default is cubic d = 3. (only linear and polynomial are supported). And finally, -c 1 is a cost parameter that makes some margin adjustments to generalise results by misclassifications.

- *Instance-Type*: is the custom named instance that we have named *comment* as seen in the annotation section. The *NGRAM* refers to the bag-of-words model, which defines how lengthy the sequences of words in the text are going to be. For instance, if n = 1 (unigram) every word is taken independently, but in the bigram model, the algorithm takes words by pairs of two. Further, *attribute* contains the features we are looking for regarding the particular instance (the comment). The feature is the sentiment and the values (the labels) are inferred by the algorithm when all training documents have been processed.

After setting the configuration file for the learning PR, a GATE corpus application should be assembled. This pipeline consists of various processing resources the developer chooses in order to extract certain linguistic features from the documents. These are text engineering resources and the pipeline created uses recommended features of GATE, supplemented with some minor adjustments. The pipeline is presented in the next figure.

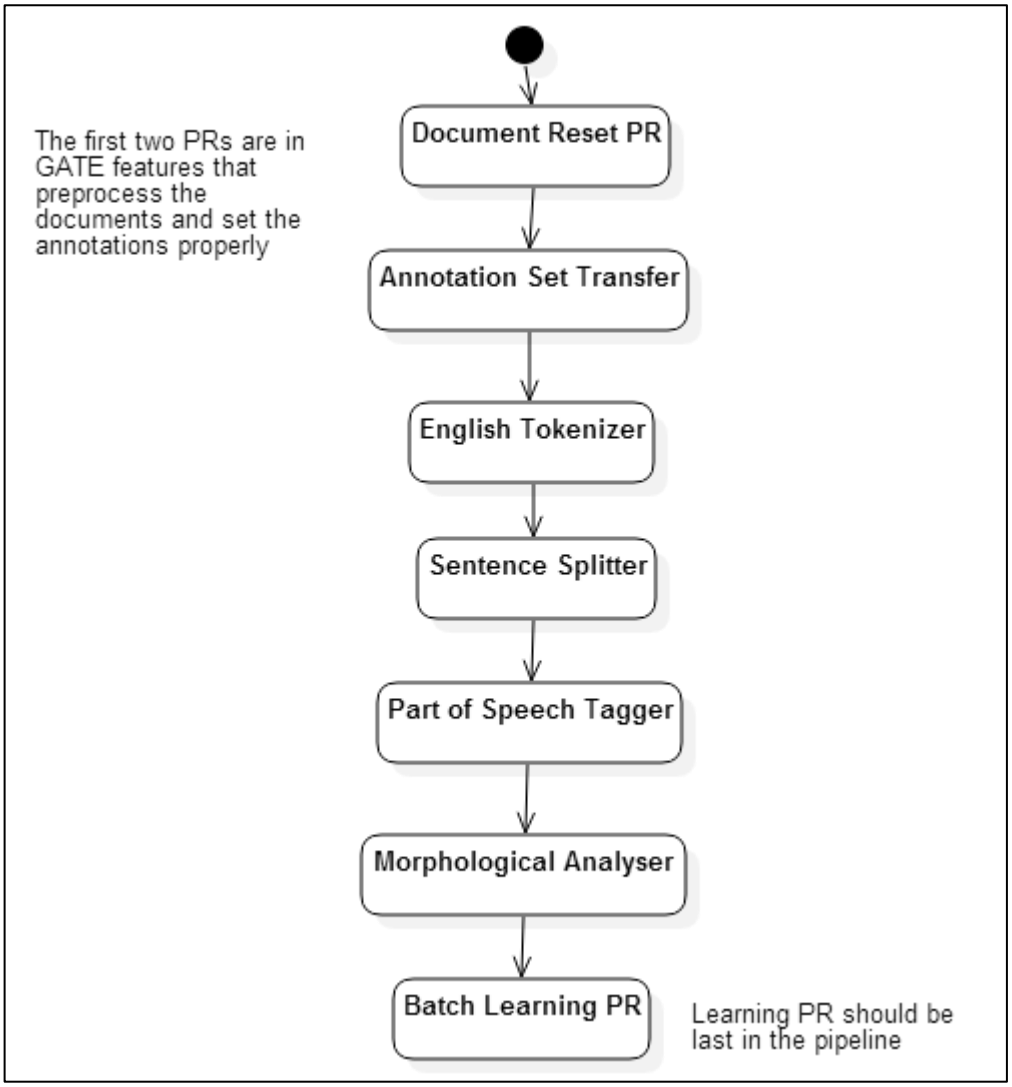


Figure 4.2. NLP pipeline

The first two resources are GATE internal features that allow the developer to set various specifications in order to view certain document characteristics and read the appropriate annotations of documents. The English tokenizer splits the text stream into words, phrases, and symbols and these resulting tokens are input further into the pipeline for processing. Subsequently, the sentence splitter is tasked with segmenting the text into distinct sentences. This is important, as sentences can be viewed as groups later on and it is required before the POS tagger can be used. Afterwards the document passes through the POS-tagger (part-of-speech) that simply tries to identify to which category each word belongs in the English grammar, namely, verbs, adjectives, nouns etc. The morphological analysis is a form of linguistic analysis that attempts to discover morphemes and other features in the language. Finally, the Batch Learning PR describes the ML algorithm that given the set of documents outputs a list of linguistic features, labels, and feature vectors. The focal algorithm as stated is the SVM. The ML PR requires the linguistic features that the previous processing resources produce because they are necessary for the learning [54].

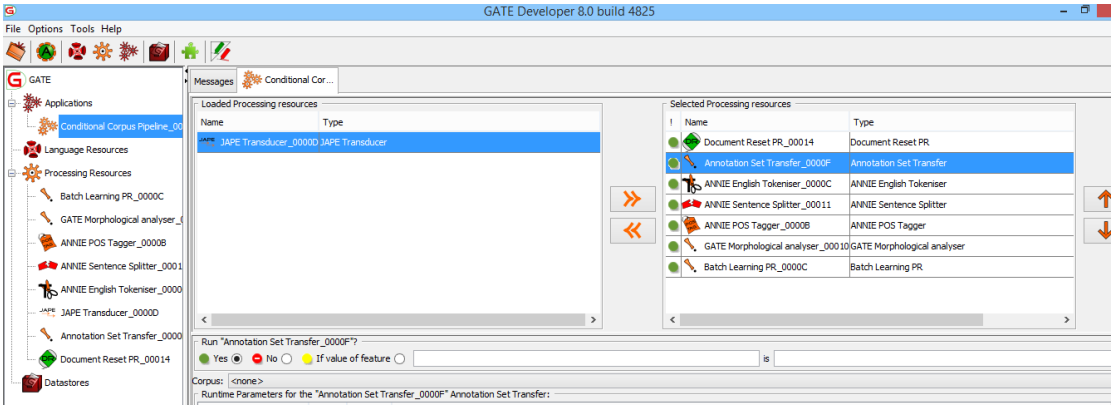


Figure 4.3. GATE application pipeline

The above figure 4.3 shows the pipeline of PRs in GATE. This is the pipeline used for training and evaluation, but not for testing (application). It is a conditional corpus application in GATE, and the order of processing is important. GATE runs each document, processes it and then moves on to the next one, but the learning PR receives a set of documents, therefore, it must be placed in the last position of the pipeline.

The GATE application is ready and now the language resources, that is, the documents can be loaded to GATE to start the training of the classifier. We create a corpus and populate it from the directory where the XML annotated documents are placed.

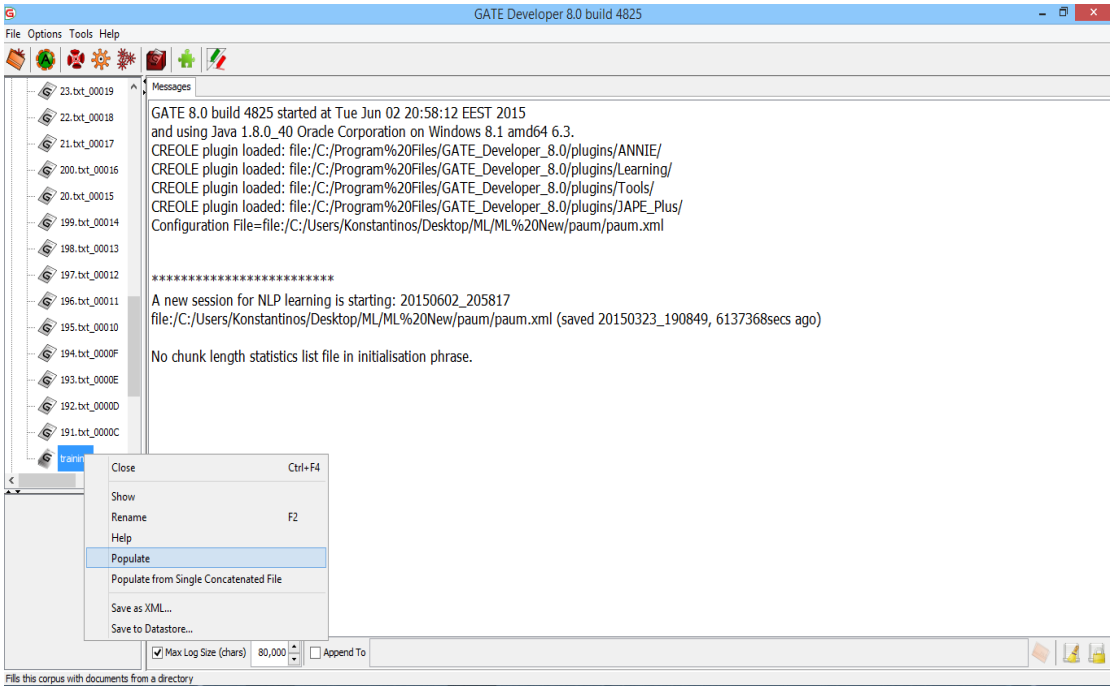


Figure 4.4. Populate GATE corpus

All documents are loaded in GATE from the provided folder, where they can be viewed and edited. These documents are loaded all together, but this should be avoided if there are too many documents or they are huge. This would cause memory issues on runtime. There is a method to solve this problem through GATE features and it is explained later. Continuing, the application can be started to train the classifier on the *training* corpus that was populated with the training dataset.

