**Unsupervised aspect based opinion mining using Double Propagation**

LICENSE THESIS

|  |  |  |
| --- | --- | --- |
|  | Graduate: | **Darius Andrei SUCIU** |
|  |  |  |
|  | Supervisor: | **Prof.dr.eng. Rodica POTOLEA**  **Prof.dr.eng. Mihaela DINŞOREANU** |

**2014**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| DEAN, |  | | HEAD OF DEPARTMENT, | |
| **Prof. dr. eng. Liviu MICLEA** |  | **Prof. dr. eng. Rodica POTOLEA** | |

Graduate: **Darius Andrei SUCIU**

**Unsupervised aspect based opinion mining using Double Propagation**

1. **Project proposal:** *Design and develop a solution for extracting opinions, respective opinion words and opinion targets, by using an unsupervised method, ensuring domain independency. The proposed solution identifies in almost real-time opinions in a text belonging to any domain, at aspect level, by using a double propagation approach and computes its polarity.*
2. **Project contents:***Presentation page, advisor's evaluation, Introduction, Project Objectives, Bibliographic Research, Analysis and Theoretical Approach, Detailed Design and Implementation, Testing and Validation, User`s Manual, Conclusions, Bibliography, appendices.*
3. **Place of documentation**: Technical University of Cluj-Napoca, Computer Science Department
4. **Consultants**: -
5. **Date of issue of the proposal:** March 1, 2013
6. **Date of delivery:** July 3, 2014

|  |  |  |
| --- | --- | --- |
| Graduate: | | Darius Andrei SUCIU |
|  | |  |
| Supervisors: | Prof. dr. eng. Rodica POTOLEA  Prof. dr. eng. Mihaela DINSOREANU | | |

**Declaraţie pe proprie răspundere privind**

**autenticitatea lucrării de licenţă**

Subsemnatul SUCIU DARIUS ANDREI, legitimat cu C.I. seria AX nr. 399999  
CNP 1911104015560, autorul lucrării EXTRAGERE NESUPERVIZATA DE OPINII LA NIVEL DE ASPECTE FOLOSIND PROPAGAREA DUBLA elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea Calculatoare din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea IULIE a anului universitar 2014, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

|  |  |  |
| --- | --- | --- |
| Data  03.07.2014 |  | Nume, Prenume  Suciu, Darius Andrei |
|  |  |  |
|  |  | Semnătura |

Table of Contents

[Chapter 1. Introduction 1](#_Toc391403453)

[1.1. Data mining 1](#_Toc391403454)

[1.2. Opinion mining 2](#_Toc391403455)

[1.3. Unsupervised aspect based opinion mining 2](#_Toc391403456)

[1.4. Thesis structure 2](#_Toc391403457)

[Chapter 2. Project Objectives 4](#_Toc391403458)

[2.1. Problem Statement and Motivation 4](#_Toc391403459)

[2.2. Objectives 5](#_Toc391403460)

[2.3. Specifications 7](#_Toc391403461)

[2.3.1. Functional Requirements 7](#_Toc391403462)

[2.3.2. Non-functional Requirements 7](#_Toc391403463)

[Chapter 3. Bibliographic Research 9](#_Toc391403464)

[3.1. Natural Language Processing 9](#_Toc391403465)

[3.2. Opinion Mining 9](#_Toc391403466)

[3.3. Opinion Mining Approaches 10](#_Toc391403467)

[3.3.1. Supervised Sentiment Analysis 10](#_Toc391403468)

[3.3.2. Unsupervised Sentiment Analysis 11](#_Toc391403469)

[3.3.3. Feature Extraction 12](#_Toc391403470)

[3.3.4. Feature and Opinion extraction 12](#_Toc391403471)

[3.4. Text pre-processing tools 14](#_Toc391403472)

[3.4.1. Part of Speech Tagger 14](#_Toc391403473)

[3.4.2. Parser 14](#_Toc391403474)

[3.4.3. Tokenizer 14](#_Toc391403475)

[3.4.4. Lemmatizer 15](#_Toc391403476)

[3.5. Lexical Resources 15](#_Toc391403477)

[3.5.1. WordNet 15](#_Toc391403478)

[3.5.2. SentiWordNet 15](#_Toc391403479)

[Chapter 4. Analysis and Theoretical Foundation 17](#_Toc391403480)

[4.1.1. Aspect Based opinion mining 17](#_Toc391403481)

[4.1.2. Opinion Word and Target Extraction Rules 18](#_Toc391403482)

[4.1.3. Double Propagation Algorithm 22](#_Toc391403483)

[4.1.4. Proposed Solution 24](#_Toc391403484)

[4.1.5. Evaluating Extraction 27](#_Toc391403485)

[Chapter 5. Detailed Design and Implementation 29](#_Toc391403486)

[5.1. The overall opinion mining system 29](#_Toc391403487)

[5.1.1. Architecture 30](#_Toc391403488)

[5.1.2. Retriever Service 30](#_Toc391403489)

[5.1.3. Polarity Aggregation 31](#_Toc391403490)

[5.1.4. Polarity Service 31](#_Toc391403491)

[5.2. Feature-opinion pair identifier 31](#_Toc391403492)

[5.2.1. The Data Model 33](#_Toc391403493)

[5.2.2. Double propagation module 35](#_Toc391403494)

[5.2.3. Opinion word extractor 37](#_Toc391403495)

[5.2.4. Opinion word pruning module 39](#_Toc391403496)

[5.2.5. Feature extractor 40](#_Toc391403497)

[5.2.6. Feature pruning module 41](#_Toc391403498)

[5.2.7. Extraction service 41](#_Toc391403499)

[5.2.8. Source-Target pair merger 44](#_Toc391403500)

[5.2.9. POS relations aggregator 45](#_Toc391403501)

[5.3. The Evaluation System 46](#_Toc391403502)

[5.3.1. Opinion mining system in evaluation mode 47](#_Toc391403503)

[5.3.2. Opinion word and Feature Extraction evaluators 47](#_Toc391403504)

[Chapter 6. Testing and Validation 50](#_Toc391403505)

[6.1. Data set 50](#_Toc391403506)

[6.2. Experiments and results 51](#_Toc391403507)

[6.2.1. Initial Results vs. Final Results 51](#_Toc391403508)

[6.2.2. Seed words number influence 52](#_Toc391403509)

[6.2.3. Polarity Threshold influence 54](#_Toc391403510)

[6.2.4. The X-Filter influence 55](#_Toc391403511)

[6.2.5. POS tags influence 57](#_Toc391403512)

[Chapter 7. User’s manual 59](#_Toc391403513)

[7.1. Generalities 59](#_Toc391403514)

[7.2. Prerequisites 59](#_Toc391403515)

[7.2.1. **Java JDK** 59](#_Toc391403516)

[**7.2.2.** **Eclipse IDE** 59](#_Toc391403517)

[7.3. Additional requirements 60](#_Toc391403518)

[**7.3.1.** **GIT and GIT Cloning** 60](#_Toc391403519)

[**7.3.2.** **Maven in Eclipse** 61](#_Toc391403520)

[**7.3.3.** **Importing the project** 62](#_Toc391403521)

[7.4. Running and testing the project 63](#_Toc391403522)

[**7.4.1.** **Configuration management and initial settings** 64](#_Toc391403523)

[**7.4.2.** **Running the project** 64](#_Toc391403524)

[Chapter 8. Conclusions 65](#_Toc391403525)

[**8.1.1.** **Summary and contributions** 65](#_Toc391403526)

[**8.1.2.** **Experiments and results conclusions** 65](#_Toc391403527)

[**8.1.3.** **Future Development** 66](#_Toc391403528)

[Bibliography 68](#_Toc391403529)

[Appendix 1 KDIR 2014 Article 7](#_Toc391403530)1

[Appendix 2 Students’ Conference 2014 Article 8](#_Toc391403530)4

Table of Figures

Table 4.1 Opinion word and target extraction rules 19

Table 4.2 Rule Examples 21

Figure 4.1 The architecture of the aspect based opinion mining system 25

Table 4.3 Confusion matrix for extraction 27

Figure 5.1 The architecture of the OMS system 30

Figure 5.2 The architecture of the FOPI component 32

Figure 5.3 FOPI module UML 33

Figure 5.4 Graphical representation of a semantic graph 34

Figure 5.5 The architecture of the Double Propagation module 36

Figure 5.6 Opinion word extraction flow 37

Figure 5.7 Feature extraction flow 40

Figure 5.8 The OMS in evaluation mode 47

Figure 6.1 Initial vs final results of opinion word extraction 51

Figure 6.2 Initial vs Final results of the target extraction 52

Figure 6.2 Seed words influence on opinion word extraction 53

Figure 6.3 Seed words influence on target extraction 53

Figure 6.4 Seed words number influence on the extraction time 54

Figure 6.5 Polarity theshold influence on opinion word extraction 55

Figure 6.7 X-filter inflience on opinion word extraction 56

Figure 6.8 X-filter influence on target extraction 56

Figure 6.9 POS tags influence on opinion word extraction 57

Figure 6.10 POS tags influence on target extraction 57

Figure 6.11 Cross domain comparison of opinion words and targets 58

Figure 6.12 Polarity Assignment scores 59

Figure 7.1Menu Items 61

Figure 7.1 Git Clone Menu 62

Figure 7.2 Install Menu 63

Figure 7.3 Maven Project Import Menu 64

# Introduction

We are becoming is more and more focused on quantifying the information surrounding us, with great efforts being made for storing enormous amount of information, mostly collected from the internet in large private data centers. By what use is information that just exists, if no way of effectively searching or processing it exists. Imagine having 1000 fiction books, which you can freely use for the creation of a new one, considering no one knows what they contain, but the time allowed for the creation being of two days. This scenario accurately reflects the current situation in many of the companies that provide services for humans, meaning all of them.

The need of systems which extract the needed relevant information is obvious, a need directly reflected in the rapid and continuous increase in data mining interest and progress.

## Data mining

Any company has a base of customers; it’s a fundamental condition for its existence, meaning satisfying the customers’ demands is their main objective. The number of customers ranges greatly, from a few dozens to the more frequent number of thousands, millions or even billions, depending on the type of service provided. For example 1.3 billion people use Microsoft every day. If we consider that only 10% of them experience problems with the new updates installed, this means the company risks using 130 million customers, if they do not read each and every one of their complains, as fast as possible, as they need to be also solved timely. The amount of human power needed clearly exceeds the possibilities of the company, as billions of dollars would be needed every month to support it. This is the reason why system capable of processing that human generated data are in great demand, a demand reflected in the birth of the data mining domain.

Let’s begin by clearly defining data mining and then continue to the specific domain that handles the subjective texts, a domain where the solution proposed in this thesis belongs.

Data mining can be defined as the non-trivial extraction if implicitly, previously unknown and potentially useful information from data, according to [1], one of the many definitions available. There are many tasks that can be done when manipulating data, some which use machine learning and pattern recognition, leading to classification that needs data from which it can infer certain rules, called training data and then uses them on a related set of data. The other tasks include clustering, regression, summarization, associative rule learning and anomaly detection, each responsible for solving a different class of problem, but all using the semi-automatic analysis of large quantities of data to extract previously unknown relevant information, such as data records for cluster analysis or unusual records for anomaly detection.

So data mining is used for getting relevant information, like the feedback of users in the case of the example given, but there is a multitude of relevant information contained in a text, depending on the needs of the user, thus the different tasks contained by the domain. One might look for the results of a research paper, other of reported problems, and another for finding out the technique used for solving a problem that has been bugging him for months. In the beginning of the “interest boom”, data mining represented the domain used for all the extraction and processing techniques, but now just mentioning data mining as the context of a system is far from sufficient from making anyone have a general idea of what it’s supposed to do.

## Opinion mining

The importance of opinion is obvious, considering the interest dedicated by companies and important persons for creating and maintaining a positive image. A sub task in handling publicity is collecting feedback, thus opinions, a task so important that it lead to the creation of a large amount of survey sites or survey companies, whose sole role is the collection of opinions. But the cost of such methods is immense, as both the time and resources required are huge. The interest, along with the possibilities offered by the advances in technology gave birth to the creation of a domain specialized in such tasks, called opinion mining.

Opinion mining is a specialization of data mining, responsible for handling the subjective information contained in texts. In the example with Microsoft given, the user feedback contains both important technical information, like the issues discovered, but it might also contain evaluation of the services performed by the company, evaluation represented by personal opinions, which are much harder to identify. The solving of this difficult task is the focus of all opinion mining approaches, by the usage of natural language processing, text analysis and computational linguistics. Generally this domain aims to determine the attitude of a writer, customer or speaker in respect to a topic or product.

Depending on the approach used, the domain is split into supervised and unsupervised techniques, the supervised ones relying on learning from a known text, processed by humans, while the unsupervised aim to handle totally autonomous the task of retrieving opinions.

## Unsupervised aspect based opinion mining

The unsupervised aspect based opinion mining represented the most specific domain where the solution proposed aims to bring a difference. The aspect based opinion mining is used for extracting opinions on features of objects, which are generally products. For example, a phone has a display, a battery, a keyboard and so on, each of which can be the targets of an opinion and are considered features of the phone, during its evaluation or description. The focus of aspect based mining lies on identifying relevant information on each feature of a product separately.

The focus of the thesis is extracting the opinion on such features, contained in user reviews, in an unsupervised, human independent way, by using grammar rules which exist in natural written text to break the domain dependence barrier that exists in most similar solutions.

## Thesis structure

The thesis is structured in 8 chapters, each focusing on describing a part of the problem solved or solution developed.

Chapter 2 gives an overview of problem is given, along with the objectives and requirements of the solution needed for solving it.

Chapter 3 focuses on presenting the evolution of the existing techniques used for solving such problems, along with the most important works done in this direction and technologies available.

Chapter 4 gives the theoretical foundation of the solution proposed, containing explanations of techniques and algorithms used in its development.

Chapter 5 starts with the presentation of the overall solution and its implementation, continuing with the description of the component responsible for extracting the opinion words and targets from a text, with the justification and explanations necessary and ending with details regarding the technologies used.

Chapter 6 presents the experiments conducted and their results, with a short analysis on each of them, detailing the reasons for the system behavior.

Chapter 7 contains the necessary information for installing and using the solution implemented, along with the tools needed.

Chapter 8 shows the conclusions drawn from the work done, by making a short summary, presenting the improvements added and possible future work.

# Project Objectives

## Problem Statement and Motivation

The millions of reviews which are freely available on the internet, providing invaluable feedback, with the only requirement for harvesting it being the time needed to read each and every one of them are the main motivation for our drive to make an opinion mining system. We live in the information era, which means that collecting information is one of the most important jobs today, and will be even more in the future. Of course just collecting all the possible information is useless, because even we as humans can only remember a small finite amount, even though our brains can absorb much more information than any of the current storage devices.

We constantly try to filter all the garbage, specifically information which is not relevant to us at the time we read it, in order to retain the one which we can use and need. So we waste time collecting as much information as possible, then more time filtering it, time which is also a precious resource. At the same time, we focus on the creation of machines which can handle all hard, time consuming tasks which we do not want to do, moreover, we design them to do them faster, more effectively, so building a system to do this task seems almost intuitive.

The value of reviews as information is obvious to everyone, as we all want to know as much as possible when purchasing a product, to see if it fits our needs, if it`s want we are looking for and perhaps most importantly to see if the ones that have it are satisfied with the decision of purchasing it. But the value of feedback is even greater to the companies which produce them, as they constantly seek ways to improve the product, to overcome the competition, the best guidelines being the enormous information provided by the users, which can point our flaws, and effectively say what needs to be done in order for it to sell better, which is the main aim of all companies. The efforts for getting and processing good feedback can be seen in the increasing number of surveys done, each of which requires over 30 minutes of a paid employee and a customer and much more time and money invested in analysts, meaning for a relevant amount of surveys, an amount of money is needed to be invested. But why do companies do surveys, if they have hundreds of reviews freely available on the internet? The answer lies in the processing time and effort needed to draw the needed information from them, specifically the opinions.

Opinion mining is the response to the demand of getting the relevant information from reviews and all other subjective texts, respectively opinions. It`s a relatively new domain, but its rapid development and increased interest shows its value, because even though the results were not promising in the beginning, before 2000, the researches done in this domain continued to increase and improve in quality, until they provided viable solutions. The systems developed in late years provide satisfactory results, and can be effectively used for harvesting opinion from any text, but only by providing a lot of learning material, which is another very time consuming process, as correctly labeling a text does not only require reading it, but also identifying the opinions in it correctly, a subjective process which needs several passes to be done by different persons. This led to the dedication of additional effort in the creation of systems which do not require learning material, ideally system which we can almost just switch on, just like a printer, provide the text and get the information out immediately.

The desire to know others opinions seems like a basic need, we want to know what other people think about a movie before we watch it, in order to decide if it’s worth dedicating 2 hours of our lives, when deciding on a school or workplace, we try to collect as much opinions as possible, because we need that information to make the most probable correct decision. In the case of movies, we generally read one or two reviews and decide if it’s worth our time by looking at its ranking which is provided by a dozens of sites, like *imdb* or *rottentomatoes*, but what we actually want is not just the rating put to represent the movie`s value, because it does not offer sufficient information, as people have different metrics for its evaluation just a score is insufficient. We want to see what they think about it, if it’s scary, boring, disgusting or what part of it is good, the plot, the action, the acting, in order to have all the relevant information, but this would require reading thousands of reviews, meaning much more time would be spent on reading about a move than actually watching it. This gives the value of opinion mining systems for all users, as the reviews are not only restricted to movies, as mentioned before, reviews of products are desired by all people to be known, of restaurants, companies, effectively everything that has any connection to us, but we need to have the information in a format that requires at most a few minutes of our time, or the information becomes irrelevant, because it exceeds the timeframe allocated for making a decision.

To further justify the values of opinions and of their extraction, we can use the stock markets as an example. The value of stocks of a company decrease if the people who invested in it become aware of bad publicity, meaning if people think that a company is doing bad in terms of business, it will actually do so because the stakeholders will think that as well and begin to sell stocks, which in turn makes it loose more stocks, beginning a snowball process, all because the opinions of some people affect the actions of other people directly. This is a reason why effort is dedicated to predict the stock market`s behavior based on the subjective texts written in forums by combining opinion mining systems with probabilities and machine learning.

## Objectives

The main objective of this thesis is the creation of a system capable of extracting all the opinions from a text, in a manner as close as real-time as possible, without the need of training data, whose creation is a very time consuming task. The system should not be restricted to a single domain of usage, because as stated before, opinion mining systems have a large usage domain, and the current trend is the creation of domain independent systems, even language independent. For the purpose of providing the opinions, the system aims to provide all the opinions, represented by a set of opinion words, for each feature in a text, which are the components or the target of the review itself. By target we refer to the movie, product or any other entity on which the review focuses on providing feedback.

The aim is to eliminate all the human effort consumed in the extraction of feedback from any text, which was previously wasted in the processing of information by analysts, or more recently by annotators which basically have to do the same job as before, just on a smaller amount of data.

An opinion can be seen as comprised of two components, the words which are used to reflect an opinion, like *good* or *bad* and the polarity of the opinion. The system we propose, gets opinion by extracting their components separately, a component is responsible for the extraction of the opinion words and the features they target, and one for calculating the polarity of the opinions. The extraction of opinion words can be done independently from the calculation of polarity, but the calculation of polarity cannot be done without any opinion words.

Considering we extract the opinion words for each feature in a text, the problem we solve is contained in the aspect based opinion mining, the most difficult and desired type, because we want as much details as possible, the extraction of opinions which describe the features of a product rather the whole product is more valuable. The aspect based opinion mining provides the most complex opinion possible using text processing, and it can be easily be used to create the document level opinion mining, which is represents a higher granularity of opinion mining, by simple aggregation of results. For better understanding the value and justify the effort dedicated to extracting opinions for each feature in a text, let’s consider a telephone. In some cases, we are satisfied knowing the telephone is good overall, but generally we want to know how good its camera is, how well its microphone works, if the speakers are loud and clear enough. Finding out the overall opinion of a telephone represents the job of document level opinion mining, but the details regarding its components require the extraction of its features and all the opinions associated to each feature.

The goals for each of the system component, which when combined accomplish the task of aspect based opinion mining are the following:

* Get a text with the necessary information, called preprocessed text that is ready to be used by opinion extracting systems.
* Extract the opinion words and their targets in a form that maintains their relation, from a preprocessed text.
* Assign the polarities to every opinion word and target extracted, by using their relationship.

In this thesis, the focus is on the creation of the component responsible for extracting the opinion words and targets from a preprocessed text, the other two subtasks being accomplished by the work done by two of my colleagues, the preprocessing component being presented in [2] and the component responsible with computing the polarities being detailed in [3].

The aim of the component implemented is to work in correlation with the other two components, which when combined solve the problem of opinion mining, but at the same time, the implementation of the component is designed to independently solve the problem of extracting opinion words and targets. It cannot perform the extraction process without any preprocessing component, but a big focus was places on ensuring it can do it with any preprocessing component, due to the large number of existing systems that accomplish this subtask, which continuously evolve, meaning each of the components of the system might become obsolete. The component is designed to be usable by any opinion mining system which would need or benefit from the fast and reliable extraction of opinion words and features.

## Specifications

In this section the specification of the system are presented which consists of the functional and non-functional requirements. The specifications are the starting point of a project design, which need to give a clear picture of the purpose the systems objective and the way it aims to accomplish it.

### Functional Requirements

The description of the intended behavior of the system is covered by the functional requirements. They present the tasks which need to be accomplished by the system in order to fulfill its intended end objective. The functional requirements are directly correlated to the objectives of each component, consisting in the text preprocessing, opinion word extraction, target extraction and polarity aggregation, each of which is an objective of a system component, separated in order to fulfill some nonfunctional requirements.

For fulfilling the preprocessing requirement, the system takes as input a set of text files, representing subjective texts with opinions. The text is preprocessed and a set of annotated sentences are produced as a result, containing the part of speech tags and the dependency relations between the words. The features contained in a text must be from the same domain, because the system does not make any distinction between possible domains, and opinions words from different domains can have different polarities. The goal of fulfilling the preprocessing requirement is accomplished by proving a set of sentences, which contain all the necessary information required for extracting opinions from a text.

The extraction of opinion words and targets are correlated, a characteristic which the extraction component is designed to exploit. By using a set of annotated sentences as input, the requirement of extracting both opinion words and targets is fulfilled by the creation of a set of tuples, containing pairs of opinion words or targets. Based on the procedure used for extraction, the tuples may contain a pair of opinion words, a pair of targets, or a pair consisting of an opinion word and target. From the produced set, the opinion words can very easily be seen and used as well as the targets, the reason for constructing the pairs of different types, based on the extraction method used is to help backtrack the extraction process, in order to help improve its accuracy, by finding out the words which introduce noise.

The polarity aggregation requirement is fulfilled by the final component in the system, which assigns the polarities to all the opinion words and targets extracted, and including the polarity information in the tuples, with the purpose of delivering finally the opinion words and their targets contained in the texts feed as input, including their polarity which is mandatory for the opinion extraction process.

### Non-functional Requirements

The non-functional requirements represent the qualities of the system which are needed. They direct the design phase of the project and the constructed architecture, having a tremendous impact on the solution implementation, which gives the need for their presentation.

The extraction of opinions has numerous requirements, the most important lying in the accuracy of the results and the performance of the system, the most important of which will be presented shortly:

* **Performance**: The need for a fast and efficient system was motivated in the Problem Statement and Motivation section, and the objective of the creation of system to fulfill that demand was stated in the Objectives section. This means that one of the most crucial requirements lies in ensuring a fast processing time. This is translated in using the most time consuming process, which is the preprocessing part, as little as possible, by cashing the results of the component responsible for it. By eliminating this process from further usages on the same text, the system should be able to perform in almost real time.
* **Accuracy**: An opinion mining system is useless unless it can provide reliable information to the user, the whole purpose of such systems is to remove the garbage from the useful information. This requirement is the most difficult to address, because this domain is known for presented difficulty in terms of providing information that is correct all the time, due to the large variety of forms in which the information exists. This issue cannot be addressed only in the design of the solution, but experiments need to be conducted in order to evaluate it and then used to provide an increase in accuracy. The extraction of elements (opinion words and targets) along with the elements used for extraction provides the means for evaluating the data, with the purpose of increasing the accuracy. The evaluation of accuracy is opinion mining systems done by using two metrics, precision and recall.
* **Modularity**: The addition of modules should not cause issues nor require great architectural changes, as the results can be further used, or additional components can be used to help in the task of mining opinions. Adding more modules should be a seamless problem with as little implementation impediments as possible.
* **Scalability**: The domain of opinion mining is used to processes the enormous amounts of data available, so the system must be designed to handle any amount of data, especially considering that information constantly appears and hardly disappears, meaning its volume will constantly increase.
* **Compatibility:** As mentioned in the Objective section, each of the components has a role to fulfill, which is useful to other opinion mining systems. This means that each of them must be compatible with any other component, by constructing them independently and providing reusable output. The set of sentences produced by the preprocessing step should be stored, so any other systems can user them, as well as the extracted opinion words and targets. The usage of structured text files seems best suited for this task.
* **Reliability:** The need for a reliable system is translated in a need for reliable components. Each of the components must be able to effectively fulfill their purpose, and in the case of errors, attempts for recovery must be made when possible, continuing with the extraction process, by skipping unreadable files or data and providing adequate messages for the user.
* **Usability:** The system must be as simple to handle as possible for the user, since the aim of the system is to remove all the required human interaction needed for supervised systems. This means that any user should be able to use it, in order to find the opinion they desire, with a minimum effort for learning the system or operating it.

# Bibliographic Research

Bibliographic research has the purpose of presenting the objectives, techniques and acquired knowledge from researching and developing the project. First the work done in the domain of the solution will be presented, followed with the approaches similar to the proposed solution along with alternative approaches.

## Natural Language Processing

Natural Language Processing(NLP) applications are developed to cope with the problems related to human-computer interaction like information extraction, machine translation, relationship extraction and sentiment analysis. The solutions devised by NLP use language specific algorithms and are based on machine learning or probabilistic approaches. As Liu says in [4], there are no easy problems in this domain, because studying an arbitrary text for extracting a specific level of information detail is extremely difficult. NLP techniques rely heavily on machine learning algorithms, and even the most efficient ones, like support vector machine and conditional random fields only perform well on domain dependent texts and are language specific.

## Opinion Mining

Opinion mining, also known as sentiment analysis represents one of the most difficult tasks of NLP, consisting of classifying the polarity of a feature, sentence or document as positive, negative or neutral. The aim of the research done in this field is to determine the opinion on a product of a text writer, by using language rules and machine learning.

The document classification is the simplest form of sentiment analysis and the most widely used, and is used to determine if a document expresses a positive or negative opinion about an object, assuming that only opinions related to that specific objects exist in the sentence. For reviews the assumption is valid excepting comparative sentences, but it cannot be used for blogs and forums, as the opinion expressed most likely have numerous targets.

For sentence level sentiment analysis, the problem can be split into 2 sub tasks, determining if a sentence is subjective or objective, meaning identifying the sentences which hold opinions, called subjectivity classification and determining the opinion expressed by the sentence called sentence level sentiment classification. These 2 tasks are studied both separately and together. For composite sentences, especially comparative sentences, the possibility of multiple opinions on multiple products still exists, and constitutes a problem.

Feature based sentiment analysis, also known as aspect based sentiment analysis, is the finest grained approach, and is the only one providing the necessary detail of information needed for identifying opinions on particular product features, making it the most desired approach. The reasoning for level of details needed consists on the non-uniformity of opinions on different product features, meaning a product can have several good features and several bad ones, and obtaining their particular opinion polarities is much more useful and desired. At this level of granularity, the problem of opinions on different products is replaced by the problem of finding the product features. This new problem, called feature extraction has a higher degree of difficulty, and is studied extensively in [5].

## Opinion Mining Approaches

In the early years of opinion mining, before 1990, the sentiment analysis systems used manually lexicon-based approaches to offer a shallow analysis of texts, as no labeled data was generally available and the NLP existing systems were not sophisticated enough to allow large scale evaluation of texts.

Currently, for resolving opinion mining problems, there are two general directions based on the approach for resolving the classification problem. The most used one is called supervised learning, consisting on using a training set, and a machine learning algorithm to provide context dependent solutions, while the other one, called unsupervised learning does not require any learning data, and is based on syntactic and semantic language rules, providing solutions with far less context dependency. Due to the language dependent tools for NLP, both approaches are language dependent.

### Supervised Sentiment Analysis

The rise the existence of manually annotated data by researchers and the usage of several techniques, like the usage of rated reviews made available by reliable source like Amazon or IMDB, gave the resources needed for a new approach in this field, called supervised sentiment analysis.

From the first experiments done by [6], the results provided already exceeded the ones provided by all previous approaches, as the their research suggests, meaning the potential of the field was seen early, and it became the focus of the work done in this field.

The problem to be solved by opinion mining generally consists of classifying texts into two opposite classes, namely positive and negative, offering the possibility of machine learning tasks to be employed in its solving process.

The work done by [6] focused on using three machine learning techniques, specifically Naïve Bayes, maximum entropy classification and support vector machines(SVM) in order to determine if a review is positive or negative. They used a as a data source a corpus of reviews, taken from the Internet Movie Database, which offered a label training data, due to their existing ratings, which were converted into polarities, and human based classifiers as a baseline for experiments. The human generated baseline produced by their work offered at most an accuracy of 69%, proving the need exploring a better way than relying on human intuition. The comparison on the 3 machine learning techniques in using unigrams as features yielded similar results for all 3 techniques, all surpassing the human generated baseline, with SVM performing the best, confirming the effectiveness of machine learning techniques in solving opinion mining problems.

Another supervised approach which also focuses on the domain of extraction opinion from movie reviews, [7], aims to provide the necessary information for summarizing the reviews. The summary should contain sentences from the review which capture the author’s opinion, by selecting them from the text by creating a structured sentence list. The solution proposed consists of using a feature-opinion word list to record information about the opinions in the review and the features associated with them. The problem is decomposed into four subtasks:

* Identifying feature words and opinion in a sentence
* Determining the class of the feature and the polarity of the opinion word
* Identifying the relevant opinion words for each feature, obtaining feature-opinion word pairs
* Producing a summary using the discovered feature-opinion word pairs.

For the first task, WordNet [8] is used along with movie casts and labeled training data for generating a keyword list. The keyword list is generated by taking the most frequent feature words in the labeled data, including the movie names and people names as features. For opinion words, the 100 most frequent are taken from the training set and the rest being searched in WordNet, by searching for their synonyms. After the opinion words and features are found, grammatical rules are applied to identify valid opinion word-feature pairs, finally using them to construct the summary by collecting all the sentences containing them, identifying the polarity of the sentence from the opinion words and then listing each sentence in one of two categories *PRO* and *CON*. The objective of our work can be seen as a subset of the task of summarization, as our solution also identifies a list of opinion word-feature pairs, which can be used to construct a summary, similarly to this supervised approach.

### Unsupervised Sentiment Analysis

The unsupervised sentiment analysis is the focus of most of the current works done in the domain of opinion mining, as domain independence and language independence is a hot topic, due to the reusability property they provide.

The language independence is very hard to achieve even by not using a domain dependent training set, because almost all unsupervised approaches rely on syntactic and semantic rules, which are of course language dependent.

One of the first works was done in [9], where the conjunctions between adjectives, which are considered to be opinion words are used to calculate the polarity of opinion words. Opinion words joined by a conjunction must have a similar polarity, like for example in the sentence *The camera was good and small*, if we know that *good* has a positive polarity, we can infer that *small* also has a positive polarity in this sentence. The system proposed identifies the polarities of opinion words joined by a conjunction in the following stages:

* All conjunctions of adjectives are extracted from the corpus along with the relevant morphological relations.
* The information from different conjunctions is combined to determine if two conjoined adjectives are of the same or different orientation, resulting in a graph with orientation links between adjectives.
* A clustering algorithm is used to separate the adjectives into two subsets of different orientation.
* Finally the average frequencies in each subset are compared and the subset with the higher frequency is labeled as positive.