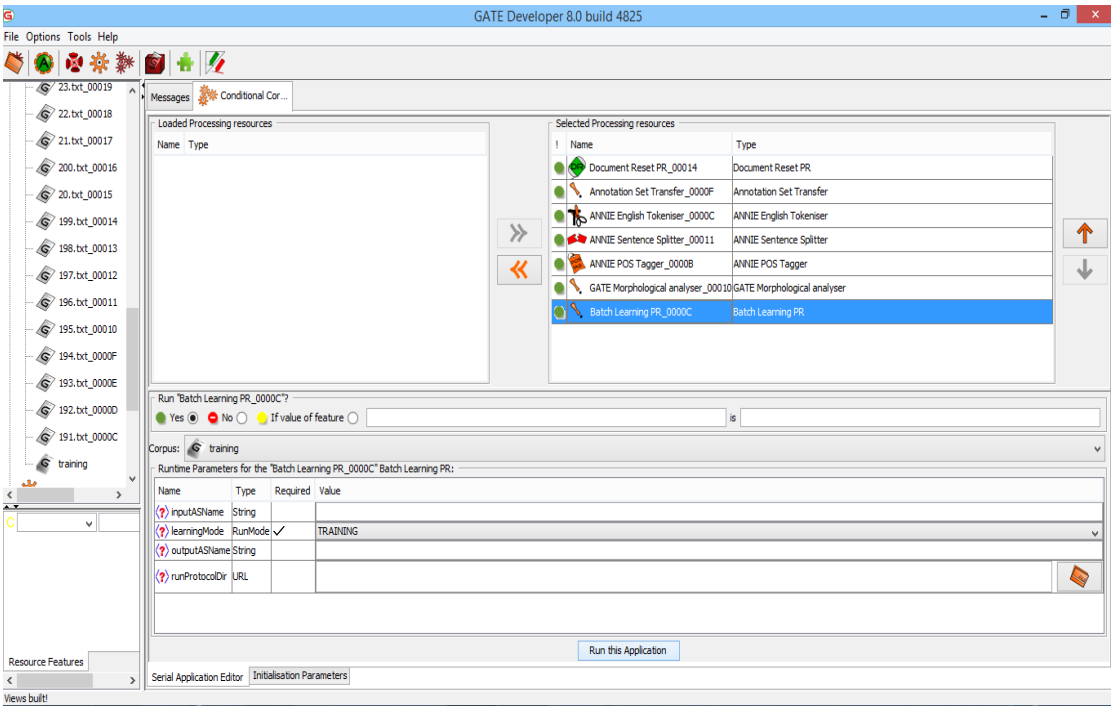


Figure 4.5. Training classifier in GATE

By running the application the process starts and each document is processed and presented to the ML classifier at the end, at which point, it constructs the model based on the training data. The procedure's time depends on the number of documents, the size on the documents, the algorithm used, and the n-gram model. The polynomial SVM is somewhat more time consuming than the linear, but the n-gram model is a larger factor. The files of the model are saved next to the ML config file and the hence the model can be reused by using the same config file to start a new learning session.

After the training is completed, a few quick tests can be made before moving to the evaluation phase. The pipeline setting for an application of the model has some differences. The testing corpus is labelled, however we want the classifier to predict a label for the test samples instead of knowing it, so we use a transducer of GATE in order to effectively hide an annotation. The goal is treat these documents as instances without the sentiment feature to have the classifier assign a label. The following JAPE transducer.

```
Phase:  CopyCommentSpan
Input:  comment
Options: control = all
Rule: comment
( ({comment}):span )
-->
:span.comment = {}
```

The transducer is placed in the pipeline after the disabled annotation set transfer and the learning mode is switched from training to application. The testing data should also be uploaded as a corpus under name *testing*. Thence, the application can run on the corpus; it will annotate the documents and provide a quality assurance analysis that displays the classifier's performance. Moreover, a new annotation is created called *Output* that will apply for the comment instances. The annotation is exactly the same, i.e., a sentiment, value pair that represents the classifier's prediction. The probability of confidence is included too.

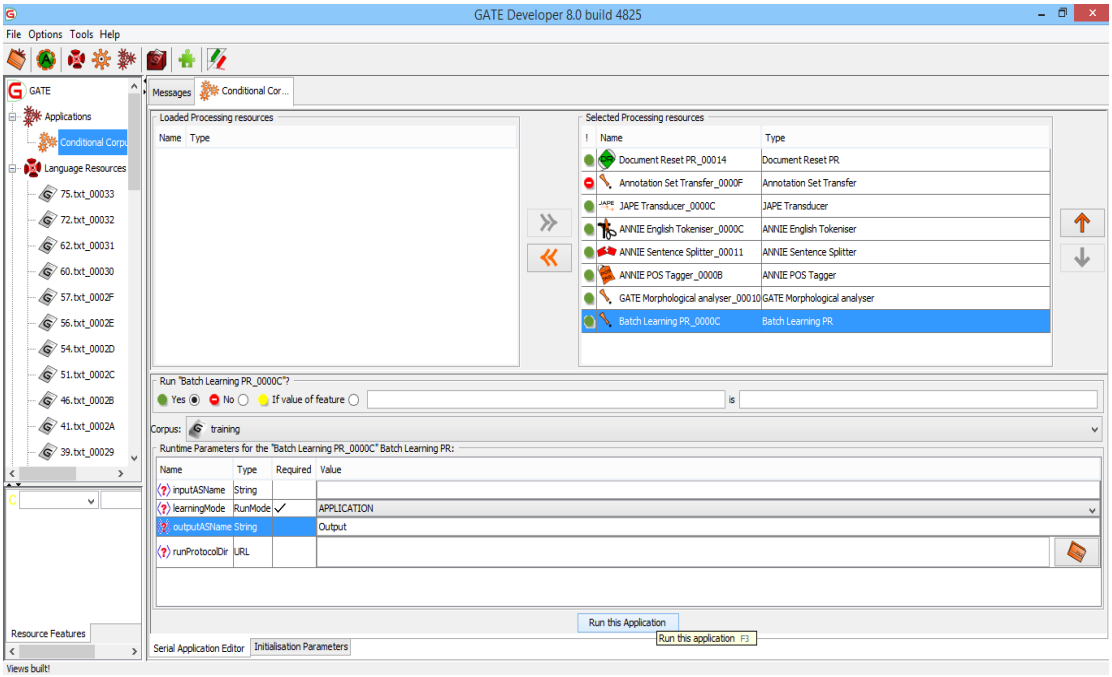


Figure 4.6. Test application pipeline in GATE

The output annotations are compared with the original annotation and the average precision is calculated by GATE. Looking at the corpus quality assurance tab of GATE, the annotations one wants to compare are selected and the results are displayed. The testing corpus comprised of 220 short instances from Twitter and YouTube that were extremely hard and peculiar for the classifier to understand. Spelling mistakes, acronyms, and abbreviations described these instances; however, the classifier did unexpectedly well.

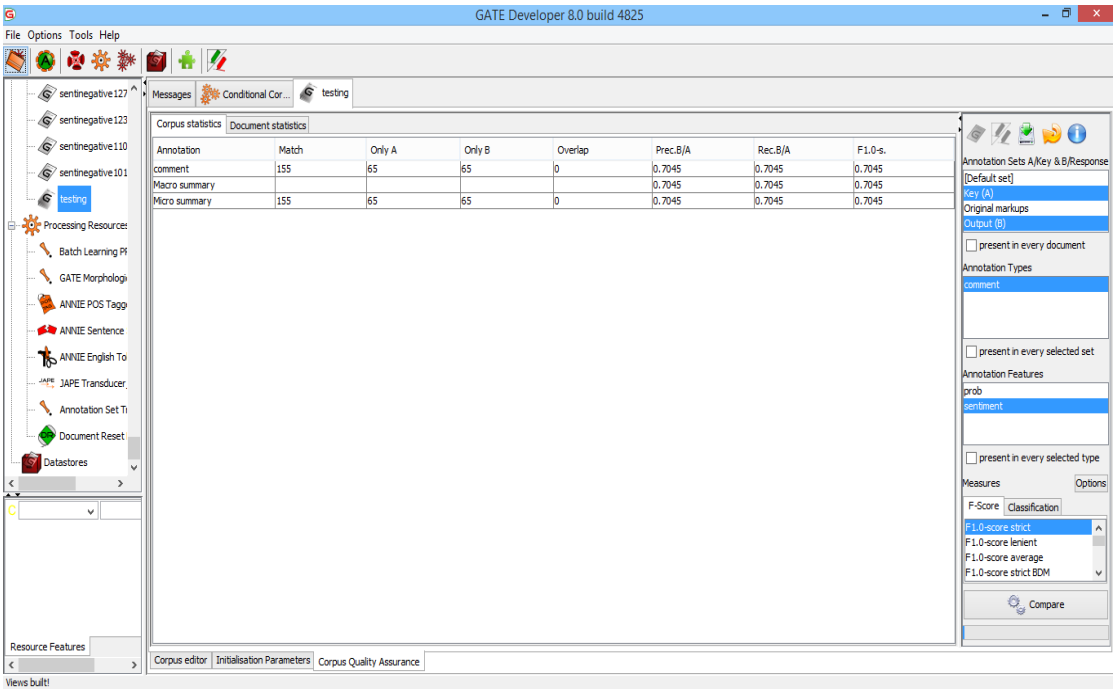


Figure 4.7. Corpus quality assurance GATE

The precision is under Prec.B/A along with Recall and F1 score. They have the same value because every instance in the dataset was classified. The overall accuracy is this number: 70.45% and it is fine for these rough examples of social network posts. As seen in the figure, 155/220 instances are a match for the type *comment*. To show some examples of instances:

"@ShiedhaO well damn! Renee still aint playin, is she?! and neither is Jack!! @LikasParody" **Negative**

"That's just too adorable! Congrats!" **Positive**

Another dataset for testing was evaluated by the model, this time it consisted of 2,000 instances that were somewhat more straightforward regarding language and meaning. The classifier scored 88.25% and all instances were labelled.

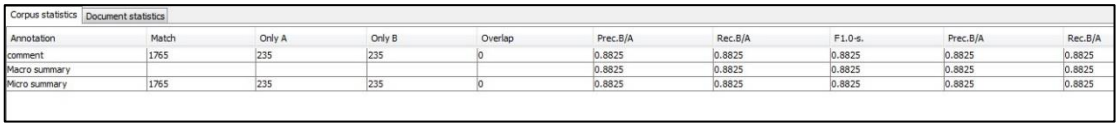


Figure 4.8. Test results GATE

After the testing, each document can be further inspected in order to check which instances were misclassified. In the following figure 4.9 all the annotations can be seen. The original annotation was a comment annotation under the set Key; the others were generated from the processing resources such as the tokenizer and POS tagger. For example, the Token annotation in blue, will indicate the tokens this text document was divided in. In the annotations list below, the old annotation of comment is compared with the new annotation of comment that belongs to the *Output* set. This is the label that the classifier produced. The labels match, both are negative and the classifier is certain of this prediction.