In our approach, the basis of the rules which use conjunction relies on the work done in [9], which proves the effectiveness of the use of conjunction relations to propagate the polarity between related opinion words, reporting accuracy over 80%.

In [10], an unsupervised learning algorithm is presented relying on a three-step method, which will be described. The objective of the work is to classify reviews by predicting the semantic orientation of the phrases in the review, using the adjectives and adverbs present, which can be considered as opinion words. The semantic orientation is calculated by using the Pointwise Mutual Information (PMI) between the given phrase and the word *excellent* minus the PMI between the phase and the word *poor*. The three-step method stats with the extracting of phrases that compose the review and contain adjective or adverbs, making an estimation of their orientation score and then using the average score obtained to classify the review. The association between words is calculated by sending queries to a search engine (specifically Alta Vista). The 3-grams used are a set of 5 patterns containing the part of speech of the three consecutive words. It is similar to our approach, as both consider opinion words to be formed by adjective and adverbs, which must be in a close proximity to the features which are nouns, the difference consisting on the way the patterns between the opinion words and nouns are considered, our approach relaxing the constraint, by allowing an additional word to exist between the opinion word and noun and using the dependency relations which offer a better solution for identifying patterns, as the order of words does not influence the relation between features and opinion words.

### Feature Extraction

Feature extraction can be seen as half of our objective, as our aim is the extraction of both opinion words and features. There are approaches which focus on this problem, isolating it from the extraction of opinion words, among which one proposed by [11], which aims to extract and rank product features from documents containing opinions. Their technique is based on the Double Propagation algorithm, similarly to our approach for extracting features, the difference being in the grammatical relations used by the algorithm for the extraction process. The algorithm is discussed in the Theoretical Foundation chapter extensively, so the only the technique used for ranking the product features will be explained here. The ranking technique relies on using Hyperlink-induced topic search (HITS), a link analysis algorithm that rates Web pages for the computation of relevance of features, used for determining the ranking along with the feature frequency. Bipartite graphs consisting of feature indicators on one side and features on the other are feed to HITS which computes the hub scores, which indicate relevant features, by summing up the estimated the importance of the feature identifiers related to each respective feature.

### Feature and Opinion extraction

There are several works similar to our approach, focusing on extracting both features and opinion words at the same time, some use techniques which use their own extracted features to find additional opinion words and vice versa, like the Double Propagation algorithm.

The work done in [12] proposes and uses the Double Propagation algorithm, which is also adapted and used in our approach to extract in a semi-supervised manner both opinion words and targets. The details regarding the algorithm are described in the Theoretical Foundation chapter. In this approach, only adjectives are considered to be possible opinion words and only nouns to be possible features, which is not sufficient to extract all opinions in a document, because adverbs can also form opinion words and well as pronouns can be used to reference product features. There are numerous approaches which also focus on also extracting opinion words using adverbs, including [10] and [13]. The extraction process relies on a set of dependency rules between opinion words (adjectives) and features (nouns), which are used by the Double Propagation algorithm. The dependency rules state that an adjective must have a dependency relation with a noun, which can be used for extraction opinion words using nouns that are features and extracting features by using adjectives which are known opinion words. The rules also include the case when adjectives or features are joined by a conjunction dependency relation, which can be used to extract opinion words by using known opinion words, and features by using known features. The semi-supervised nature of the approach comes from the usage of a set of seed words to bootstrap the propagation process, as the extraction of features relies on existing opinion words and vice versa, a set of possible opinion words must be provided at the beginning. This set of opinion words was also used in our approach until experiments shown that actually only 2 words, representative for each class (*negative* and *positive*) are needed for an efficient extraction.

The work done in [13] has the same objectives as our own approach, extracting both opinion words and targets at the same time by further developing the rules used in [10]. The feature extraction is done in a supervised manner, by using a training corpus to identify possible features which can be nouns or noun phrases, leading to the need for the creation of a training set for each domain. The opinion words are considered to be adjectives or adverbs adjacent to features, which are already extracted in a previous step. The inclusion of a predefined set of verbs that are considered to be opinion words is also proposed which include: *like, recommend, prefer, appreciate, dislike,* and *love*. We also considered using verbs for the extraction process, as they can surely help in identifying opinions, but a solution which extracts all opinions inferred by verbs is very difficult to implement as there are no known dependency rules for the extraction of opinion expressed this way, and using a predefined list can only offer a minor improvement, without constant updating based on new findings, which would span across multiple years, if we consider the seed words list proposed in [14], which grew to over 6000 adjectives which can express opinion words in the span of 10 years, as an indicator of the possible number of verbs which need to be found.

Another unsupervised approach, [15] , proposes the use of OPINE, an unsupervised extraction system that extracts from reviews fine-grained features and the associated opinion. The OPINE system is built upon a web-based domain independent information extraction system called KnowItAll, which by using PMI scores between phrases, estimated by using a web search engine hit counts, extracts explicit product features from a data parsed by Minipar [16]. The opinion words associated with the extracted features are then identified, by using the syntactic dependencies created by Minipar and 10 extraction rules similar to the ones used in our approach. The polarity of the opinion words is computed by the use a relaxation-labeling technique, which assigns labels to opinion words in multiple iterations, updating them based on the previous values and the features related to it. A constraint for the system is the requirement of knowing the product class, leading to the need of some user interaction during its usage. The usage of online sources makes the technique time consuming and dependent on continuous online connection, two constraints which we avoided by only using offline tools and caching the input of the system when possible, with the intent of making a viable real-time solution.

## Text pre-processing tools

An important step in the opinion mining process consists of preparing the data and filtering irrelevant or unreliable data, by cleaning, normalizing or transforming it, in order to obtain a text in a form that can be processed by any data mining algorithm.

A set of tools that can be used to fulfill this objective are provided freely by Stanford, in a form of a language processing group called Stanford CoreNLP [17].

The set of processing tools includes a part of speech tagger, a parser, a tokenizer and a lemmatizer, which will be described shortly.

### Part of Speech Tagger

The part of speech tagger delivers one of the basic needs for the rules used for the extraction process, consisting of setting the Part of Speech (POS) tags for each word in the text. The process of tagging takes into account not only the word, but also its context, which refers to the its relation with other words from the sentence. In [18], the different methods for tagging are presented, which include the usage of learning based transformation, Hidden Markov Models and the maximum entropy method, which is used also by the parser implemented in Stanford CoreNLP. The results obtained by this implementation, the accuracy which exceeds 97%, led to the usage of the CoreNLP package for the pre-processing component of our proposed system.

### Parser

A parser works out the grammatical structure of sentences which groups the phrases together. The one included in Stanford CoreNLP is a statistical parses, which uses the knowledge of language gained from hand-parsed sentences to form the analysis of new sentences. There are different implementations for different languages, while the English one is the most developed, there are also implementations for Chinese, German or even Arabic texts. The output of the parser consists of typed dependencies set, also known as grammatical relations, which are extensively used in the proposed rules used for the extraction of opinion words and targets. Details regarding its implementation can be found in [19].

### Tokenizer

The role of the tokenizer is to split the text in a series of tokens represented by words. The list of tokens can be then used for further text processing, like parsing or data mining.

### Lemmatizer

The lemmatization process is used to reduce the inflectional or derivated forms of a word to a common base form, life for example *is, am, was* become *be*  after the process or *computers, computer`s* become computer. It is similar in nature with the stemming process which uses a more crude procedure, effectively chopping the end of words with the hope of getting a correct base word and including the removal of derivational affixes. The lemmatization process used by this tool uses a vocabulary and morphological analysis of words for getting the base form of a word, resulting in a more reliable process. The result of the process allows for disregarding the numerous possibilities for a word to appear in a text and the usage of seed words or the predefined list of words which can be searched in the text.

## Lexical Resources

Lexical resources are databases which consist of one or several dictionaries. The can be monolingual, bilingual or multilingual, meaning words may be connected from a language to another by using bilingual links that ensure their equivalence by using multilingual notations. There are two major lexical resources used extensively in the domain of opinion minion, WordNet and SentiWordNet.

### WordNet

WordNet is a free tool that contains a network of meaningfully related words, which can be nouns, verbs, adjectives and adverbs. The words are grouped into sets of synonyms, each of which express a different concept. The groups based on the meaning of words do not use only the string of letters which form the word, but also their semantic relations.

The relations which are stored in this database include the super-subordinate relation (*hyperonomy*), which links general words, like *furniture* to more specific one like *bed*, forming a hierarchy of relations that go up to the root node called *entity*. Another relation included is *meronymy*, where links of the type part-whole are considered, like *backrest* and *seat*, where backrest is a part of a seat. The relations between verbs are also stored based on the manner of characterizing an event, for example *communicate* is linked to *talk* and *whisper*. The adjectives are grouped by in terms of antonyms, which can be also used to extract the synonyms.

From the various types of relations contained, the value of such a database is obvious in any data mining work, a reason for which it was considered to be used in our approach, for linking different product features together.

### SentiWordNet

The lexical resource called SentiWordNet, [20] is open source database, explicitly devised for supporting opinion mining operations. It is licensed to over 300 research groups, due to the value it can provide to the extraction process, by offering the polarity of adjectives, adverbs and verbs based on their context. It is an enhanced version of WordNet, because it contains annotations depicting the polarity for all the words contained in WordNet. The polarity of each word can be inferred for each word by considering their positive, negative and objectivity scores, which are stored in the database. The scores are depicted for each sense of the word, depending on the context stored, on a [0.0, 1.0] range. The three scores are obtained by combining the results produced by a committee of eight ternary classifiers, characterized by similar accuracy levels, but different classification behavior.

# Analysis and Theoretical Foundation

In this chapter the theoretical aspects of the design and implementation are explained, with the purpose of explaining the operating principles of the implemented solution. As the focus of the thesis is the extraction of opinion words and targets, the extraction of each will be discussed while presenting a proposed solution design to solve both problems together. The presentation stars with the domain of opinion mining to which the opinion word target extraction belongs, detailing the concepts associated with the domain. The basis of the solution is presented next, consisting of a set of grammatical rules between opinion words and targets, some of which are proposed by Liu in [12], and others being original concepts. The solution proposed is detailed next, including the algorithm used for the extraction of both opinion words and targets, followed by details regarding the enhancements discovered during experimentation and finally the evaluation mechanism used.

### Aspect Based opinion mining

First of all the definition of opinion mining is that *an opinion is a subjective statement, view, attitude, emotion or appraisal about and entity or an aspect of an entity from an opinion holder*, according to the work done in [14], [21] and [22]. From the definition we can see the two elements of an opinion, consisting on the word opinion word which gives the attitude in a sentence or the emotion and the entity or its aspect which is the target of the emotion. This means that in order to effectively extract the opinions from a text, both of these two elements must be extracted, which is the purpose of aspect based opinion mining. An entity is a concrete or abstract object which can be represented as a set of components, like for example a car, which has an engine, wheels and so on. Extracting the opinions of each component of a car is much more valuable that only extracting details about the whole car. For example, if a person is much more interested in the engine that the paint of the car, the negative opinions regarding the paint should not be as important as the positive opinions of the engine, meaning the weight of opinions regarding different parts of an entity are different. In order to extract opinions at the component or aspect level of an entity, the relation between each opinion word and entity aspect must be preserved, so the form of a tuple as output is used.

The tuple must have the following components:

* Aspect (also known as feature or target), which is the entity aspect targeted by the opinion.
* Opinion Word, the word which reflects the emotion of the writer regarding an entity aspect.
* Tuple Polarity, the polarity of the opinion reflected on the aspect, which is denoted in a scale ranging from -1 to 1, with values below 0 reflecting negative opinion and values above 0 positive ones.
* Sentence Index, the index of the sentence from where the opinion was extracted.
* Relation, the relation or link between the opinion word and target

The extraction of opinion words and targets can be done both in a supervised and unsupervised way, the latter being the focus of the thesis. Domain independent grammatical rules are used for extracting opinion words and targets as well as their relationship which has the form of a grammatical dependency between words. The polarity of a word can be extracted by using a lexical resource for opinion mining, which contains a database of opinion words and their polarities depending on their context and part of speech. The sentence index can be computed by knowing the structure of the whole text, meaning the order of the sentences, and the current sentence when performing opinion word or target extractions.

The usage of an unsupervised approach is motivated by the need of domain independent solutions, as the problem of opinion extraction is not specific to any domain, but a general problem existing in all domains with humans associated and where their feedback is important. The lack of need for creating training data is also a major advantage of unsupervised approaches, as it allows the solutions to be used immediately, without lengthy and error prone manual annotation processes.

### Opinion Word and Target Extraction Rules

For opinion to exist in a text, they must be represented by words, meaning there must be an opinion word for each opinion to exist. Each word in a text has a part of speech, so opinion words also must have a part of speech (POS tag). Considering the types of words which can reflect opinions, there is a limited list of POS tags which can belong to opinion words, more specifically, they can be represented only by either adjectives (good, bad, ugly) or adverbs (wonderfully, fine). In the case of targets, they can be either implicit or explicit, the former which also must have an explicit presence in the text. The POS tags of entities can also be of only two types, either nouns (phone, car, computer) or pronouns (it). Considering the structure of a sentence, a dependency relation exists between words, with the root being a verb, the dependency being a one-to one correspondence between every word in a sentence and at least one other word which corresponds to that element. For example, in the sentence *Sam has a good phone* the adjective *good* describes the noun *phone*. Considering the number of dependencies required to get from a word to another as a distance metric, the distance between adjectives or adverbs and nouns or pronouns is small, having a maximum value of two, meaning there is at most another word between them in the dependency path. The relation between opinion words and targets can be extracted by using the possible dependencies between opinion words and targets, for which Table 4.1 is devised, depicting dependency relations between opinion words and targets. In the case of multiple opinion words related to the same target, the opinion words themselves are also related by a grammatical dependency between the two adjectives or adverbs, which is a conjunction formed by either the words *or, and*, or by a comma (*,*).

In Table 4.1, the types dependencies for each possible combination, opinion word-opinion word, opinion word-target, target-target are represented as separate rules, which used by the proposed solution for the extraction of both opinion words and targets. They are also grouped based on the element extracted, which can be either an opinion word or a target and by the element used for extraction, also either a target or an opinion word.

Table 4.1 Opinion word and target extraction rules

|  |  |  |  |
| --- | --- | --- | --- |
| **Rule** | **Used** | **Obtained** | **Dependencies** |
| 1 | Opinion Word | Target | 1.1 2  1.2 1  1.3 3 |
| 2 | Opinion Word | Target | 2.1 11  2.2 10  2.3 12  2.4 27 |
| 3 | Opinion Word | Opinion Word | 3.1 5  3.2 6 |
| 4 | Target | Target | 4.1 6(1)  4.2 14  4.3 15 |
| 5 | Opinion Word | Opinion Word | 5.1 7(1)  5.2 28 |
| 6 | Target | Opinion Word | 6.1 23  6.2 24  6.3 22 |
| 7 | Target | Opinion Word | 7.1 16  7.218  7.319  7.4 20 |

In Table 4.1 we denote the POS tags of the words extracted as:

***NN*** = Noun, ***JJ*** = Adjective, ***RB*** = Adverb, ***PR*** = Pronoun,

***X*** = Any POS except if it is the same as the one which follows it.

Also we denote the relations between words with:

***MR-Rel*** = A relation between an opinion word and a target

***Conj-Rel*** = A conjunction relation between two targets or two opinions words (usually ***and***, ***or*** etc.)

Rule 1 is used to extract targets by using a known opinion word. The opinion word can be either an adjective or an adverb, and the target can be either a noun or pronoun. In the case of adjectives, the relation between them and targets, denoted by MR-Rel can be that of an adjectival complement, or adjectival modifier, while the adverbs can be related to pronouns by an adverbial modifier relation.

Rule 2 is also uses opinion words to extract targets, but the difference is in the distance considered in the dependency graph, in this case, there is an additional word in the dependency path, meaning opinion words are related to a word, denoted by “X”, which can be a verb for instance, which is in turn also related to a possible target, for example, the adjective can be the complement of a sentence while the target being a subject, related to each other by a predicate.

Rule 3 is used to extract opinion word by using known opinion words, and used the conjunction relation which exists between two adjectives or adverbs. In this case, the target of the known opinion word is also the target of the newly extracted one, and the tuple in the form required by aspect based opinion mining can be formulated.

Rule 4 is based on the same principle as Rule 3, but the elements joined by a conjunction being targets in this case.

Rule 5 is similar to Rule 3, but the two opinion words do not have to be directly related, they can also be joined by having a conjunction relation in the dependency graph.

Rule 6 is uses targets to extract opinion words, using the same element types as Rule 1, with targets of the type noun and pronoun being subjects or complements containing adjectives or adverbs which modify them.

Rule 7 is constructed on the same principle as rule 2, but with the elements being targets not opinion words.

There was an additional rule proposed initially, consisting of finding targets using targets which both have a conjunction relation on a third word, similarly to the relation between opinion words present in Rule 5, but the experiments showed that it was not effective in extracting targets, as most nouns or pronouns extracted in this way were not targets.

As mentioned before, Liu in [12], proposed the some of the rules in Table 4.1, namely 1.2, 2.2, 3.1, 4.1, 5.1, 6.2, 7.1 and demonstrated their effectiveness in aspect based opinion word and target extraction. The opinion words were considered to have only adjectives as POS tag, while targets being nouns, but the cases of opinion being expressed by adverbs also exist, as Turney presented in [10], where he specified rules for adjective and adverbs, also obtaining good results. By using the principle proposed by Turney, of opinion word expressed by adverbs, the rules containing adverbs were introduced, by using the same principles as the ones proposed by Liu. In addition to these rules, we also introduced new completely original ones, to cover the cases when the target is referenced by using the pronoun *it,* which can replace any subject in a sentence, obtaining the set of rules in Table 4.1. The impact of the rules can be seen in the Testing chapter, where the evolution of recall and precision can be seen.

Examples for each rule are presented in Table 4.2, where we can see that the distance between opinion word and targets is of at most two dependencies, with verbs being the middle word in the relation in most of the cases. The POS tags are set for each word related to the rule, serving in the depiction of targets and opinion words in the sentence. Some of the cases seem similar, but the underlying dependency graph is different, for example in the case of conjunction, the usage of the word *and* creates a different relation that the usage of a comma, thus leading to the need of all the rules in Table 4.1.

Table 4.2 Rule Examples

|  |  |  |
| --- | --- | --- |
| **Number** | **Rules** | **Example** |
| 1 | 1.1, 6.1 | The processor(NN) works fast(RB). |
| 2 | 1.2, 6.2 | The good(JJ) game(NN) was bought. |
| 3 | 1.3, 6.3 | It(PR) is working wonderfully(RB). |
| 4 | 2.1, 7.1 | The camera(NN) works(X) fine(RB). |
| 5 | 2.2, 7.2 | The ringtone(NN) included(X) is awesome(JJ). |
| 6 | 2.3, 7.3 | It(PR) works(X) fine(RB). |
| 7 | 2.4, 7.4 | It(PR) feels(X) comfy(JJ). |
| 8 | 3.1 | I have a good(JJ) and awesome(JJ) game(NN). |
| 9 | 3.2 | The game works great(RB) and smooth (RB). |
| 10 | 4.1 | The laptop’s display(NN) and keyboard(NN) are the best. |
| 12 | 4.2, 4.3 | Me(PR) and my wife(NN) are pleased with the product. |
| 13 | 5.1 | I liked the new(JJ), good(JJ) and cheap product. |
| 14 | 5.2 | The processor works fast(RB),silent(RB) and smooth. |

All the rules need an existing opinion word and the POS tags for all the words in the sentence to be specified, leading to the need for a preprocessing step and a component which provides known opinion words and targets.

### Double Propagation Algorithm

In this subchapter the solution for extracting all the opinion words and targets in a text is presented, which uses the rules specified in Table 4.1 and an extraction algorithm based on those rules.

First of all, the rules provide a way for extracting both opinion words and targets, but they all require either opinion words or targets, making them impossible to use independent of a component which provides known opinion word and targets. This is true if we do not consider reusing the output of each rule as input for another one, for example Rule 1 needs a known opinion word, which can be provided by Rule 3, 5, 6 or 7. This means that by considering the rules as a whole, a self-sustainable process can be implemented that can be used for the extraction of all opinion words and targets in a text, if we consider the rules as being sufficient to cover all the possible cases where opinion words are related to targets. Reusing the results is not enough to create the process, as initially no rule can produce an output as it has no input, so we still need an additional component to boot start the extraction process. There are two options, either another method is used to initially extract some opinion words and targets, which leads to a domain dependent and supervised solution, or a set of known opinion words or targets can be used. We decided to create a domain independent solution, so a set of known opinion words is used, denoted as seed words, as in contract to targets, the opinion words themselves are domain independent, even if their polarity is not, and this set of seed words is enough to start the whole extraction process.

The self-sustaining process is actually an algorithm which uses the rules iteratively, one at a time, repeatedly until no rules can be used to find additional opinion words or targets. The pseudo code of the algorithm is the following:

#### Double propagation Algorithm

Input: Seed Word Dictionary {S}, Processed text{T}

Output: All Features-Opinion Pairs {F}, All Opinion-Feature Pairs {O}

Constant: Objectivity Threshold {Th}

Function:

1. {O} = {S}
2. {F1} = Ø, {O1} = Ø
3. For sentence in T:
4. if( Extracted features not in {F})
5. Extract features {F1} along with the opinion words used by using R1, R2 with {O}
6. endif
7. if( Extracted opinion words not in {O} and opinion words objectivity < {Th})
8. Extract opinion words {O1} along with the opinion words used by using R3, R5 with {O}
9. endif
10. endfor
11. Set {F} = {F} + {F1}, {O} = {O} + {O1}
12. For sentence in T:
13. if( Extracted features not in {F})
14. Extract features {F2} along with the features used by using R4 with {F1}
15. endif
16. if( Extracted opinion words not in {O} and opinion words objectivity < {Th})
17. Extract opinion words {O2} along with the features used by using R6, R7 with {F1}
18. endif
19. endfor
20. Set {F1} = {F1} + {F2}, {O1} = {O1} + {O2}
21. Set {F} = {F} + {F2}, {O} = {O} + {O2}
22. Repeat 2 until size({F1}) = 0 and size({O1}) = 0

The objectivity threshold is a constant value, which depicts the maximum objectivity value for an adjective or adverb, which can be expressed in an interval from [-1,1].

The idea behind the threshold is to disregard adjectives or adverbs which are not actual opinion words. An adjective or adverb is considered to be an opinion word if its polarity, taken from a lexical resource, is above (in case of a positive opinion word) or below (negative) a certain threshold. This threshold ensures that objective words are not extracted, thus reducing the chances of noise propagating through the algorithm. The finding was triggered in the initial experimental phase, when we noticed its necessity since many adjectives express a property, not necessarily an opinion (first, other, long, etc.). Thus, they have to be pruned out from the set of opinion words, as they only induce noise (which can eventually, due to the propagation process, induce errors).

The processed text represents a preprocessed standard text with the operations of tokenization, ssplit, pos tagging, lemmatization and parsing performed, in order to obtain a structured list of sentences, containing the roots of words with POS tags attached to each one, with the purpose of providing all the necessary information for the rules to be applied directly.

Initially all the set of features is empty and the set of opinion words is populated with the seed words, which must be removed after the algorithm finishes, as they do not represent actual data from the text.

The collection of sentences must be iterated multiple times by all rules, in order for the rules to be applied with the new opinion word and targets found, each of which we denote as an iteration. We have a set of possible output data after each iteration, as all the rules have been applied to all the sentences, as after each iteration we can see as expanding the domain, decreasing the precision and increasing the recall, as the most common cases are extracted in the early iteration, while the more rare ones and additional noise is extracted in later ones, the extraction ending when no more opinion words or targets are being extracted in an iteration.

We have several steps in every iteration, corresponding to using one or two specific rules for the extraction. First rules 1 and 2 are used to extract features using the existing opinion words, which initially are seed words. The second step consists of extracting opinion words by using rules 3 and 5, checking the polarity of the adjectives and adverbs, in order to ensure the extracted words are actual opinion words. At this point we also have some features, if the first step was successful in the extraction process, allowing the third step consisting of using Rule 4 to extract additional features using them. Finally, the new found featured can all be utilized to extract additional opinion words by using rules 6 and 7. At each step, only new opinion words and targets are added to the set, because repeating the same process on a larger set of data, which includes the original one will of course extract duplicates, which must be filtered, in order not to alter the results and increase the extraction performance, because each found result will be used in two steps, being checked against each sentence, increasing the duration of the processing considerably. The best example for this case is the usage of a large seed word list versus a small one, which can be seen in the Testing chapter, where the running time can be seen to be increasing exponentially.

The beauty of the algorithm is in the usage of two separate extraction processes, each which requires data provided by the other, together, providing an unsupervised domain independent solution. The process is still language dependent due to the grammatical rules used, but separating the algorithm from the rules could allow for the creation of a solution for multiple languages, by providing a set of rules for each language.

In terms of concurrency, all the steps of the algorithm are dependent of exactly two other steps, meaning the creation a parallel version is difficult, but this is not an issue, due to the fast running time, as iterations are completed in almost real-time, as can be seen in the Testing chapter.

### Proposed Solution

The solution proposed to solve the aspect based opinion mining problem consists of creating an efficient and reliable system which can perform well on cross-domain corpora thus assuring an unsupervised approach to the problem of opinion mining. The solution devised is a parameterized system which can be tweaked and experimented in various conditions, whose overall architecture can be seen in Figure 4.1, with the details being described in the Detailed Design and Implementation chapter.

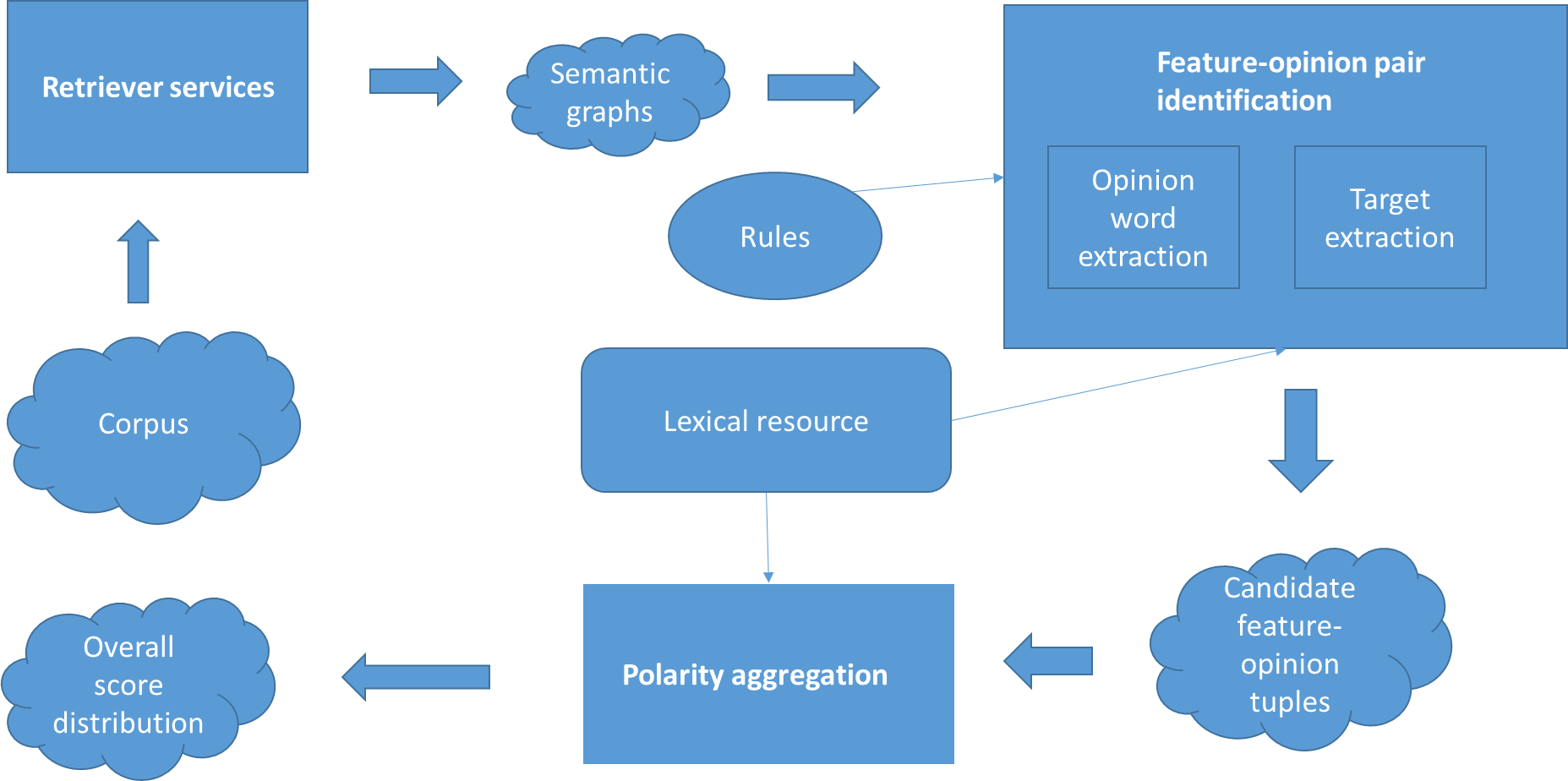


Figure 4.1 The architecture of the aspect based opinion mining system

The Retriever services generate the *semantic graphs* (or *syntactic trees*) from the given input corpus, representing the preprocessed text. This module handles all the tasks which involve a direct manipulation of the text documents. The transformations, which apply at sentence level, are: tokenization, lemmatization, part-of-speech tagging and syntactic parsing. First, each review document is segmented into sentences, which are used for discovering words in the tokenizing step. Lemmatization reverts the word to its base (root) form. The parsing step generates syntactic trees for each sentence, given the output of the previous steps. This syntactic decomposition will be used as input for the second main task of the system, the identification of feature-opinion pairs.

The Feature-Opinion pair Identification Component plays a central role. It extracts feature-opinion pairs using the double propagation algorithm already explained. Double propagation introduces much noise related to target extraction. To eliminate this kind of noise, a pruning technique which filters targets by their occurrence frequency is used. This is based on the idea that in reviews, the product or its features will occur more often with opinion words than other nouns. So the pruning is performed by counting the number of occurrences of each target after the extraction algorithm is finished and removing the ones which are not extracted at least t number of times, where t is a target frequency threshold.

The third component, Polarity Aggregation, performs the task of assigning polarity scores to the extracted opinion words and targets. Also it generates a polarity summary by aggregating the individual scores. We assign polarities to seed words using a lexical resource. Because a lexical resource usually contains multiple polarities for the same word, depending on the context, and because it is difficult to find the exact context for a given word, the resulting polarity will be the weighted average of all the polarities for that given word. After all the opinion words and targets are extracted, we assign new scores to existing opinion words and targets by *propagating* the polarities from the seed words to the newly discovered opinion words and targets.

The polarity assignment is based on the following rules:

* Opinion words discovered using previously assigned targets receive the same score as the target
* Targets extracted using previously assigned opinion words receive the same score as the opinion word
* Targets discovered using previously assigned targets receive the same score as the previously assigned targets
* Opinion words discovered using previously assigned targets receive the same score as the previously assigned opinion words

In order to maintain consistency, if the same target is discovered using different opinion words, the resulting score will be the average value of those opinion words.