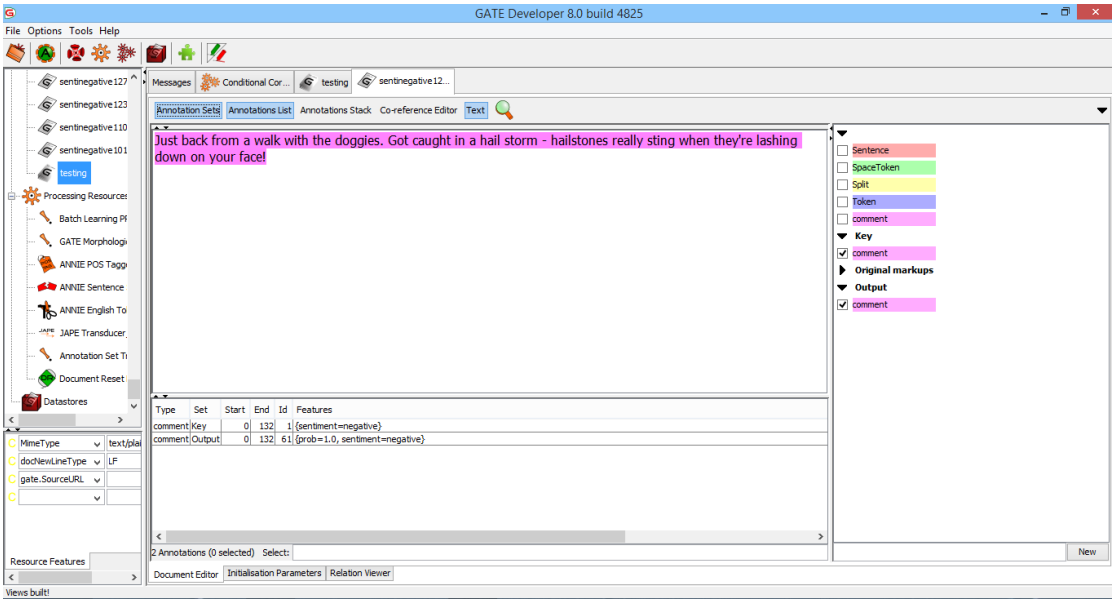


Figure 4.9. Annotations generated from tests

Upon completing some preliminary tests on the machine learning model, the final evaluation can be initiated. Cross-validation will be performed on the training dataset, and the technique, as explained in the ML configuration file parameters, is k-fold validation. The STS-Gold Balanced training dataset for the Twitter model has 2,632 documents. The evaluation will be 10-fold, meaning the corpus is split in 10 equal parts and it trains and tests 10 times on separate sets. Cross-validation is a standard technique in machine learning because the classifier is stretched and the results of this evaluation are more important than the isolated tests per se. The batch learning PR of GATE has a learning mode of evaluation that does that. It runs the 10 fold validation and averages the results on the screen. The results can be observed below.

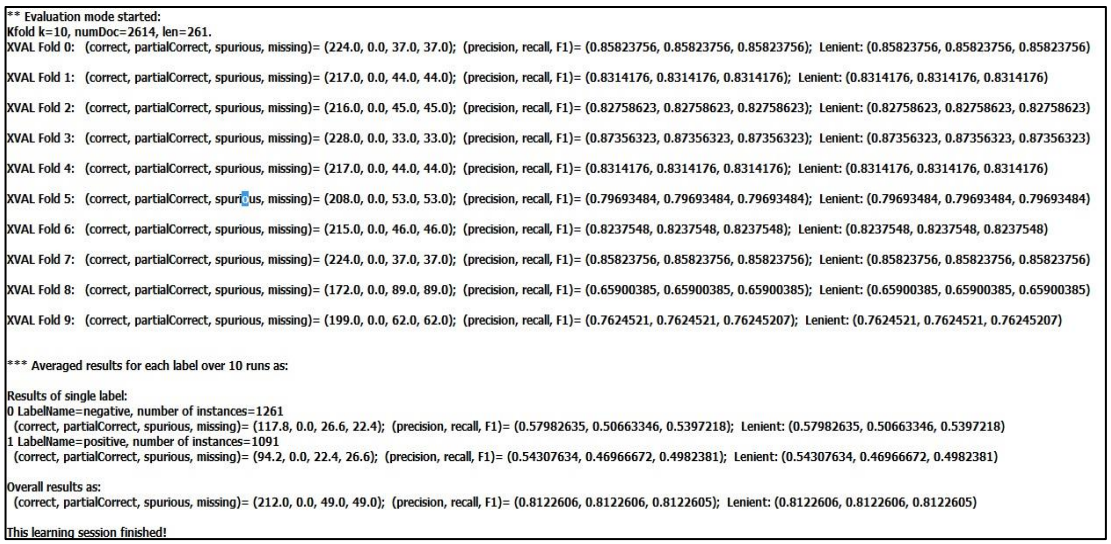


Figure 4.10. 10-fold cross-validation results

Figure 4.10 displays the precision, recall, and F1 scores for each fold and further down shows the average results (*overall results as).* The percentage amounts to 81.22%. This result is extremely promising and exciting, if one takes into consideration the difficulties of social network posts analysis. The trained model files and the ML file are generated and can be reused. The GATE corpus application can be exported as a GATE file to be used later in the programmatic integration of the model.

An identical procedure was followed for the film review model as well, the only difference was the ML config that had different parameter options and the appropriate film datasets were used. In chapter 5, a detailed comparison and analysis of the learning results is conducted.

4.4 Software Development Environment

This section constitutes a transition from the machine learning development phase to the software development of the supporting system that wraps the functionalities intended. The languages, tools, frameworks, etc. used in the development process of the application are presented here.

To begin with, the software is written in Java; it offers direct integration with the GATE API and Java is a general purpose language that can handle most tasks well. The GUI is built with JavaFX, the new user interface framework of Java for rich internet applications that aims to substitute Swing. The standard version 8 is used along with several utility frameworks and web service APIs. The complete list is the following:

- Language: Java
- Environment: JDK 1.8 update 45
- Frameworks: JavaFX 8, GATE 8
- Tools: JavaFX Scene Builder 2.0
- APIs: Apache HTTP Components 4.4.1, Apache Commons IO 2.4, JSON Simple 1.1.1, Alchemy API (web service)

Following this information, the overall design of the software is presented with detailed diagrams such as package and class diagram. The interface between GATE and the application is discussed and how the documents are passed from the UI to GATE and back. The application effectively, uses the annotation library presented earlier to annotate user input text documents, save them, and through the interface run the GATE pipeline developed on these documents and subsequently, extract the classifier's predictions.

4.5 Application Design

The application is structured and designed following a classic object oriented approach with entity classes that represent the business logic, utility classes, and the GATE interface to control and operate the learning model from within the application. The business logic is minimal. The only entity objects are the comments that contain the text and the label (positive, negative, and neutral). The GUI is specified in XML format using the JavaFX FXML specification and was created with a design tool, namely, the JavaFX scene builder 2.0. The package structure of the application will assist in understanding the underlying design of the system.

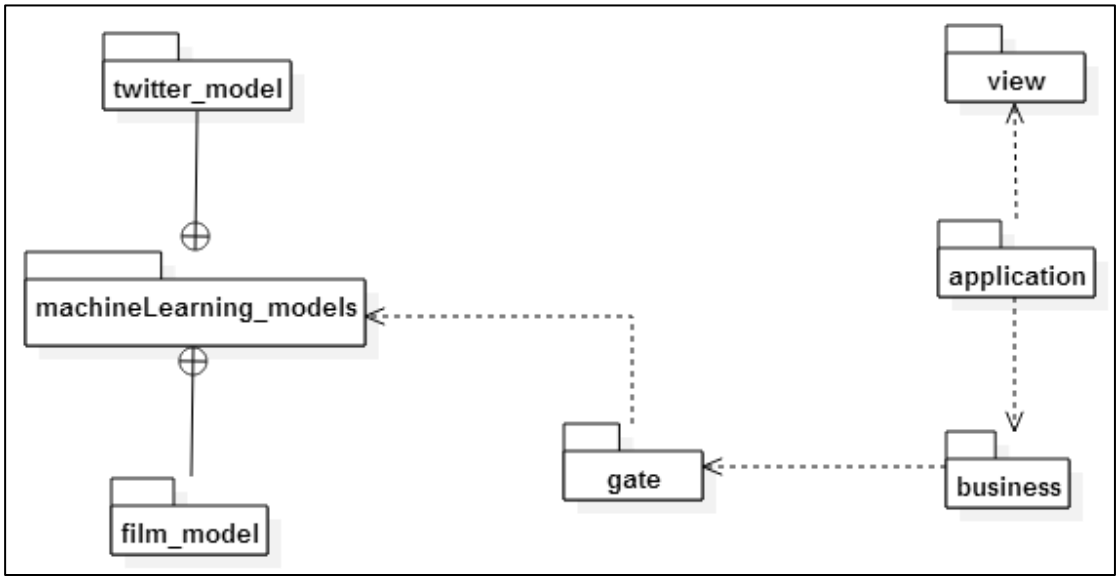


Figure 4.11. Application package diagram

The machine learning models package contains the model files of both the film and twitter model, and their GATE application files that are loaded from the API. These files are accessed by the classes in package gate, where the connection between the gate API and this application is made. The business package contains the logic of the application and the main class that wraps the functionalities of the relevant classes and interfaces in the GUI, following a mediator design pattern. The application package has the Java classes that control the user interface, while the user interface XML files are separately kept in the view package. Next, a dive into the contents of these packages is done, through a complete class diagram and design explanations.