The system constructed can extract opinion words and their targets, with the polarities populated on both, effectively completing the task of aspect based opinion mining

We specified creating a parameterized system, due to the number of possible configurations which can be easily made by changing one or more of the following parameters:

* Polarity threshold

The polarity threshold is a constant value, which can be provided by the user of the system before the start of each processing. The purpose of the threshold consists on eliminating noise generated by the double propagation extraction, by filtering adjective and adverbs which have a too small polarity to be actual valid opinion words. As can be seen in the Testing chapter, changing the polarity threshold yields significant changes in the generated output, the increase in precision and decrease in recall being directly proportional to the polarity threshold value. This means that it can be used to tweak the precision-recall ratio effectively.

* Target frequency threshold

The target frequency threshold is used after the propagation algorithm is finished, as any additional value, even if not an actual target can be used for extracting additional opinion words, which might in turn extract actual targets. This means that we postpone the filtering process until we get all the possible opinion words, and then initiate the filtering process. By utilizing this method, we avoid the negative impact of reducing the number of targets in the extraction process, but still eliminate noise. In the end, only the target extraction is influenced by this threshold, but the effects are similar to the ones of the polarity threshold, meaning it can also be used to manipulate the precision-recall ratio.

* List of seed words

The list of seed words can be used to give domain dependent information to the system. It can function with only two seed words, *good* and *bad*, but providing additional opinion words influences the extraction process. The list of seed words can be configured from an external file, meaning it can be preset, and changed only when needed, becoming a system parameter.

* List of POS tags used

POS tags are essential to the extraction rules, but the ones specified in Table 4.1 are generic POS tags, meaning each of them has a more specific classification, for example, a noun can be either a common or a proper noun, singular or plural. This gives us the opportunity to filter noise by using more specific from for each POS tag. These forms can also be configurable by using an external file, thus becoming a system parameter which can be tweaked based on the type of text.

### Evaluating Extraction

The proposed solution must be evaluated in order to prove its correctness and usefulness and in order for it to be comparable to other supervised and unsupervised solutions, the two fundamental metrics used in the opinion mining domain are used namely **precision** and **recall**.

Precision represents the fraction of relative instances retrieved while recall depicts the fraction if relevant instances retrieved.

Based on the predicted retrieved value and the actual value of an instance, that can be either a positive result, meaning a correct extraction, or a negative result, meaning an incorrect one, four classes can be used to classify the output, which are best depicted in the confusion matrix from Table 4.3.

Table 4.3 Confusion matrix for extraction

|  |  |  |
| --- | --- | --- |
|  | **Predicted positive** | **Predicted negative** |
| **Actual positive** | TP | FN |
| **Actual negative** | FP | TN |

The description of the representations in Table 4.3 is the following:

* TP = TRUE POSITIVE

The extracted element is an actual opinion word or target.

* FP = FALSE POSITIVE

The extracted element is not an actual opinion word or target

* FN = FALSE NEGATIVE

The actual opinion word or target is not extracted.

* TN = TRUE NEGATIVE

The word which is not either an opinion word or target is not extracted.

These classes can be computed from the extraction results and can be used to compute the precision and recall.

The formula for computing the precision is presentenced in (1), and the one used for calculating the recall is depicted in (2).

Precision = (1) Recall = (2)

The values are calculated in percentages the with ideal value of 100% for both values representing a perfect extraction, but which is almost impossible to achieve due to the way the two values are correlated. The methods which can improve one value generally influence negatively the other one, because recall is increased by expanding the results, including additional correct results, but by doing so, additional noise is also introduced which reduces the precision of the extraction.

# Detailed Design and Implementation

This chapter presents in detail design of the system proposed to implement the theoretical aspects presented in the earlier chapter. The aim of the chapter is to present a viable implementation of the presented concepts, and the technologies currently available used to facilitate it, ensuring that the functional and non-functional requirements are fulfilled. The overall system architecture of a larger system will be presented first, in which the proposed system is design to fit into, followed by a presentation which will go into details regarding the proposed system itself. The presentation of the system itself will be based on detailing each system module as a separate subchapter, explaining where needed the concepts and techniques used.

## The overall opinion mining system

This subchapter focuses on presenting a system designed to extract opinions named opinion mining system, and which we will refer to as OMS, where the solution presented in this thesis is intended to be integrated.

The system is composed of three components, along with several third-party systems integrated in the components. The three components, presented in order of the information flow are called Retriever Service, developed and presented in [2] by one of my colleagues and referred to as RS, Source-Target Pair Identifier, which is the focus of the thesis and further referred to as FOPI, and will be presented below in detail, and lastly the Polarity Aggregation component, developed by another colleague and presented in [3] and further referred to as PA.

The other components used will also be presented briefly, in order to facilitate the understanding of the overall system, as they were developed with the intention of a solving a bigger and more complex problem, and each represent a part of the implementation of the solution to that problem.

Even though all components were designed to be used together, they only depend on the output of each other, there is no additional dependency implemented, as they are also designed to function separately. Their independence was considered very early in the development phases, as it`s a very important requirement, considering the numerous approaches which can be used to replace the ones implemented in each components and the several technologies available to be used by them, which can offer significant changes to the end results.

All the communication between the components is done by in memory collection of objects. The main reason for choosing this type of communication over physical storage is the performance. The data required to be passed between the components is similar in size to the original input, and considering the objective of the solution, the input data is in general is large enough to make the storing time large enough to impact significantly the overall performance of the system. In the case of FOPI, considering the very fast running time, it could double in size. Considering the fact that the memory space needed does not increase after the first component, meaning there would be no additional problems which could be caused by the decision in favor of performance was taken very early in the implementation stages.

### Architecture

The general architecture of the OMS is presented in Figure 5.1, containing the main modules of each component, and their interaction, provides a minimal picture of the information flow, which is needed for understanding the system.

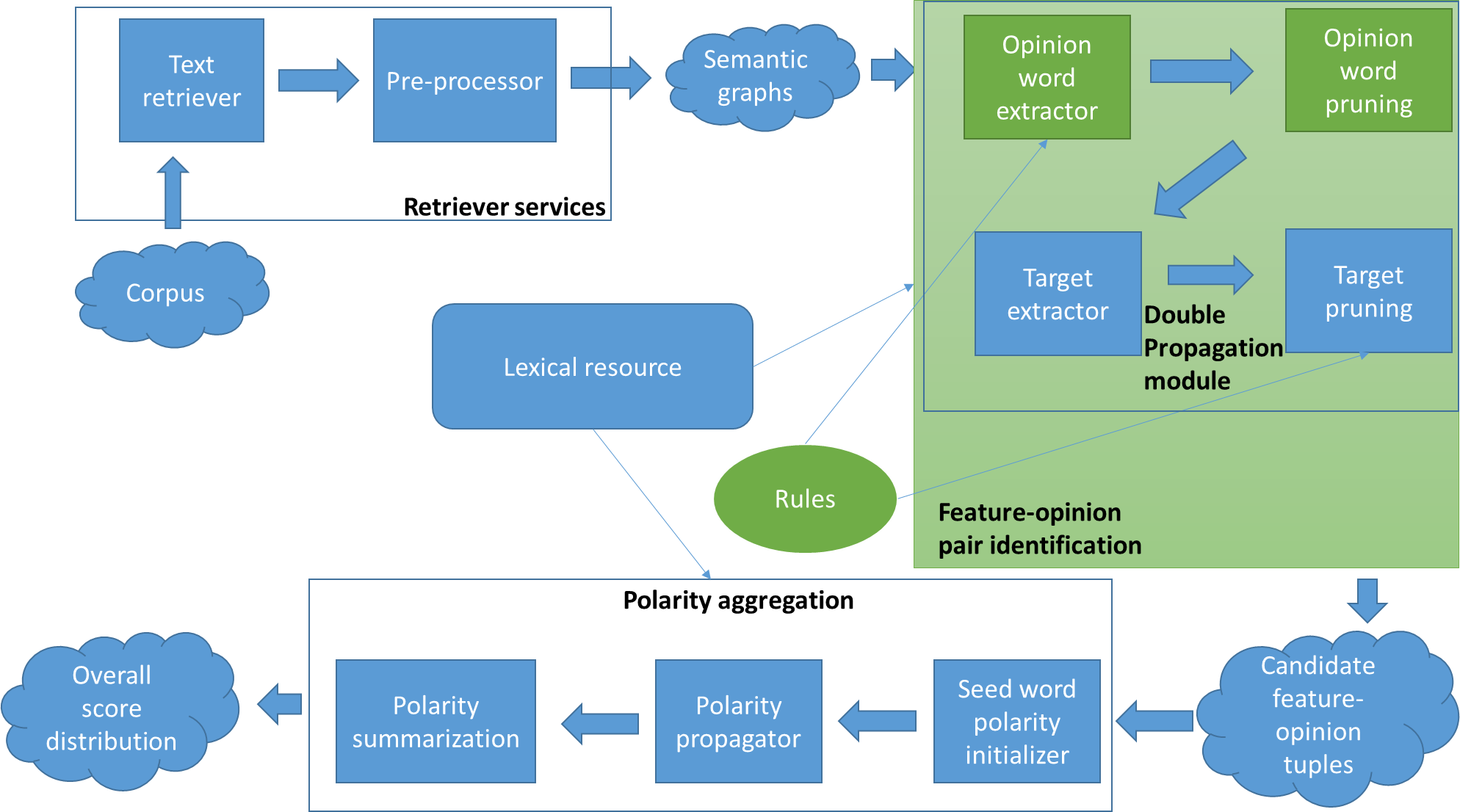


Figure 5.1 The architecture of the OMS system

The components and modules represented in green boxes are the focus of the thesis and will be further detailed, while the ones represented in blue are tools or components which are only used, the implementation being done by my colleagues and details in [2] and [3].

### Retriever Service

The retriever service component is presented in detail [2], and only the aspects regarding its purpose will be described shortly, with the aim facilitating the understanding the OMS system, and the problem solved by RS.

The role of the RS component is to extract from a user created text, usually considered a customer writing a review, details concerning the syntactic and semantic relations between the words in the text, specifically their POS tags and the POS tags of the dependencies between them, and offer them as a collection of semantic graphs to any other system.

### Polarity Aggregation

The details concerning the implementation of this component are presented in [3], and only a brief overview of its purpose will be further presented, facilitating a minimal understanding of it, and its place in the OMS system.

The PA component requires a collection of source-target pairs to be provided as input, and using a lexical resource, assigns polarities to each member of all of the Source-Target pair.

### Polarity Service

The polarity is a module implemented by a colleague, and is not included in any of the three components. It is implemented as a replaceable third party service, used for obtaining polarity values for features or opinion words. The details regarding its implementation can be found in [3].

The details of its integration with the FOPI will be presented in detail during the presentation of the opinion word extraction module.

## Feature-opinion pair identifier

This subchapter focuses on presenting the component feature-opinion pair identifier and its design details, showing how the final architecture of the subcomponent was reached and detailing each module in the component separately, going from the high level modules to the lower ones, layer by layer. The independence of the modules at the same level will be visible when presenting them, as well the common lower level modules, revealing the high degree of reusability.

FOPI is the component designed with the purpose of extracting opinion words and features from a text with additional information regarding the words and their dependencies, specifically with the words POS tags known. The implementation of FOPI considers such a text to be provided as a collection of semantic graphs in order considering the implementation of the RS as well and provided a viable integration in the OMS system.

The dependency on semantic graphs is deeply rooted in a low level module called Extraction Service, but at the same time, it is also contained there. Additional details regarding the usage of semantic graphs are presented during the detailing of the respective module. This ensures that there is a clear separation of the implementation designed for the OMS system, from all the other modules which are designed for general use, allowing the developing of a replacement or additional module, for the usage of the same system providing different types of inputs.

This component does not rely on any online or external tools, with the exception of the polarity service, which could also be integrated inside if needed as the only reason of its independence is its reusability, which is applied in the PA component. The polarity service is also not a required tool, as FOPI works without it, but the results provided are generally lower, as can be seen in the experiments done. This independence of third party tools and internet usage, ensure a much more stable and more importantly a reliable system.

The FOPI overall architecture was designed to be modular, as can be seen in its representation in Figure 5.2, with the two main components, developed separately, Opinion Word Extractor and Feature Extractor being detailed separately in their own subchapter.

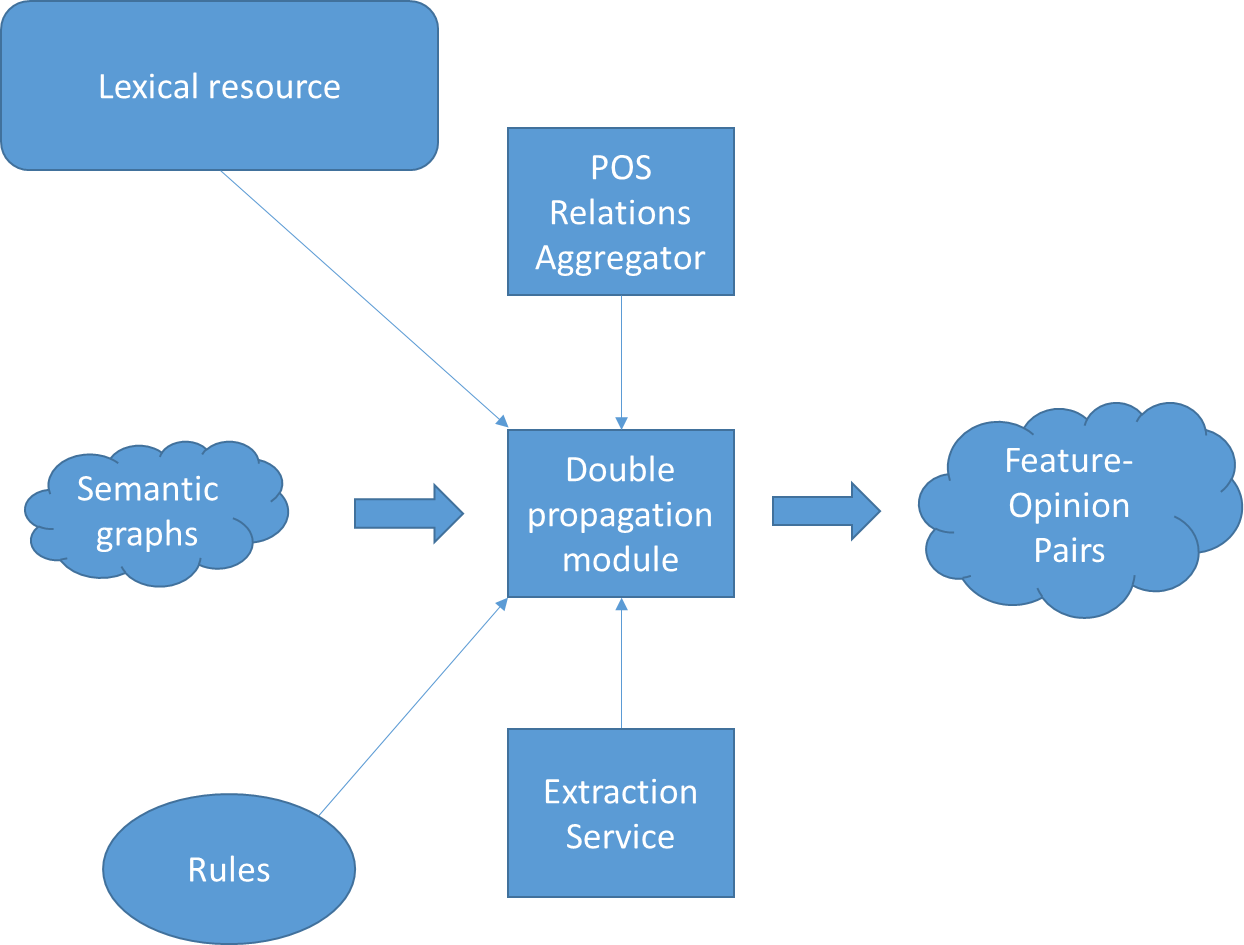


Figure 5.2 The architecture of the FOPI component

The implementation of FOPI architecture presented in Figure 5.2 can be seen in the UML diagram shown in Figure 5.3, where each of the class and interface used are depicted, and the relations between them can be clearly seen, relations which can be seen to respect the proposed architecture.