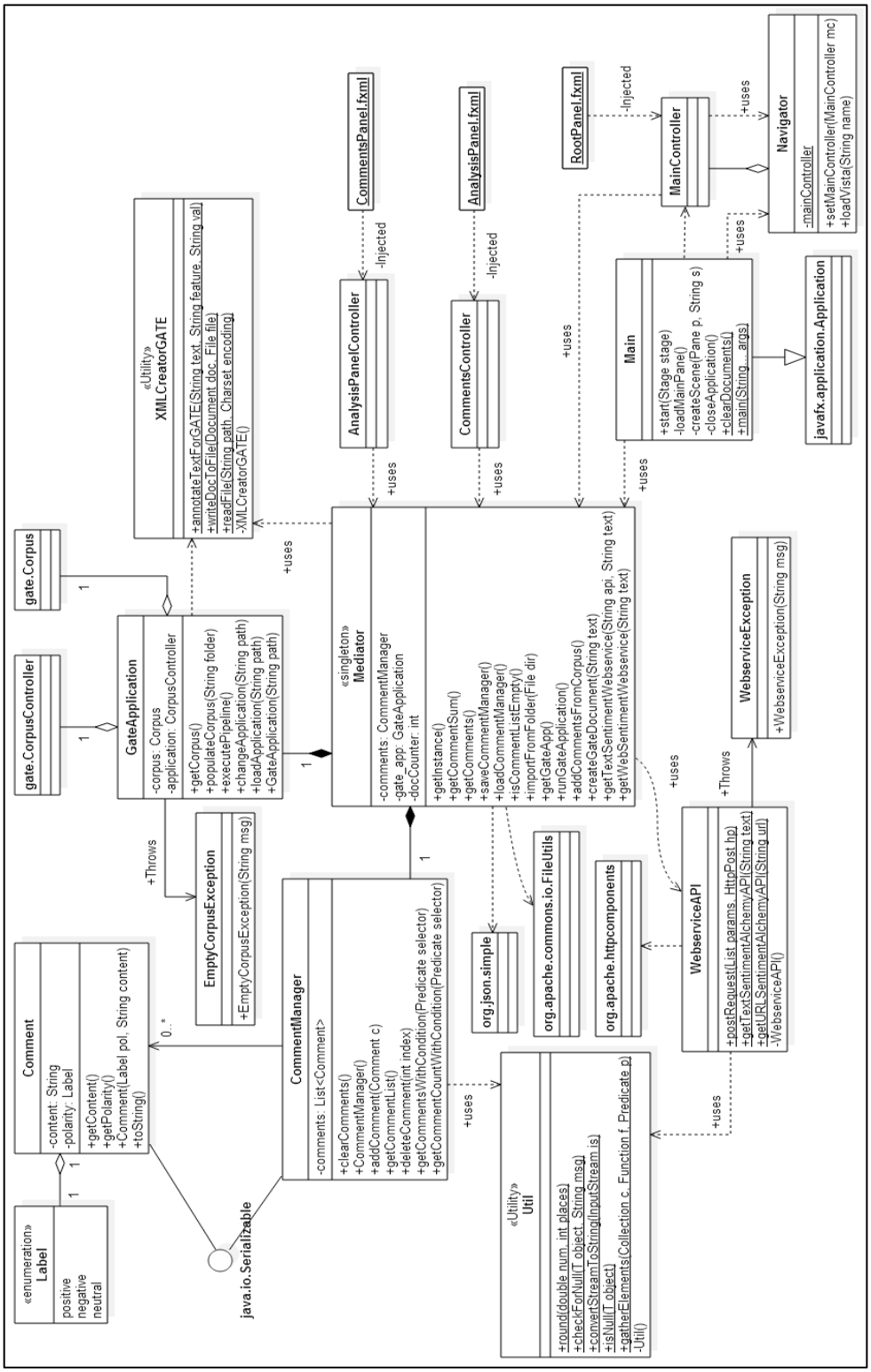


Figure 4.12. Class diagram

The primary entity object is specified by the Comment class that represents an instance of a document that is analysed by the classifier. A Comment object encapsulates the text of the document and the label of polarity. The label is a separate enumeration that defines the labels that this SA application handles (positive, negative, neutral). Moreover, the CommentManager is a helper class that provides a useful interface for collection processing for Comment objects. It manages a java.util.List of comments and provides processing methods because mainly collections of comments are processed in this application.

The GateApplication is the class that encapsulates the GATE communication between the model and the Java application. Through this class, clients can populate corpora, upload ML configurations, and execute GATE pipelines of operations. All of the functionality is fit into a single Java class thus offering extensibility and modularity. The class depends solely upon the GATE API and the annotation library (XMLCreatorGATE) and hence it can be used in any other system as a Java class or more conveniently, it can be published as a web http service. The usage of GATE embedded in conjunction with client written Java programs are operated through GATE Developer (the GUI application). Figure 4.13 shows the interaction diagram.

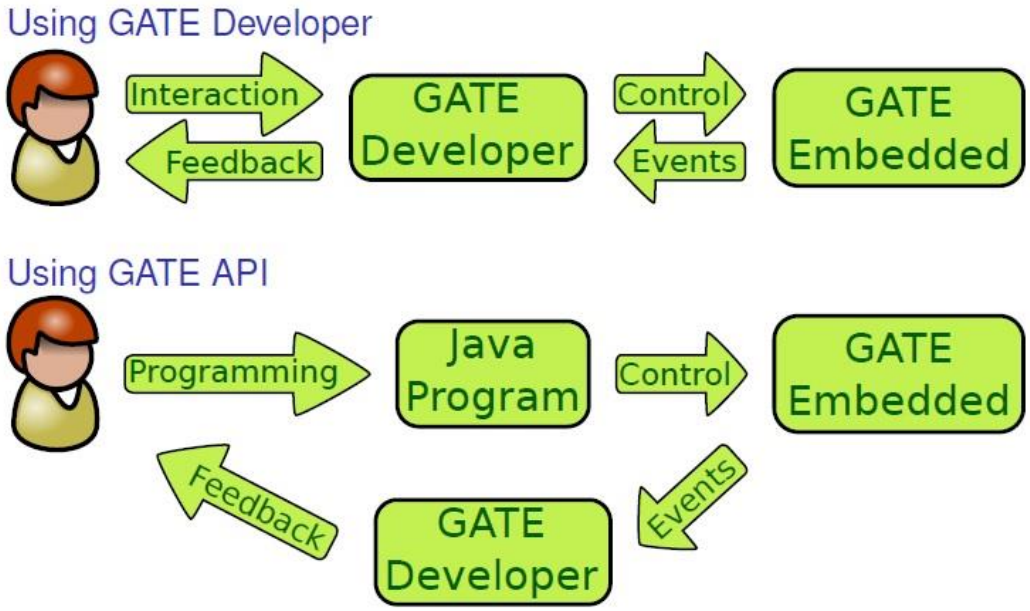


Figure 4.13. GATE API interaction (excerpted by [54])

In addition, the GateApplication class and the CommentManager are both encapsulated in the Mediator class that represents the primary interface between all the components of the application. It wraps the business logic and GATE methods and it is subsequently placed in the UI controllers to allow interaction with the user interface. This is a sound design decision due to the fact that the GUI acknowledges the existence of the Mediator only, providing room for refactoring and reusability because there are no other dependencies. It follows the general mediator pattern where a single god-like object is used to handle the communication between the objects of the system, thus reducing coupling and dependencies between the respective classes. The mediator is also a singleton object in the application as only one instance is required throughout the application. This concludes the vital internal design of the application; next the main use case is discussed along with the GUI design.

Following, the flow of operations regarding the main functionality of the system is described in detail. We are referring to the sentiment analysis of the input text documents. The general process that takes place from the initial documents to the final results is modelled as an activity diagram.

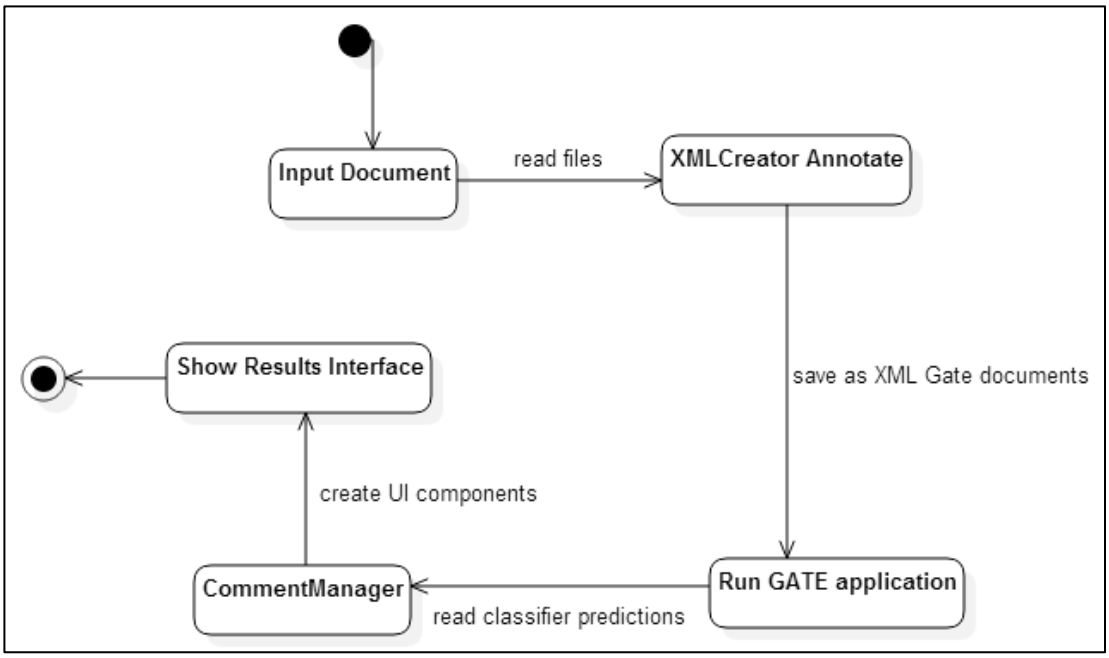


Figure 4.14. Business flow of sentiment analysis in the application

Text documents are input into the application, they are annotated by the presented library that transforms them in GATE XML documents and they are saved locally. Thereupon, these documents can be loaded as a corpus to GATE through the API and have the GATE pipeline application developed in GATE developer to run on these documents, producing various annotations, as shown in the model training section. The resulting annotations are read, meaning, the classifier's predictions are extracted and Comment objects are created and inserted into the manager class. This list is processed in the front-end to produce the overall results to the user.

Concerning the GUI design, a few quick prototypes were produced using JavaFX scene builder 2.0. The main concept behind these prototypes was kept, and after some adjustments they were concluded as the final. The UI was designed with a mind-set of a clean and simplistic look and feel. In order to avoid potential confusion, only two windows were created. The analysis panel shows the statistics and information about the inserted data and where users can perform main system operations, such as importing documents and analysing. Secondly, a comments panel was added to display each analysed document with the corresponding text and sentiment, according to the classifier. Users can hence check every processed document.

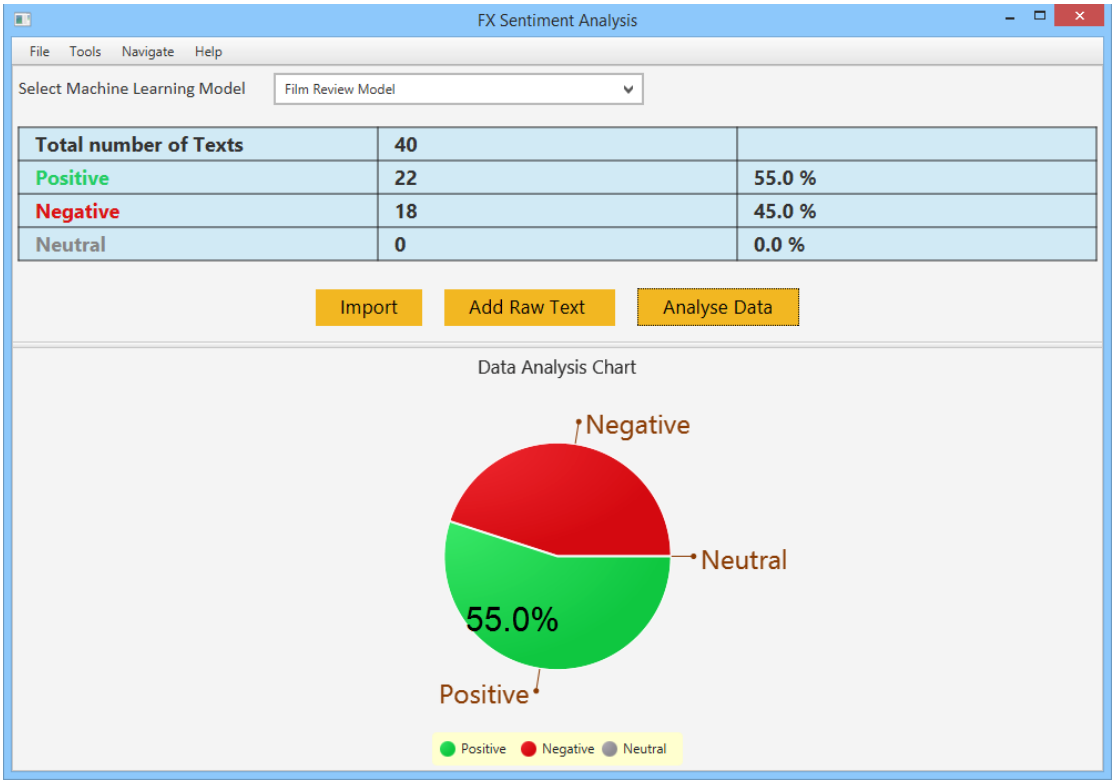


Figure 4.15. Analysis window of application

Figure 4.15 shows the main analysis window of the FX application. The table summarises important metrics about the data and through the button list, users can access various operations like importing and analysing. The ML model can also be changed through the combo box to address data of different domains (Twitter, and review of films). The south part of the split pane contains a pie chart, which is available in the JavaFX library. The three labels are the attributes of the chart and they are calculated based on the counts of each comment's label. The other window is the comment's panel and it consists of a table with all comments and their respective details. The textual content and the label assigned by the classifier. The next figure displays a populated comments table.

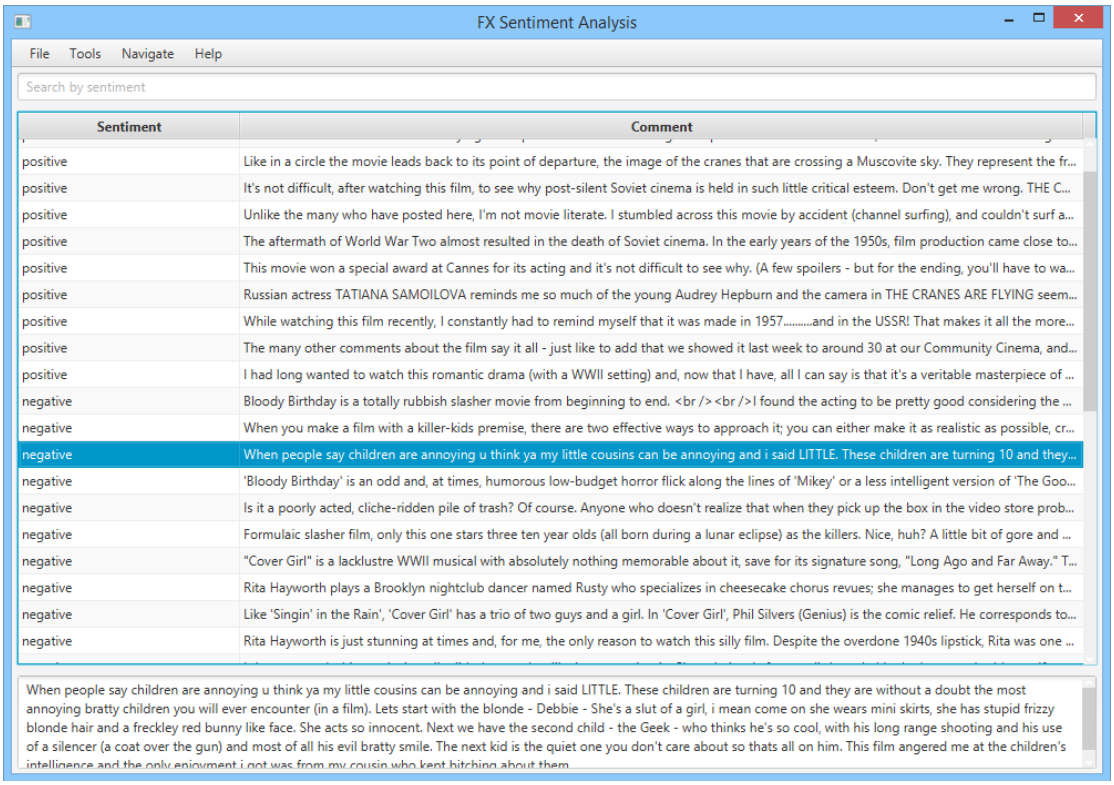


Figure 4.16. Comments window of application

It can be observed that these comments are actually film reviews. Each label is stated in the table under Sentiment and the comment's content can be seen in the below text area when clicked on the table. There is also a live search text field on the top where users can choose which type of sentiment comments to display.

4.6 Application Implementation

After the initial design of the system (Sent-Affect) and the GUI described in the previous section, the implementation aspects and considerations are discussed and presented in this part of the paper. To begin with, the first milestone of the implementation phase was to deliver a functioning integration and connection of the machine learning model developed in GATE with the Java application. The core functionality we aim to achieve is to assemble of corpus of documents and use our gate pipeline application with the tokenizers, sentence splitters, the learning algorithm, etc. on those input documents. A part of the main implementation is presented next.

```
public class GATEApplication {

    public GATEApplication(String appPath) {
        try {
            Gate.init();    // must run first before any GATE API
calls can be made.
            loadGATEApplication(appPath);
        } catch (GateException e) {
            System.err.println("Error-" + e.getMessage());
            e.printStackTrace();
        }
    }
    private void loadGATEApplication(String path) {
        // If the file cannot be loaded, a serious non-recoverable
error has occurred.
        try {
            ClassLoader classLoader = getClass().getClassLoader();
            File file = new File(classLoader.getResource(path)
                    .getFile());
            application = (CorpusController) PersistenceManager
                    .loadObjectFromFile(file);  // read the .gapp
file and set the controller
            corpus = Factory.newCorpus("mainCorpus");
            application.setCorpus(corpus);
        } catch (GateException | IOException e) {
            System.err.println("Error-" + e.getMessage());
            e.printStackTrace();
        }
    }
```

Listing 4.5. Gate interface implementation I

The GATEApplication class represents and encapsulates a GATE corpus pipeline application developed in GATE developer. The constructor requires .gapp file in order to run and that can be produced by GATE upon completing the model. The file is an XML specification of the processing resources used, the batch learning PR and the machine learning configuration file. The application file can be changed by the load method; this can be utilised when the machine learning model needs to switch from twitter to film and vice versa. The next two primary functions is the corpus management and executing the pipeline application. The corpus can be populated from a folder; this folder is within the class path of the Java project and every annotated document is saved there directly. Afterwards, the path is provided and the documents are loaded to the GATE corpus. Running the application requires a single command, as inspected in the second function in listing 4.6.

```
public void populateCorpus(String corpusPath)
                throws ResourceInstantiationException, IOException
{
            corpus.clear();
            File directory = new File(corpusPath);
            ExtensionFileFilter filter = new
ExtensionFileFilter("XML files", "xml");
            URL url = directory.toURI().toURL();
            corpus.populate(url, filter, null, false);
    }

public void executePipeline() throws ExecutionException {
        if (!corpus.isEmpty()) {
            application.execute();
        } else {
            throw new IllegalStateException("Cannot run pipeline on
empty corpus");
        }
    }
```

Listing 4.6. Gate interface implementation II

Having completed the GATE API integration for running the classifier on documents programmatically, the primary class that wraps the functionality can be implemented. The Mediator class is a singleton designed to encapsulate the ML requirements and every operation regarding comment management and processing. The outlines of the primary logic classes are the following.

```
public class CommentManager implements Serializable {
    private static final long serialVersionUID = 1L;
    private final List<Comment> comments;
...

public List<Comment> getCommentsWithCondition(Predicate<Comment> pred)
{
        return Util.gatherElements(comments, Function.identity(),
pred);
    }

public int getCommentCountWithCondition(Predicate<Comment> pred) {
        return (int) comments.parallelStream()
                            .filter(pred)
                            .count(); }
}
public final class Mediator implements MediatorInterface {
    private CommentManager comments;
    private final GATEApplication gateApp;
    private Mediator() {
        comments = new CommentManager();
        gateApp = new GATEApplication("Gate_apps/media_svm.gapp");
    }
    public static Mediator getInstance() {
        return Holder.INSTANCE;
    private static class Holder {
        private static final Mediator INSTANCE = new Mediator();
    }

public interface MediatorInterface {
    public void runGATEApplication();
    public void importFiles(File f);
    public void createGATEdocument(String text);
    public void addCommentsFromCorpus();
    public Label getWebpageSentiment(String url);
    public void loadCommentManager();
    public void saveCommentManager();
}
```

Listing 4.7. Mediator class implementation

The collection of analysed documents is represented by the CommentManager object and the other data field is the GATEApplication developed for the ML operation handling. The Mediator implements a singleton through the initialization on demand holder idiom the way it is described in Effective Java [57]. The constructor of the

GATE application receives the relative path of the pipeline application, the default is the Twitter model. The rest of the standard functionality is outlined in the interface that specifies the Mediator's methods.