We will not go into details about all the classes, as the modules using them are described thoroughly, so an short explanation of the role of the most ambiguous named will be done.

The Algorithm Runner represents the manager of the application, containing all the input and output files, directing and coordinating all the components of the entire solution.

The Initializer handles the configuration of the system. As mentioned the solution has multiple configurable parameters, which must be read from an external file, a task done by this class.

The DoublePropagationAlgorithm represents the physical implementation of the DoublePropagationAlgorithm, being constructed according to the pseudo code described in the Theoretical Foundation chapter.

The NLPService handles the integration of the StanfordCoreNLP tool.

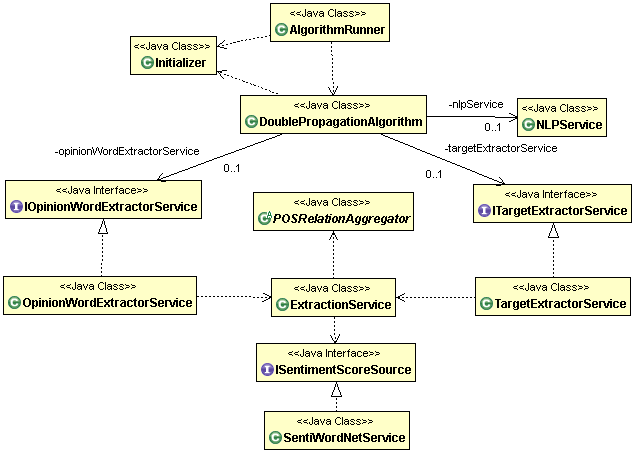


Figure 5.3 FOPI module UML

### The Data Model

In this subchapter the data types which are used by the modules are detailed, in order to help understand the tranformations the information goes though in the modules.

* Semantic graph

Represents a sentence processed by Stanford CoreNLP, in the form of a directed graph, with words as nodes and word dependencies as edges. In nodes the information regarding the POS tag of a word is located, while in the edges the information regarding the POS tag of the dependency is located. Using the concept that a picture is a thousand words, in Figure 5.4 the structure of a semantic graph can be seen for an example consisting of the sentence *HP, located in New York, creates and repairs computer products*.

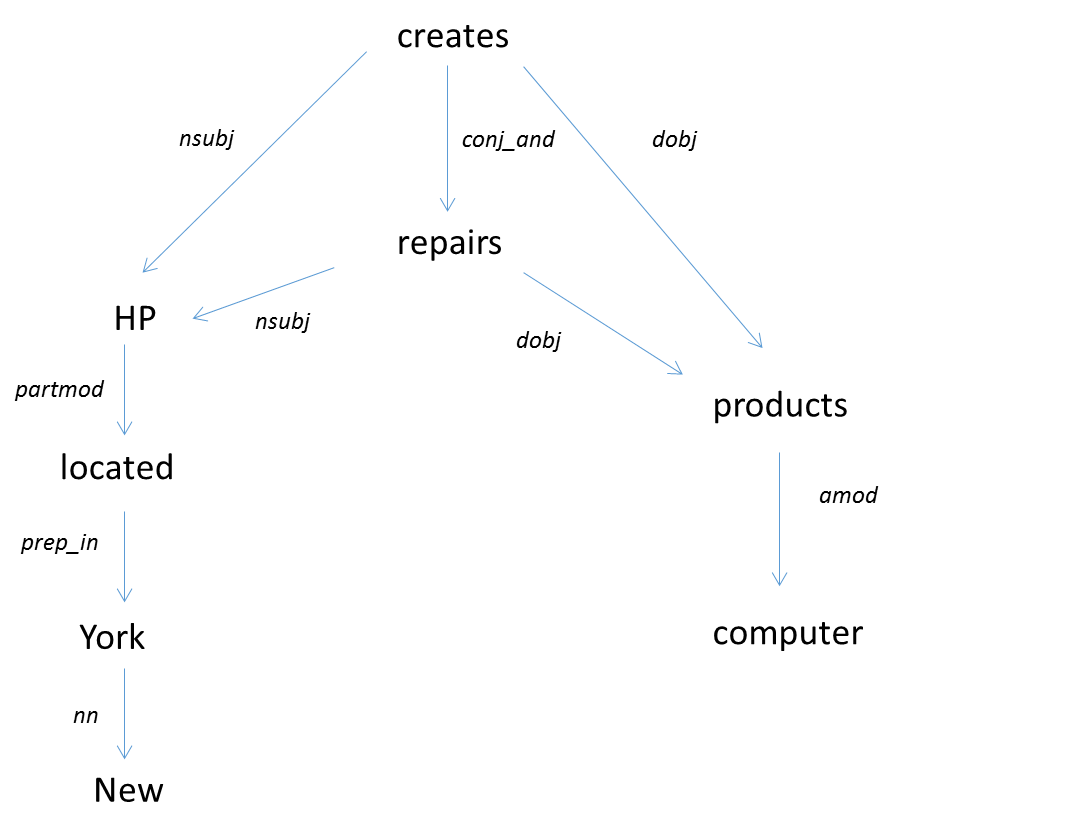


Figure 5.4 Graphical representation of a semantic graph

* Word

Represents a word from a sentence, as its name sugests, but contains additional information required by the system, which includes the POS tag, the ElementType, the opinion polarity extracted from SentiWordNet to which we refer to as sentiWordScore, the sentenceIndex, representing index of the semantic graph in the collection of semantic graphs set as input for the FOPI module and the number of times it has been found by extraction rules, even thought in the end duplicate findings are filtered, the information is valuable for the user.

* ElementType

An enumeration used to depict if the word extracted by the system is an opinion word, a feature or neither.

* Tuple

Represents a Source-Target pair, which can be an Feature-Opinion, Opinion-Opinion, Feature-Feature or Opinion-Feature pair, depending which word was used for extraction, denoted on the left and the word extracted denoted on the right. It contains two Words, one representing the source and one the target in the pair, and a TupleType.

* TupleType

An enumeration to depict the type of the Tuple, which can be Pair or Triple.

* Pair

A type of Tuple which contains Source-Target pairs constructed from rules 1,3,4 or 6, to which we reffer as rules without „X” words. In addition to the Tuple attributes it also contains an attributed depicting the relation type between the source and the target.

* Triple

A type of Tuple which contains Source-Target pairs constructed from rules 2,5 or 7 to which we reffer as rules with „X” words. In addition to the Tuple attributes it also contains another Word deppicting the „X” word and two relation types, one between the source and „X” word and the other between the „X” word and target.

* Evaluation Model

Used by the evaluator modules and depicts an annotated word from a annotated text used to evaluate the system. It contains information regarding the word, the sentence where it was found, the semantic graph index of that sentence and the mannualy assigned score.

### Double propagation module

The double propagation module is the implementation of the Double Propagation algorithm presented shortly, created with the intention to be as flexible as possible. The flexibility is ensured by separating the algorithm steps from the necessary data processing done in each step.

The double propagation algorithm is inspired from the one proposed in [12] for solving the same problem, but adapted for the usage of the rules presented in Table 4.1.

One of our original additions to the algorithm includes using a filtering condition on opinion words based on their polarity when extracting them. This filtering is explained in detail in the Enhancing Opinion Words Extraction subchapter.

The usage of pairs of feature-opinion and opinion-features is also an original concept developed during the thesis creation, with the scope of providing additional information to the user concerning the connection of the opinion words with the features, information which cannot be retrieved by similar systems in the domain, and which is very valuable also evaluation process of the system.

The input of the algorithm consists of a collection of semantic graphs, which must be set at the beginning of each usage, from which a collection of tuples are generated as output, by using Opinion Word Extractor and the Feature Extractor modules.

The seed words are used to boot start the extraction process, by providing a list of known opinion words, which might be present in the text. During experimentation, the seed word list size is determined to be of no real importance to the performance of the algorithm, as can be seen in the Testing chapter, and can be reduced to two words, which are enough to find at least one feature, starting the double propagation algorithm. Due to the possibility of the algorithm running without a list of seed words, as the two values can be embedded, the seed words are regarded as a parameter to the algorithm rather than required input.

The output consists of two different collections of data, a collection of features and a collection of opinion words, which represent the extracted opinion words and targets.

In the architecture shown in Figure 5.5, the usage of the Opinion Word Extractor and the Feature Extractor modules can be seen, as well as their mutual independence, ensured by the usage of intermediary data, generated by each of them in turn.

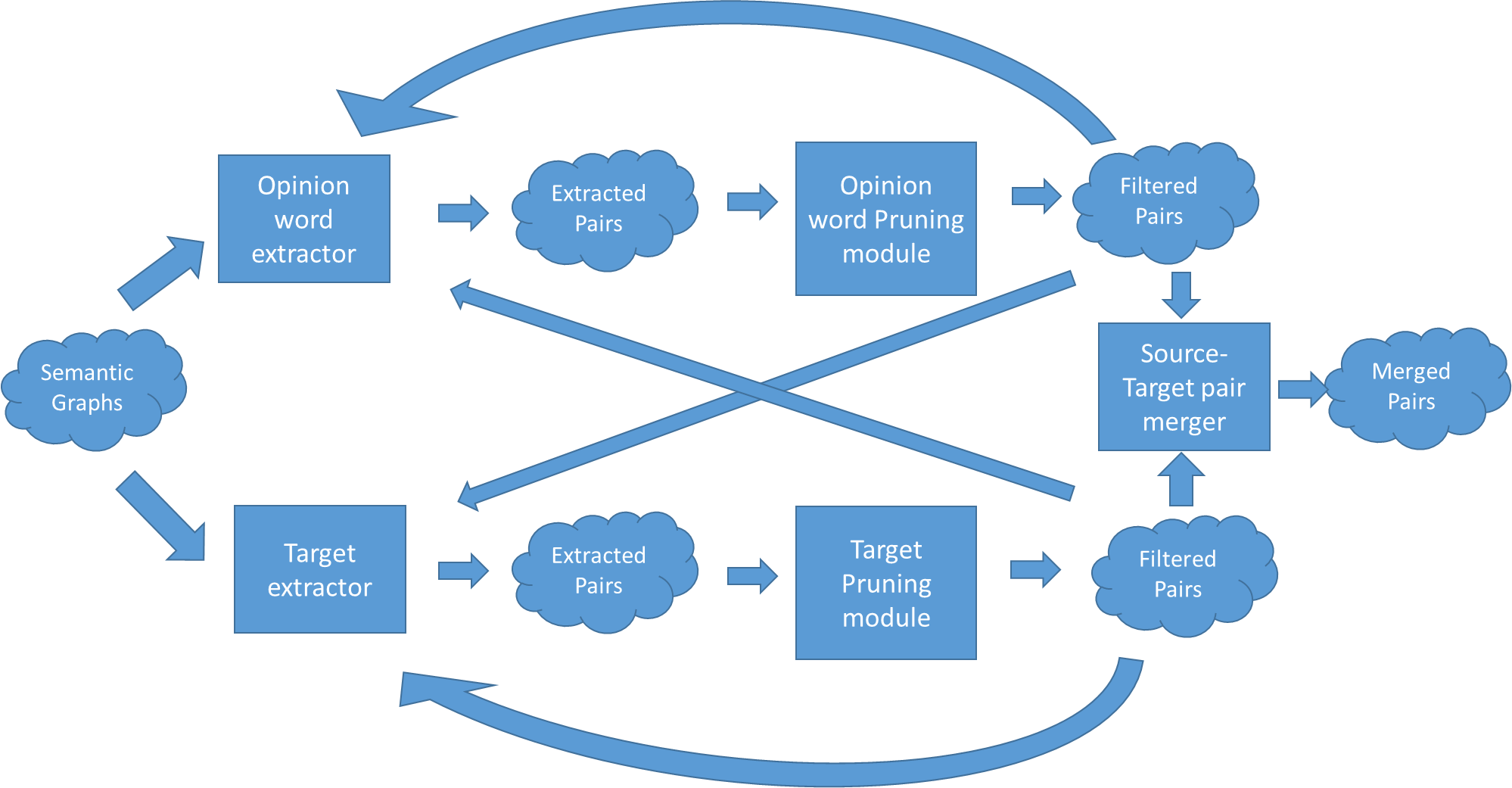


Figure 5.5 The architecture of the Double Propagation module

The generation of the output is done according to the algorithm proposed, meaning there is a version of output at each pass though the input, to which we refer to as an iteration. We saw this detail as a source of information worth investigating, for a possibility of improving the system, leading to the development of a functionality of providing the output at any iteration. The results of the investigation can be seen in Testing and Validation chapter, where the value of the functionality can be clearly seen.

An important detail regarding the implementation of the module is the extensive usage of the functionalities provided by Sets. The collection of source-target pairs, meaning the output data is provides as sets, as well as all the other intermediary data used by the module. In the early design phases, the usage of sets was decided in order to treat as effectively as possible the condition of only extracting new pairs in each algorithm step, and it seemed to solve the problem of duplicate features or pairs. In fact, the problem of duplicate feature and pairs was partially solved, as no duplicate pairs were extracted due to the selected implementation, which provided a significant improvement of the algorithm running time and the system performance, as each extracted pair is reused by the algorithm at each iteration for extracting additional pairs.

The complexity of the pairs given by the different type of pairs which can be extracted, which are explained in detail in the Extraction Service subchapter, created the need of checking each element of a pair against all the others, in order to completely eliminate the duplicates. This functionality is implemented in the Output Service detailed in [2], due to the importance of the additional information which is provided by a pair, which can be used for further improvements of the overall system. For example, if an opinion word is extracted using two different rules, the chances of the extraction being correct increases.

The decision of using a more relaxed filter of duplicates in this module, leads to the need of additional output processing to be done by the output user, which is handled by the Opinion Word pruning module and the Feature pruning module.

### Opinion word extractor

The Opinion word extractor module is designed to handle the task of extracting opinion words, as its name suggests, incorporating the rules 1,2,3 and 5 presented in Table 4.1.

During the creation of this module along with the Target Extraction module, a lot of emphasis was put into make them scalable. The biggest concern was making the implementation of additional rules to be as simple as possible. The solution we came up with was making each rule as a method, with the similar input data and output data. This allows for interchanging and combining rules. For example, rules 1 and 2 provide the same output data type and require the same input; just the processing done differs, allowing us to combine them into a single method which we can provide to other modules, making the connection simpler and looser. The same is done with rule 3 and 5, and the flow of information can be seen in Figure 5.6.