The last aspect of the functionality concerns the analysis of a webpage via a URL. A certain web service API was used in order to achieve the desired effect. That API is the Alchemy (discussed in section 2.5) and it performs sentiment analysis on webpages as a service. In order to use the service, an HTTP post request was required.

```
public static String postRequest(List<NameValuePair> params, HttpPost
httppost) throws IOException {
        HttpClient httpclient = HttpClients.createDefault();
        httppost.setEntity(new UrlEncodedFormEntity(params, "UTF-
8"));
        //Execute and get the response.
        HttpResponse response = httpclient.execute(httppost);
        HttpEntity entity = response.getEntity();
        if (entity != null) {
            InputStream instream = entity.getContent();
            try {
                return Util.convertStreamToString(instream);
            } finally {
                instream.close();
            }
        }
        return null;
    }
public static String getURLSentimentAlchemyAPI(String url) throws
IOException {
        List<NameValuePair> params = new ArrayList<NameValuePair>(4);
        params.add(new BasicNameValuePair("apikey", ALCHEMY_KEY));
        params.add(new BasicNameValuePair("url", url));
        params.add(new BasicNameValuePair("outputMode", "json"));
        params.add(new BasicNameValuePair("showSourceText", "1"));
        return postRequest(params, new
HttpPost(GET_WEB_SENTIMENT_URL));
    }
```

Listing 4.8. Web service API

The apache http components API was used to implement the web service API. The Alchemy function accepts a URL from a publicly accessible webpage, cleans the mark-up language and conducts sentiment analysis on the content. The response from

the server is a JSON object. The JSON simple API was used to process the JSON responses.

Finally, regarding the GUI implementation; the JavaFX scene builder producers XML files (FXML) and their schema defines the graphical components of the interface. Subsequently, for each such file, a controller class must be created that will handle actions, set and edit component attributes and other similar UI considerations. Components such as buttons, labels, tables, charts, etc. are injected into the Java program upon execution. A controller class for each FXML file handles actions. Notice the @FXML tag on the components; these are injected components from the FXML schema. They can be consequently used as standard Java objects with their respective API according to the documentation. Thereupon, the remaining implementation solely rests on adding the appropriate function calls from the Mediator singleton in the listeners that update the graphical components. Listing 4.9 shows a skeleton outline of a controller class for an FXML file.

```
public class AnalysisPanelController implements Initializable {
    @FXML
    private Label negPct;
    @FXML
    private Label neuPct;
    @FXML
    private AnchorPane downPane;
    @FXML
    private PieChart pieChart;
    @Override
    public void initialize(URL rul, ResourceBundle rb) {
        setUpCombobox();
        if (!Mediator.getInstance().commentListIsEmpty()) {
            setUpCharts();
        }
    }
```

Listing 4.9. FXML controller class sample

This concludes the presentation of the implementation stage. Only the integral parts of the implementation were analysed being, the Gate API interconnection with Sent-Affect and the primary wrapper class. See appendix B for application images.

5. Discussion

The development of a machine learning model and particularly a supervised model, is a process that is lengthy and requires much consideration and attention in details. It has been explained how supervised learning models need to have training data as a basis and how this greatly affects the overall performance. The second vital consideration refers to the learning configuration, namely, algorithm selection, n-gram model, and other options relating to the algorithm such as margins in the feature vector space and a careful selection of linguistic features extracted from the text documents. These features are taken from the processing resources applied to the documents; a few examples are the tokenizer, part-of-speech tagger, and sentence splitter. These are standard implementations of the framework and cannot be modified in any way; however, the learning configuration is highly modifiable and hence, in order to develop a sound model with a given dataset, experiments have to be conducted to draw some conclusions of what settings work best in the particular case scenario. The answers to these questions do not exist, neither in literature nor in any system, rather they have to be given by the developer. Therefore, the interpretations and experience gained through this process are presented next and the topic will cover the two models, i.e., twitter, film review.

To start with, the film review model turned out to be extremely robust and efficient. This is owed partly to the high quality dataset of Pang and Lee [22]. The dataset with only 2,000 examples achieved great results with the linear classifiers both in the cross-validation and separate testing. Consult the following table for the detailed tests conducted.

n-gram model	***Classifier***	***Validation***	***Average accuracy %***
unigram	PAUM	10-fold	83.1
bigram	PAUM	10-fold	70.1
unigram	SVM Linear	10-fold	81.2
bigram	SVM Linear	10-fold	78
unigram	SVM Polynomial	10-fold	82.6
bigram	SVM Polynomial	10-fold	71.4

Table 5.1. Evaluation overall accuracy - film review model

The table 5.1 displays the accuracies of 10-fold cross-validation of the training dataset, of each classifier in the appropriate n-gram model. Due to their resemblance as concepts, the SVM linear and polynomial and the PAUM algorithm had similar performances with PAUM topping the other 2. The PAUM algorithm is also faster than the others. The validation process lasted 15.7 minutes for the PAUM, while 18.1 and 19 minutes for the linear and polynomial SVM respectively. These numbers apply for the unigram models. Concerning the bigram model, the results were disappointing. The bigram model concatenates features (words) and takes the words as pairs instead of viewing them individually. Bigram models are used sometimes to deal with negation, but the accuracies in this case fell significantly and at the same time, the process is more time consuming. The graph below illustrates the differences.

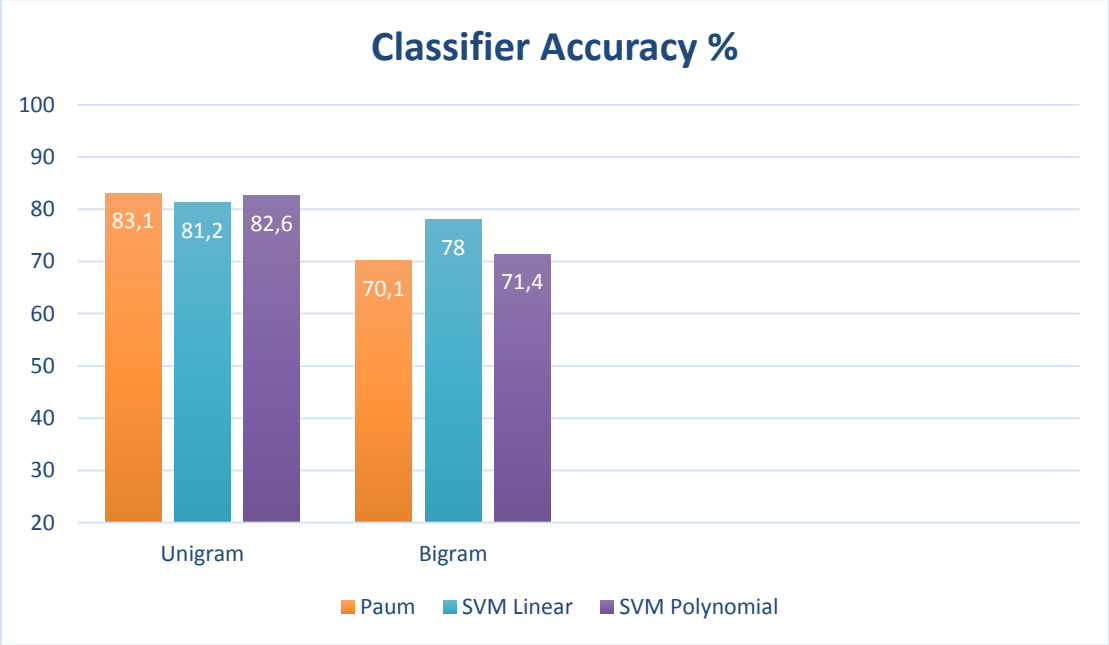


Figure 5.1. Cross-validation overall accuracies - film review model

Besides the evaluation methods, independent tests were conducted with other test datasets. A set of 1,000 film reviews was used to test each of these classifiers. The results are depicted in the following table.

n-gram model	*Classifier*	*Accuracy %*	*Classified*
unigram	PAUM	84.7	1000/1000
unigram	SVM (linear)	82.8	1000/1000
bigram	SVM (linear)	83.8	1000/1000
unigram	SVM (linear, P = 0.59)	89.4	756/1000
unigram	SVM (linear, P = 0.55)	86.9	865/1000
unigram	SVM (linear, P = 0.53)	85.2	921/1000
unigram	SVM (polynomial)	83.1	1000/1000

Table 5.2. Classifier test results - film review model

Observing the above table one can notice how one of the linear SVMs scored an accuracy of 89.4%. However, P = 0.59 indicates the probability threshold value, as seen in the ML configuration settings. As a result, the classifier was more than 59% certain for only 756 out of 1,000 documents. Recall is lost with a high threshold probability, but it can reduce some of the classifier guess work; instances that are very close to 50% to belong to a class. Further, the PAUM scored the highest accuracy with the highest possible recall, but the values are not very far from each other. On the other hand, surprisingly, the bigram SVM scored a sound 83.8%. This shows that one cannot rely on the results of one test set, and cross-validation is imperative indeed. The accuracy is calculate with the help of the confusion matrix. A confusion matrix displays the balance of predictions, that is, true positives, true negatives, etc. Consider the following matrix.

	Labelled negative	*Labelled positive*
negative	418	82
positive	71	429

The table shows on the left the instances that were negative with the top header what the classifier predicted. So, we have 418 true negatives, and 429 true positives and so forth. The accuracy hence is calculated as such:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

$$accuracy = \frac{429 + 418}{418 + 429 + 82 + 71} = \frac{847}{1000} = 84.7\%$$

Based on the aforementioned experiments, the classifier decided for the film review model was the unigram PAUM algorithm model. It showed consistency in the tests and achieved the best evaluation score. The downside is that it classifies every instance in a set, even in borderline cases of 50%, resulting in guessing, but it is at a reasonable level.

Moving on to the twitter and social networks model. A similar comparison of classifiers was conducted, but this time two different training datasets were compared as well. Due to the inadequacy and difficulty of the instances in this domain, there was trouble assembling a decent training sample. The starting data, as pointed in the data collection section was the STS-Gold dataset. Upon experimenting with it, it was confirmed that due to the imbalance of instances, the model was biased towards the negative class because they were superior in numbers. Check the following tables to verify the imbalance of predictions. It is seen that in the first confusion matrix, a test result from the STS-Gold dataset, there is a convergence on false negatives (298). Thus, the dataset that was extended with more positive examples to balance the scale improved this bias. The second confusion matrix shows the expanded dataset results.

	labelled negative	*labelled positive*
negative	919	81
positive	298	702

	labelled negative	*labelled positive*
negative	817	143
positive	92	948

Although the false positives were increased, overall the second result is better and more reliable on a larger scale. Next, the evaluation of the classifiers pertaining to both datasets is presented.

n-gram model	*Dataset*	*Classifier*	*Average accuracy %*
unigram	STS-Twitter	PAUM	78
unigram	STS-Twitter	SVM Linear	79.5
unigram	STS-Twitter	SVM Polynomial	80.1
unigram	STS-Twitter Bal.	PAUM	81
unigram	STS-Twitter Bal.	SVM Linear	80
unigram	STS-Twitter Bal.	SVM Polynomial	80.7

Table 5.3. Cross-validation overall accuracy - twitter model

In table 5.3, the evaluation of both datasets is see along with each classifier and the respective performance. The balanced dataset performed slightly better for each different classifier. This process indicates that supervised training may become a dataset evaluation as well as a classifier evaluation. Further tests verified the superiority of the balanced dataset and in addition, it supports emoticons that extend the range of the model. The chosen classifier for the Twitter model was the polynomial (cubic) SVM.

Now, some points about dealing with neutral posts. The threshold probability classification was set to 60% allowing for some instances to remain unlabelled. These were considered as neutral instances. The justification for this solution is also discussed in [58] which in short explains that neutral examples would lie in the boundaries of the feature space when solving the binary classification problem of positive and negative classes. Ergo, these boundary instances would have a low confidence probability by the classifier and can be consequently labelled as neutral. This is the most naïve approach, but due to the lack of solid neutral examples it would be counterproductive to introduce a new class in the learning process. Finally, another solution would be to create a different model for a neutrality check, but that would depend again on data, so there was definitely a compromise.

Overall, the learning stage of the project regarding both models was a success. The film review model developed is extremely robust and accurate compared to similar tries in research. The highest cross-validation accuracy surpassed slightly the highest achieved in [22] that worked with the same dataset. On the other hand, the Twitter model showed excellent results in the evaluation stage, there is much room for improvement by utilising larger training data. Conclusively, the project's objectives pertaining to the sentiment analysis engine were completed and fulfilled.

6. Conclusions

Sentiment analysis has a variety of uses and applications in review based platforms on the Web, in monitoring activities and posts in social networking sites, especially Facebook and Twitter and also as a component in other information systems (e.g. recommender). Opinion mining approaches are heavily dependent on domain and the discussion context, ergo, general sentiment analysis system may face issues in unknown topics. As a result, it is essential to develop specialised systems to cover the various domains that sentiments and opinions are expressed upon. The toughest challenges in sentiment analysis, since it is a natural language processing application are: sarcasm that invert sentiment meaning, use of negation, abbreviations and acronyms used in social networking sites (e.g. LOL, :D). Additionally, comparative sentences and conditional; these are the general issues derived from the ambiguity of natural language. Further, topic and entity identification are an issue, namely, what is the target of the commenter.

In order to tackle the problem of sentiment analysis, several techniques and methods are used nowadays. The two main being machine learning and lexicon approach. As shown, the powerful learning approach rely on known algorithms (naïve Bayes, maximum entropy) to solve the problem as a text classification issue and the polarity labels (positive and negative) are the classes. On the other hand, the lexicon approach using dictionaries and or rule based systems are heavily domain dependent and difficult to develop. For this reason and the dataset availability for supervised learning, such a ML approach was adopted for the completion of this project. The question of a NLP toolkit that supports learning was answered with the choice of GATE (general architecture for text engineering) that offers text analysis resources, plus learning features with support vector machine implementations.

The process followed to create a learning model for sentiment analysis was to collect first a training dataset. Annotate the documents according to the tool's specification in XML, build a pipeline of operations using tokenizers, part of speech taggers, and sentence splitter, and configure a machine learning file with options regarding algorithms, evaluation methods and n-gram models. We trained two models; one for Twitter and one for film reviews and they were tested and cross-validated achieving great results of over 80% in 10-fold validation. The trained classifiers can confidently understand affectivity in text and separate positive from

negative text at the document and sentence level. Subsequently, the intended application for this project, that is, Sent-Affect was designed to support the sentiment analysis engine. Using the GATE API to integrate the developed model programmatically in Java in the application, the core functionality of the system was successfully achieved. Sent-Affect is a JavaFX application that allows users to import text documents and then analyse them using the two models created. The results are summarised for the purposes of drawing conclusions from the labelled data. The other feature implemented is the ability to extract sentiment from a publicly accessible webpage, using technologies from Alchemy web service API this was realised.

Overall, the main objective of the project, i.e., to create a sentiment analysis system was completed with a favourable outcome. Both models are capable of reviewing opinionated content in the mentioned domains with a respectable accuracy. At the same time, the integration of the model to a custom-built application was completed as well. Sent-Affect meets the required functionalities described in chapter 3 along with every use case. Albeit, the quality of the models can be undoubtedly enhanced though a more refined approach, they are promising as an initial version.

Regarding potential future work, a key point of enhancing the system is to introduce, primarily, a more varied and complete training dataset. The datasets used in this project are not that big, but this also requires significant processing power. Moreover, an ideal feature in sentiment analysis systems is to be able to identify different opinions pertaining different entities in text. For example, a text may contain positive opinions about a person, but negative for a concept. This approach would require a named entity recognition system in conjunction with the sentiment classifier. Finally, the overall system could be extended to provide data mining functionalities that will extract tweets for example via the twitter API and analyse them. Such a combination would compose an extremely powerful application in market value.

References

[1] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Found. Trends® InformatioPang, B., Lee, L. (2006). Opin. Min. Sentim. Anal. Found. Trends® Inf. Retrieval, 1(2), 91–231.*, vol. 1, no. 1, pp. 91–231, 2006.

[2] B. Liu, "Sentiment Analysis and Opinion Mining," *Synth. Lect. Hum. Lang. Technol.*, vol. 5, no. May, pp. 1–167, 2012.

[3] A. Kumar and T. M. Sebastian, "Sentiment Analysis: A Perspective on its Past, Present and Future," *Int. J. Intell. Syst. Appl.*, vol. 4, no. September, pp. 1–[3], 2012.

[4] L. Carstens, "Sentiment Analysis - A multimodal approach," no. September, 2011.

[5] R. Xu and C. Kit, "Learning Multiple Level Features for Opinion Analysis," *Int. J. Comput. Process. Lang.*, vol. 23, no. 4, pp. 349–371, 2011.

[6] C. S. Analysis, "New Avenues in Opinion Mining and Sentiment Analysis," no. April, pp. 15–21, 2013.

[7] A. Westerski, "Sentiment Analysis: Introduction and the State of the Art overview," *Univ. Politec. Madrid, España*, pp. 211–218, 2007.

[8] B. Liu, "Sentiment Analysis and Subjectivity," *Handb. Nat. Lang. Process.*, pp. 1–38, 2010.

[9] F. Neri, C. Aliprandi, F. Capeci, M. Cuadros, and T. By, "Sentiment Analysis on Social Media," *2012 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Min.*, pp. 919–926, 2012.

[10] Y. A. Mejova, "Sentiment analysis within and across social media streams," 2012.

[11] B. Liu, "Sentiment analysis: A multifaceted problem," *IEEE Intell. Syst.*, vol. 25, no. 1, pp. 76–80, 2010.

[12] G. Gebremeskel and G. Date, "Sentiment Analysis of Twitter Posts About News," *Monogr. Soc. Res. Child Dev.*, vol. 76, no. May, p. 123, 2011.

[13] M. Wiegand, A. Balahur, B. Roth, A. Montoyo, and D. Klakow, “A Survey on the Role of Negation in Sentiment Analysis,” no. July, pp. 60–68, 2010.

[14] R. Narayanan, B. Liu, and A. Choudhary, “Sentiment analysis of conditional sentences,” *Proc. 2009 Conf. Empir. Methods Nat. Lang. Process. Vol. 1 EMNLP 09*, no. August, p. 180, 2009.

[15] L. Derczynski, D. Maynard, G. Rizzo, and M. Van Erp, “Analysis of Named Entity Recognition and Linking for Tweets,” pp. 1–35.

[16] D. Nadeau and S. Sekine, “A survey of named entity recognition and classification,” *Lingvisticae Investig.*, vol. 30, no. 1991, pp. 3–26, 2007.

[17] R. Feldman, “Techniques and applications for sentiment analysis,” *Commun. ACM*, vol. 56, p. 82, 2013.

[18] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, “Lexicon-Based Methods for Sentiment Analysis,” *Comput. Linguist.*, vol. 37, no. September 2010, pp. 267–307, 2011.

[19] W. Medhat, A. Hassan, and H. Korashy, “Sentiment analysis algorithms and applications: A survey,” *Ain Shams Eng. J.*, 2014.

[20] P. Langley and H. a. Simon, “Applications of machine learning and rule induction,” *Commun. ACM*, vol. 38, no. 11, pp. 54–64, 1995.

[21] R. Prabowo and M. Thelwall, “Sentiment analysis: A combined approach,” *J. Informetr.*, vol. 3, pp. 143–157, 2009.

[22] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment Classification using Machine Learning Techniques,” *Conf. Empir. Methods Nat. Lang. Process. (EMNLP 2002)*, pp. 79–86, 2002.

[23] T. M. Mitchell, “GENERATIVE AND DISCRIMINATIVE CLASSIFIERS : NAIVE BAYES AND LOGISTIC REGRESSION Learning Classifiers based on Bayes Rule,” *Mach. Learn.*, vol. 1, pp. 1–17, 2010.

[24] K. P. Murphy, “Naive Bayes classifiers Generative classifiers,” *Bernoulli*, vol. 4701, pp. 1–8, 2006.

[25] X. Zhu, “Naive Bayes Classifier,” no. 109275242, pp. 1–3, 2010.

[26] C. Fleizach, “A naive Bayes classifier on 1998 KDD Cup,” 1998.