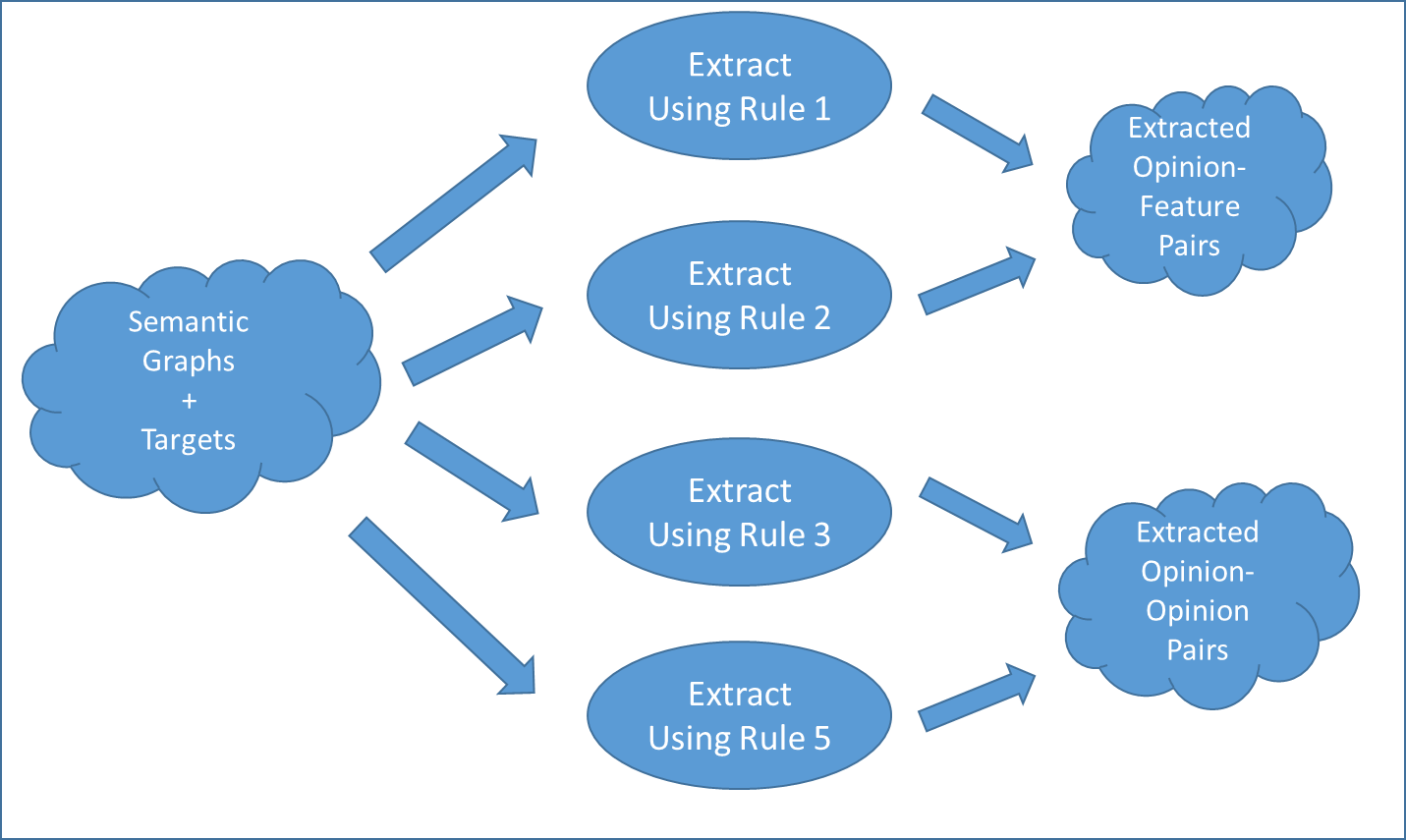


Figure 5.6 Opinion word extraction flow

As can be seen in the Figur5.6, all rules need semantic graphs and targets as input for extracting pairs. A semantic graph contains the words processed by the rules, their POS tags, their dependencies and the dependencies POS tags, and opinion words are needed in order to extract opinion words. There is additional data passed to the rules, as can be seen in the signature of a method implementing a rule presented shortly, but we don`t consider them part of the flow of information, but part of the rule implementation itself. We provide them as input parameters due to the generalization done, in order to obtain a general implementation for each rule. For example, Rule 1 is used to extract opinion words, but by changing the input parameters, we can change it to try extracting targets.

The signature of Rule 1 is the following:

**public** Set<Tuple> extractOpinionWordsUsingR1(**final** SemanticGraph semanticGraph, **final** Set<Word> targets, **final** Set<Tuple> existingOpinionWords, **final** ElementType targetType, **final** **int** semanticGraphIndex)

Explaining the signature seems mandatory for fulfilling the purpose of explaining the rule implementation and allowing the rules number to be expanded easily by anyone, while maintaining the implementation consistency.

The Tuple data type is explained in the Data Model subchapter and the Set <Tuple> represents the Source-Target pairs extracted. The targets are a set of Word elements also explained in the Data Model subchapter and represent the already extracted features used to find opinion words. The existingOpinionWords represent a set of Source-Target pairs containing already extracted opinion words, which are used to filter duplicates which may be generated by the extraction process. The targetType represents the type of the extracted Word in the Source-Target pair, always containing the value of OpinionWord for the opinion extractor rules, while for feature extraction rules having the value of Feature. Finally the semanticGraphIndex is used only for testing and evaluating purposes, as it contains the index of the sematic graph in the input data, as Sets do not contain an ordered collection, and also the content of the semanticGraph collection could be changed prior to the evaluation time, so providing a reliable index is preferred.

All the extraction logic is decoupled from this module and implemented in the Extraction service module, meaning in this module only the appropriate call for the extraction module is made, consisting in specifying the POS tags of the word to be extracted, the POS tags of the word used for extraction and the dependency POS tag, exactly as specified in the rules in Table 4.1. The POS tags are taken from the module POS relations aggregator, where the mapping from the Stanford specific POS tags to the POS tags specified in the rules table is made.

The implementations of rules 1 and 3 is just a simple call with the appropriate parameters to the Extraction Service, but for rules 2 and 5, additional logic is required, which we denote as the usage of an *X-Filter*.

The X-filter consists of finding the word mark with “X” in the rules table, which is basically almost any type of word, with the required dependencies to the opinion word used for extraction and the feature to be extracted to which we refer to as POS\_MRRel. For implementing this search, first a call to the Extraction Service is made getting any word with a relation of POS\_MRRel, and then for each word returned, we make a second call to the service, similarly to the calls made by rules 1 and 3. This means that we have to iterate over the collections much more than when using the simpler rules, and that implementing rules with a larger distance (number of intermediary words) between the opinion words and the targets would impact the performance greatly, as the number of collection iterations increases exponentially.

### Opinion word pruning module

The Opinion Word pruning module was designed to handle the problem of duplicate opinion words extracted, a problem which was discovered during the experimentation phases. The origin of the problem is the usage of Source-Target pairs as both input and output to the Opinion Word Extractor service and the Feature Extractor service and relying on the comparison of the entire pair, with a more complex logic to handle duplicate elements on either end of the pair. A lot of time was dedicated to solving the problem using the *Equals* and the *HashCode* methods provided by the Sets in Java, but ultimately, the hash codes generated by the pairs differed even if the feature and opinion contained in it were the same, leading to the usage of a much more reliable solution, consisting of extracting each element in the pair, the opinion word and the target, and comparing them with all the other elements in the other pairs of the same type. This solution was implemented in this module, separately from the extraction process as the logic of finding two matches to the exact same opinion word is a bit complex and to allow the flexibility of using duplicates if desired by other modules or implementations, as duplicates can be used to raise the confidence level of an opinion word being extracted correctly.

The implementation of the filtering process consists of the following actions:

* handling seed words separately
* extracting each opinion word from the Source-Target pairs, along with the semantic graph index and setting their number of instance to 1
* for each opinion word extracted, it is compared to every other opinion word extracted
* if an opinion word matches another opinion word, it is marked as a duplicate and the number of instances of that word is increased

The handing of seed words consists of removing all seed words from the resulting Source-Target pairs, as there are not extracted opinion words from the text, but words used to boot start the extraction process, as explained in the Theoretical Foundation chapter.

The semantic graph index is used to indicate the place in the text, from there the semantic graph was extracted and helps in the identification of identical opinion words. An opinion word is considered to be identical to another opinion word and thus a duplicate if it has the same word value, for example *good*, the same POS tag, and the same semantic graph index, meaning it’s the same word from the same sentence, which is very important, as opinion words are surely repeated thought the text, especially common ones, like *good*. When manually annotating the evaluation texts, we found that these conditions are sufficient for determining duplicates in the case of both features and opinion words, as there is only one instance of each opinion word related to a target in each relevant subjective sentence, and the repeated used of the same opinion word does not yield a better score, it just means that there is noise in the input text, as extracting the word *good* which is repeated 10 times in a sentence for example does not yield any benefit for the opinion mining process, but on the contrary, it is detrimental as one opinion influences the results greatly, by having the weight of several ones.

The number of instances is provided for flexibility purposes and is used by the Feature Pruning Module for implementing the target filtering approach by using a frequency threshold, detailed in the Theoretical Foundation chapter. It also provides important information by containing to the number of duplicates of each extracted word, which can be very useful to the system`s user.

### Feature extractor

The Feature Extractor was not the focus of this thesis and only the most important information for understanding it will be presented here, additional details can be found in [2].

This module is responsible for the extraction of features from the text. It is similar in nature to the Opinion Word Extractor, as it also implements the rules detailed in Table 4.1, and has the same input and output types. The reason for the similarity is the consistency and reusability, as even if the implementation was done separately on the two modules, we can work interchangeably on the addition of new rules, meaning further implementation can be very easily made, making the system scalable.

It is not directly dependent on the Opinion Word extractor, but they rely on each other for the extraction of opinion words and features respectively, as the output of one is the input of the other, a process which goes on until one of the modules does not generate any new output. This means that both modules require being functional and the results of one impact heavily the other. This can be seen in the results of the experiments done, as the changes in the precision and recall on the opinion extraction can be seen to be similar to the ones in the target extraction.

This module also depends heavily on the Extraction Service for the actual extraction of the features from the text, as only the implementation of the rules is done in this module. In Figure 5.7, a diagram similar to the one depicted in Figure 5.6 contains the information flow of this module.

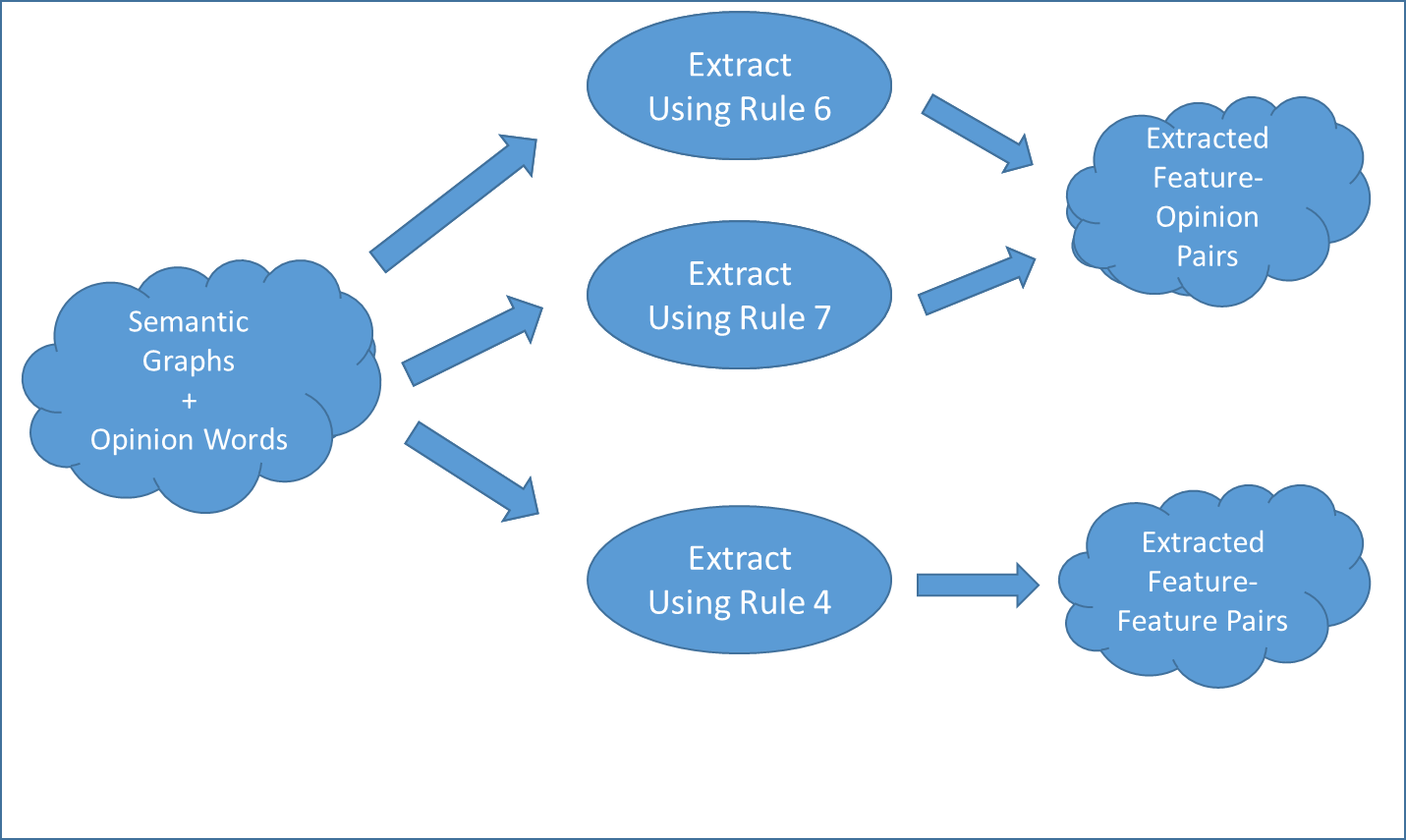


Figure 5.7 Feature extraction flow

### Feature pruning module

The Feature Pruning module is not the focus of this thesis and is presented in detail in [2], but the most important information will be presented shortly, for understanding its place in the system and the flow of information.

The purpose of the module, as the name suggests, is the removal of duplicate features, which influence the extraction result greatly, as each duplicate influences the score assigned to a feature, a score which is computed by aggregating all the scores assigned to the opinion words associated with the respective feature. Details regarding the score distribution can be found in [3].

The implementation of the modules is similar to the Opinion Word pruning module, due to consistency reasons, as the output and input of the two modules is of the same type, the difference being that one, the Opinion Word Pruning module, processes words which are considered opinion words while the other, the Feature Pruning Module deals with features.

The target frequency threshold presented in the Theoretical Foundation is also implemented in this module, and the details and results of the experiments on its usage can be also found in [2].

### Extraction service

The Extraction Service is the “hardcore” part of the FOPI component, as it is the module which handles the extraction process of words from the semantic graphs and the creation of Source-Target pairs, according to the requests made by the Opinion Word Extractor and the Feature Extractor modules.

It is implemented as a data processor, meaning there is no internal data stored, it requires no instantiation, it only has a collection of methods which process the input data in various ways, which will be explained shortly.

The implementation became quite complex, as it was made as general as possible, for handling both the extraction of opinion words and targets and will be presented from going from the requests made by the extractor modules, to the effective extraction of opinion words and targets respectively.

The need for explaining the meaning the parameters of the methods provided by the service along with the processing done is considered necessary, in order to allow the understanding of the implementation.

There are two main calls made to this service, one for the extraction of a Pair and one for the extraction of a Triple. The definition of both Pair and Triple can be found in the Data Model subchapter, along with the other structure used.

The call for a Pair extraction must provide:

* a semantic graph
* a Set of Word objects
* a GenericRelation type
* the POS type of the source word
* the POS type of the word to be extracted
* the ElementType of the word to be extracted
* a semantic graph index

The source word is the opinion word or feature used to extract a feature or opinion word respectively, each of which is denoted as “word to be extracted”.

The semantic graph contains all the details from the text required for extracting an opinion word or a feature. The structure of a semantic graph is basically a graph, like its name suggests, with the vertexes being word and the edges being word dependencies, so in order to use a word to extract another word, we must find the vertex corresponding to the extracted word, and then get all the edges connected to it, extract the vertex at the other end of the edge and then us the POS Relations and the parameters sent by the Opinion Word Extractor or the Target Extractor to see if the word fits in any of the rules defined in Table 4.1. The POS in the semantic graph are not the same as the ones defined in rules table, but the ones retrieved by the Stanford CoreNLP, so the POS relations aggregator is used to map the specific Stanford POS tags to the GenericRelation which contains POS tags defined in Table 4.1. The details regarding the mapping can be found in the POS Relations Aggregator chapter.

The semantic graphs contain directed edges for dependencies, but we do not take into account the direction in rules, so we search both incoming and outgoing edges from each vertex, basically doing two iterations for both types of edges.

The ElementType is used to denote the type of word extracted, which can be OpinionWord, Target or None, depending on the extractor module used, where None represents the word “X” from the rules table. This property is needed to distinguish between the types of words which can be extracted, as comparing POS tags requires additional processing, and for further developments, the POS tags can be common between different types of extracted words.

The SematicGraph index is just passed to each word extracted, in order to provide the sentence from where a word was found to the user, for testing purposes mostly, but can also provide valuable information to the user as well.