[27] B. Networks, F. Faltin, and R. Kenett, “Bayesian Networks,” *Encycl. Stat. Qual. Reliab.*, vol. 1, p. 4, 2007.

[28] D. Heckerman, “A Tutorial on Learning With Bayesian Networks,” *Innov. Bayesian Networks*, vol. 1995, no. November, pp. 33–82, 1996.

[29] K. Nigam, J. Lafferty, and A. Mccallum, “Using Maximum Entropy for Text Classification,” *IJCAI-99 Work. Mach. Learn. Inf. Filter.*, pp. 61–67, 1999.

[30] N. Mehra, S. Khandelwal, and P. Patel, “Sentiment identification using maximum entropy analysis of movie reviews,” 2002.

[31] Ratnaparkhi A., ‘A Simple Introduction to Maximum Entropy Models for Natural Language’.

[32] L. Wang, “Support Vector Machines : Theory and Applications,” 2005.

[33] S. Tong and D. Koller, “Support Vector machine Active Learning with Applications to Text Classification,” *J. Mach. Learn. Res.*, pp. 45–66, 2001.

[34] M. G. Patil, M. V. Galande, V. Kekan, and M. K. Dange, “Sentiment Analysis Using Support Vector Machine,” pp. 2607–2612, 2014.

[35] S. Baccianella, A. Esuli, and F. Sebastiani, “SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining,” *Proc. Seventh Int. Conf. Lang. Resour. Eval.*, vol. 0, pp. 2200–2204, 2010.

[36] GATE, “Gate: General Architecture for Text Engineering”, https://gate.ac.uk/, [Accessed: October 10, 2014].

[37] Semantria, https://semantria.com/, [Accessed: October 10, 2014].

[38] RapidMiner, rapidminer.com, [Accessed: October 11, 2014].

[39] M. Hall, et al. (2009), The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[40] LingPipe, http://alias-i.com/lingpipe/, [Accessed: October 12, 2014].

[41] Google Prediction API, https://cloud.google.com/prediction/docs, [Accessed: October 20, 2014].

[42] Python Natural Language Toolkit, http://www.nltk.org/, [Accessed: October 14, 2014].

[43] Stanford NLP, http://nlp.stanford.edu/sentiment/, [Accessed: December 10, 2014].

[44] Repustate, https://www.repustate.com/, [Accessed: October 20, 2014].

[45] Alchemy API, http://www.alchemyapi.com/, [Accessed: October 21, 2014].

[46] Bitext API, http://www.bitext.com/bitext-api-2.html, [Accessed: October 21, 2014].

[47] T. I. Jain and D. Nemade, "Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis," *Int. J. Comput. Appl.*, vol. 7, no. 5, pp. 12–21, 2010.

[48] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," *Proc. 49th Annu. Meet. Assoc. Comput. Linguist. Hum. Lang. Technol.*, pp. 142–150, 2011.

[49] H. Saif, M. Fern, Y. He, and H. Alani, "Evaluation Datasets for Twitter Sentiment Analysis: A survey and a new dataset, the STS-Gold," *Emot. Sentim*, 2013.

[50] Thelwall, M., Buckley, K., Paltoglou, G. Cai, D., & Kappas, A. (2010). Sentiment strength detection in short informal text. Journal of the American Society for Information Science and Technology, 61(12), 2544–2558.

[51] Thelwall, M., Buckley, K., & Paltoglou, G. (2012). Sentiment strength detection for the social Web, Journal of the American Society for Information Science and Technology, 63(1), 163-173.

[52] Thelwall, M., & Buckley, K. (2013). Topic-based sentiment analysis for the Social Web: The role of mood and issue-related words. Journal of the American Society for Information Science and Technology, 64(8), 1608–1617.

[53] H. Cunningham, V. Tablan, A. Roberts, K. Bontcheva (2013) Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics._PLoSComput_Biol_9(2):_e1002854._doi:10.1371/journal.pcbi.100284 — http://tinyurl.com/gate-life-sci/.

[54] H. Cunningham, et al. Text Processing with GATE (Version 8). University of Sheffield Department of Computer Science. 15 April 2011. ISBN 0956599311.

[55] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, “Sentiment analysis of Twitter data,” *Assoc. Comput. Linguist.*, pp. 30–38, 2011.

[56] Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola, “The Perceptron Algorithm with Uneven Margins,” 2002.

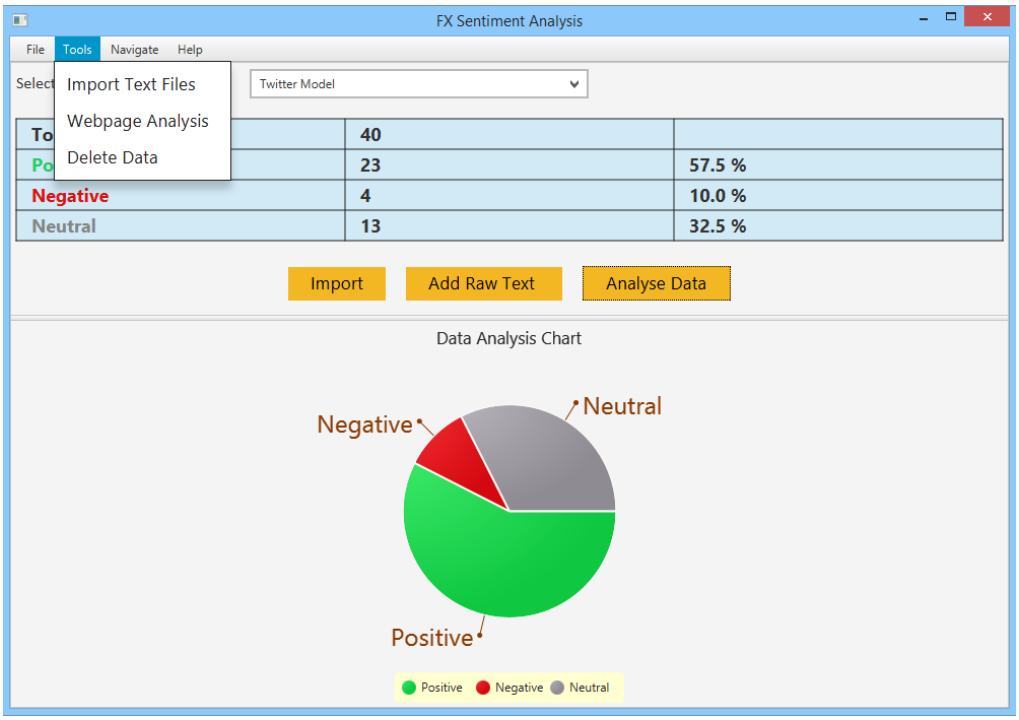
[57] J. Bloch. Effective Java, Second Edition. Addison-Wesley, 2008.

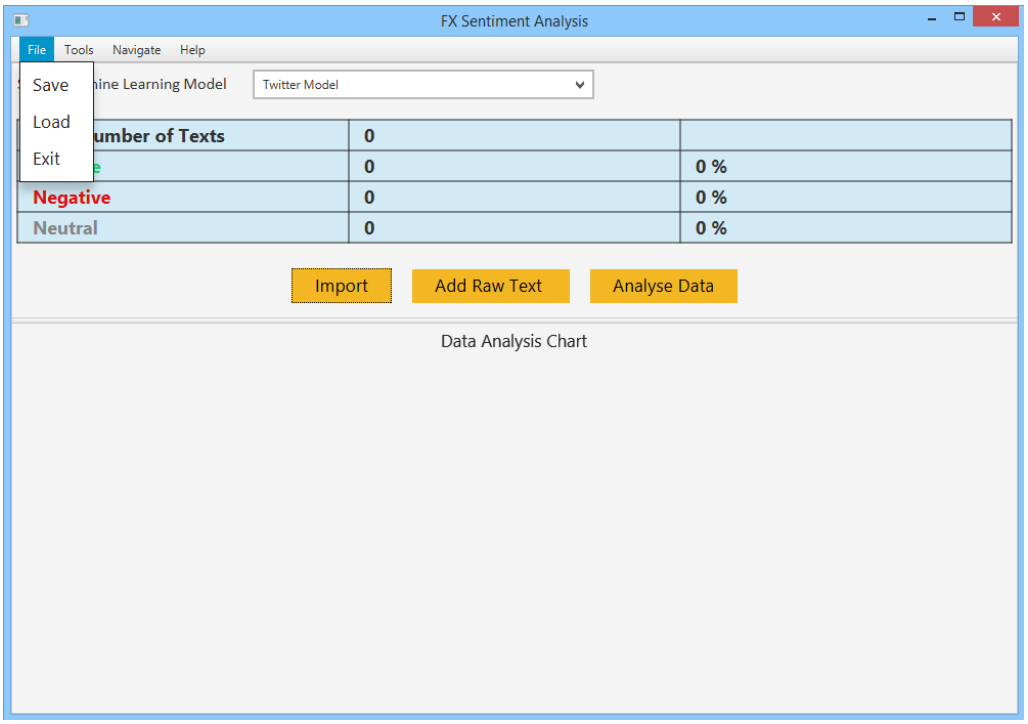
[58] M. Koppel and J. Schler, “The importance of neutral examples for learning sentiment,” *Comput. Intell.*, vol. 22, pp. 100–109, 2006.

Appendix A

```
public class DatasetReaderCSV {
    private List<Post> instances = new ArrayList<>();
    public List<Post> readPosts(String splitChar, String
fileLoc) {
        String csvFile = fileLoc;
        BufferedReader br = null;
        String line = "";
        String cvsSplitBy = splitChar;
        try {
            br = new BufferedReader(new FileReader(csvFile));
            while ((line = br.readLine()) != null) {
                String[] post = line.split(cvsSplitBy);
                String pol = post[0];
                String content = post[1];
                instances.add(new Post(calculatePolarity(pol),
content));
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (br != null) {
                try {
                    br.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return instances;
    }
```

Appendix B


FX Sentiment Analysis
File
Tools
Navigate
Help
Import Text Files
Webpage Analysis
Delete Data
Twitter Model
40
23
57.5 %
Negative
4
10.0 %
Neutral
13
32.5 %
Import
Add Raw Text
Analyse Data
Data Analysis Chart
Negative
Neutral
Positive
Positive
Negative
Neutral


FX Sentiment Analysis
File
Tools
Navigate
Help
Save
Load
Exit
Twitter Model
0
0
0 %
Negative
0
0 %
Neutral
0
0 %
Import
Add Raw Text
Analyse Data
Data Analysis Chart