The set of Words objects represents a set of already extracted opinion words or targets, which are used to filter duplicates from the extraction process.

For the implementation of the Triple, the logic becomes a little more complex, due to the “X” word in between the opinion word and the feature, as can seen from the signature which contains the following parameters:

* a semantic graph
* a source word
* the dependency between the source word and the “X” word
* the “X” word
* the direction of the dependency between the source word and the “X” word
* the POS of the dependency between source word and the “X” word
* the POS type of the source word
* the ElementType of the word to be extracted
* a semantic graph index

The additional parameters all correspond to the addition of the “X” word in the rules table. The implementation of the rules containing an “X” word induces additional iterations though the semantic graphs. The Extractor Service expects the “X” word and the dependency between it and the source word to be provided, so the extractor modules have to search the semantic graph first for any potential “X” words, and the call the Extraction Service for each word found, with the above signature, which means two iterations, one for each outgoing edge of the source word, and one for each incoming edge. After we have first part of the rule, meaning the dependency between the source word and the “X” word, we need to extract the words matching the second part of the rule, the dependency between the feature or opinion word to be extracted and the “X” word. The implementation of this second part means and additional set of iterations on the incoming and outgoing edges of the “X” word, searching for word with the POS tags matching the POS tags of the word to be extracted passed in the signature. There is an additional requirement in the set of rules containing “X” words, consisting on the equivalence of dependencies between the source word and the “X” word and the “X” word and the word to be extracted, which is implemented by first grouping all the possible dependencies of each rules under a common POS tag, obtaining generic POS tags for the dependencies of each rule. After we have the generic POS tags for each dependency extracted, it is easy to compare them and decide if they are equivalent.

In the case of Opinion-Opinion or Feature-Feature pairs, a word might extract itself, as they surely satisfy the rules, as they are both surely related to the same “X” words and with the same dependencies and POS tags, so a filter for these type of extractions was implemented.

When checking the extractions of the Pairs and the Triples, an important observation was made concerning the “X” word, consisting of its existence in Pairs also as a word to be extracted. An opinion word can have either only one target, or several targets joined by a conjunction, but if the target is also an “X” word for the same opinion word, the rules extract several targets for a single opinion word without conjunction between them, which is means that there exists a false result from the extraction process.

From the results can be seen that the extraction done in Pairs is correct in these cases, and that the results from Triples need to be filtered out in these cases, leading to the creation of additional, which led to significant score improvements, as can be seen in the Testing and Validation chapter.

Considering a Triple can have the following forms:

* Opinion word – “X” word – Feature
* Opinion word – “X” word – Opinion word
* Feature – “X” word – Opinion word

We do not extract the Triple if the POS tags in the triple have any of the following forms:

* NN-NN-JJ
* JJ-NN-NN
* JJ-JJ-NN
* NN-JJ-JJ

Where NN is the POS tag of features, JJ is the POS tag of opinion words.

All four cases correspond to at least one of the rules in Table 4.1, but the problem lies in the fact that they correspond to more than one rule which are intended to be applied exclusively, considering the concept that we have a mapping of one-to-one between an opinion word and a feature. For example in the case of JJ-JJ-NN, Rule 1 extracts the Opinion-Feature pair JJ-NN and the JJ-JJ is extracted using Rule 3, so extracting also JJ-JJ-NN does not give us any new word, but at best duplicates and at worst an incorrect extraction, due to the lack of check for a conjunction between JJ-JJ.

The case of NN-NN-JJ and JJ-NN-NN is mostly related to compound nouns, like *g3 phone*, where clearly we have only one product, so extracting only target is the correct decision, which means we should either extract *g3* or *phone*. Using *g3*  as target over *phone* gives us a more specific feature which can be more reliable, but considering the way new features are extracted, using specific features yields worse results that using general ones, because their frequency is much lower and the extraction algorithm is not extract them and thus not use them for further extractions also. The specific target words would be extracted by and JJ-NN-NN and the general one are extracted by using JJ-NN, so the filter for these words not used as targets was implemented.

As can be seen from the details description of the implementation done in this subchapter, the logic became quite complex when creating a general implementation for the extraction of both types of words, opinion word and target, using both types of rules (with or without “X” words), a sacrifice made for the sake of creating a common extraction process for all rules, and the scalability regarding the rules number.

### Source-Target pair merger

This module has the simple purpose of merging all the results provided by the Opinion Word Extractor and the Target Extractor, filtering duplicates and providing a single collection of Source-Target pairs, containing all the extracted opinion words and targets.

The implementation is done by using the power of the Sets provided in Java, which allow for addition of an entire collection of data while automatically filtering the duplication in the addition process, making it very easy to use and understand.

### POS relations aggregator

The number of POS tags provided by Stanford greatly exceeds the generic ones, and using them when implementing the rules would require enormous amounts of checks and validations, the power of the Stanford Parser, which extracts different types of adjectives and verbs, should not remain unused. The solution we came up with was the mapping of the specific POS tags into generic one.

The POS Relations Aggregator module has the purpose of providing this mapping for all modules to use and is used extensively in the Extraction Service.

The power of enumerations is used to achieve the mapping, as every Stanford POS tag is part of a type of enumeration, each of which represents a generic POS tag. For example the JJ enumeration contains the JJ, JJR and JJRS values, each of which is a specific Stanford POS tag. This allows us to check for all these values using only one call to the JJ enumeration. Of course, behind the scenes, the module checks the values in the JJ enumerations for a value provided using reflection. The advantages provided by this mapping are incredible, as we can easily add or remove specific POS tags, like JJR in the enumeration without changing the implementation in the slightest, a much desired feature during the experimentation phases, and a feature which provides tremendous extensibility and flexibility.

The usage of specific POS tags allows us to increase the precision of the system, by ignoring certain POS tags which cannot reflect any opinion, like an adjective depicting the color of an object, which has no value for the user as it is objective.

After extensive experiments, the following POS tags provided the best results for our purpose, which we will list for each generic POS tag in the rules table:

For NN:

* NN (noun, common, singular or mass)
* NNP (noun, proper, singular)
* NNPS (noun, proper, plural)
* NNS (noun, common, plural)

For PP:

* PP (pronoun)
* PRP (pronoun, personal)

PRP$ ( pronoun, possessive) was excluded from the implementation as the pronouns of this type cannot constitute valid targets for opinion words, and during experiments, the decrease in precision outweighed the increase in recall.

For JJ:

* JJ (adjective or numeral, ordinal)
* JJR (adjective, comparative)
* JJS (adjective, superlative)

For RB:

* RBS (adverb, superlative)
* RBR (adverb, comparative)

RB (adverb) was excluded from the implementation as the adverbs of this type cannot be opinion words, as they have no polarity, being generally objective. During experiments, the decrease in precision outweighed the increase in recall.

For MR-Rel:

* Acomp (adjectival complement)
* Amod (adjectival modifier)
* Rcmod (relative clause modifier)
* Npadvmod (noun phrase adverbial modifier)
* Acomp (noun compound modifier)
* Advcl (adverbial clause modifier)
* Nsubj (nominal subject)
* Csubj (clausal subject)
* Nsubjpass (passive nominal subject)
* Csubjpass (passive clausal subject)

The list of specific POS tags is essential for understanding that not any adjectives or adverbs can be considered opinion words, and not any substantives or pronouns can be considered features, because when removing the polarity and the purpose of the extraction, and examining the results, one might think that extracting opinion words means extracting adjectives, which is terribly wrong as few adjectives provide insight into the opinion of the text writer.

The list of the all the POS tags provided by Stanford, as well as additional information regarding the Stanford typed dependencies information can be found in [23].

## The Evaluation System

The system implemented has to be tested, and considering that in a collection of reviews summing up to about 500 sentences usually more than 200 opinion words are found, a better way than manually cross referencing the annotations with the system output needs to be used, an automatic evaluation process. Basically what needs to be done is provide a way for the system to extract the annotated opinion words and features in a format similar to the one used for the extracted opinion words and features. The Evaluation System is designed as an original solution to this problem.

Considering that the extraction of opinion words and targets can also be done independently, two inner components called Opinion Word Extraction Evaluator and Feature Extraction Evaluator are designed to for each evaluation process. There is no need to evaluate the elements of the pairs extracted together and it makes much more sense to do it separately in order to ensure that the evaluation module can be reused for other systems and make the module independent on the system output format, providing additional flexibility.

### Opinion mining system in evaluation mode

When using the Evaluation System, there is additional input required, consisting on the evaluation models which need to be provided for both opinion word and target extraction evaluation.

The architecture in Figure 5.8 represents the integration of the additional components of the systems used for evaluating the FOPI module of the OMS system.

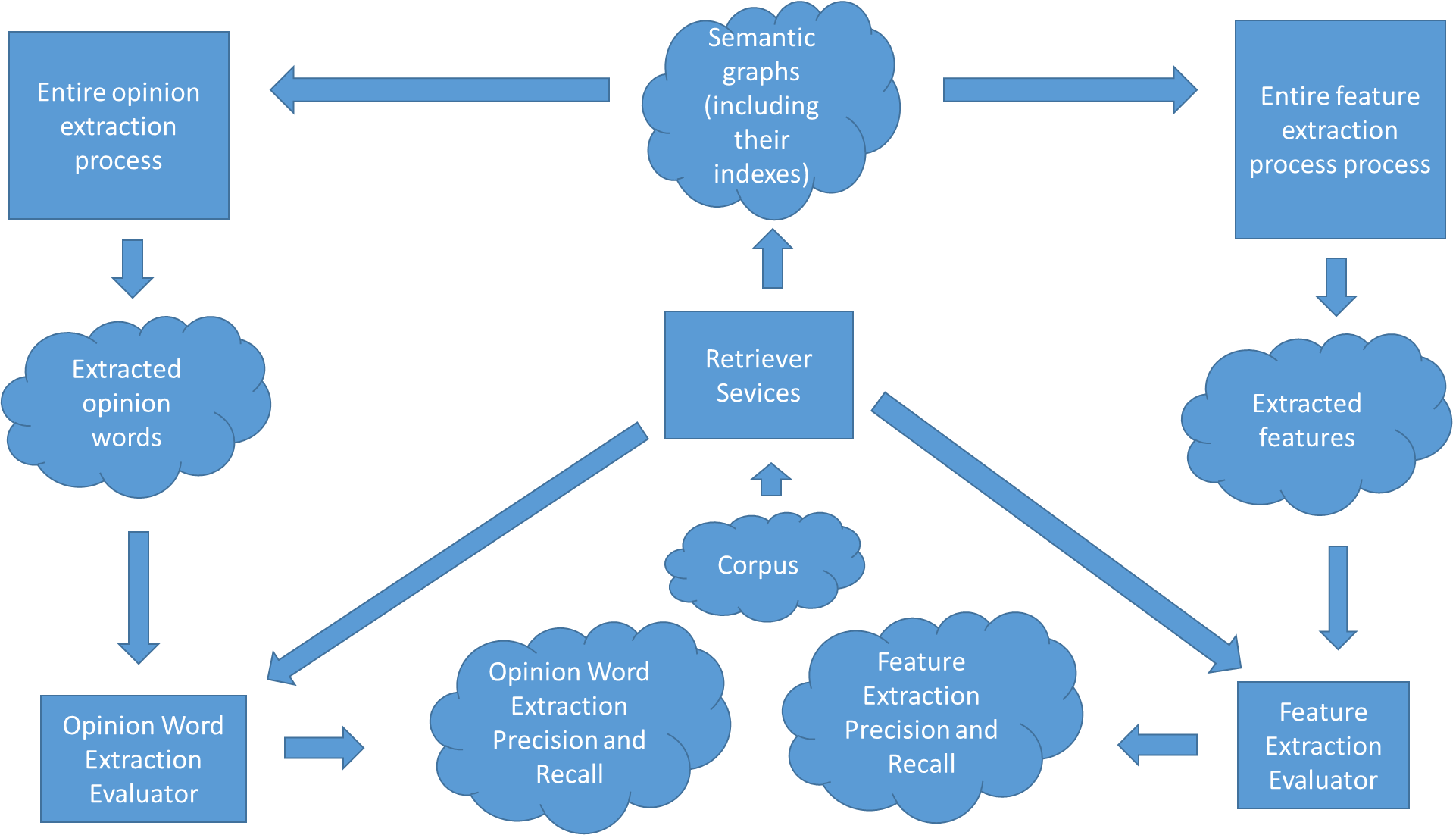


Figure 5.8 The OMS in evaluation mode

### Opinion word and Feature Extraction evaluators

The evaluator modules are responsible for the extraction of evaluation models and the evaluation of the results obtained by the system. Due to their similar implementation, the only change being the type of evaluation performed (the opinion word extraction evaluation and the feature extraction evaluation), they will be presented together..

In order to compare an annotated word with an extracted word, there are three properties which have to be known for both:

* The lemmatized word
* The POS tag of the word
* The sentence from where the word was extracted/annotated

The first two properties can be obtained by using the Stanford CoreNLP for both the annotated and the extracted word. For extracting annotated words, we must provide a way for the system to identify them, which is provided by the usage of special characters at the beginning of each annotated word. The special characters need to be unique to the system, be removable and easy to set by annotators. For each type of word (opinion word or feature) there is series of three characters added at the start of the word, in the case of opinion words *### is* used, and for features, *%%%* was considered to be a good choice, due to the improbability of the appearance of these patterns in any other scenarios.

The patterns are extracted by using a regular expression, which searches for the special patterns in the semantic graphs obtained using CoreNLP, ensuring the first two properties are contained in the evaluation model.

In order to ensure that we have a “clean” text as input, any additional annotation needs to be removed when the preprocessing on the text is done for the extraction process, as the dependencies between words change with any new text, which leads to the necessity of processing the text twice with CoreNLP, once in order to obtain the annotated word with the required properties and the second time for the extraction process. The annotation does not influence the processing when extracting the annotated word from the text, because we are only looking for words with annotations, without taking into consideration the dependencies between words which would be changed due to the annotations.

For the third property, the sentence of the words, there were two possible implementations considered. The first one consists on storing the semantic graph for each word extracted, which requires a lot of memory space, increasing the memory demand exponentially for each extracted word, a factor which lead to thinking of a more space efficient method of comparing the sentences from there the words are extracted. Keeping in mind the memory constraint and the information needed, the second approach was discovered and implemented, which consists on using the position of the sentence rather than the sentence itself to denote the provenience of each word. This approach is possible due to the format of the input of the FOPI module as an ordered collection of semantic graphs, each which is actually a sentence, a collection which can be used also for the construction of evaluation models. By using exactly the same preprocessing for obtaining semantic graphs for evaluation models, we can safely rely on the order of the semantic graphs, and thus of the sentences in the text for obtaining the relative place where a word is extracted. This means that the index representing a sentence must be stored for each extracted word from the actual extraction process, and thus it must be propagated from the Retriever Services to the Extraction Service and finally arrive with the word in evaluators modules.

The extraction algorithm is not complex, but rather intuitive as can be seen in the pseudo code presented for the Opinion Word Extraction Evaluation module:

Input: Actual Opinion Words {A}, Found Opinion Words {F}

Output: Precision {P}, Recall {R}

Function:

1. {TP } = 0, {FP} = 0, {FN} = 0
2. For each opinion word {O} in {F}:
3. if ({A} contains {O})
4. {TP} = {TP} + 1
5. else {FP} = {FP} + 1
6. endif
7. endfor
8. For each opinion word {O} in {A}:
9. if ({F} does not contain {O})
10. {FN} = {FN} + 1
11. endif
12. endfor
13. Set {P} = {TP} / ({TP} + {FP})
14. Set {R} = {TP} / ({TP} + {FN})

The input consists of two collections, one containing extracted opinion words and one annotated opinion words. Each of the opinion words extracted needs to be cross referenced in the list of annotated opinion words, in order to be marked as a true positive result(TP). In the case when an extracted opinion word is not found in the list of annotated opinion words, then we have the case of a false positive, meaning the extracted word is not actually an opinion word. For computing the false negative results, meaning the opinion words which were not extracted by the system, we need to change the direction of the cross referencing, meaning we find which opinion words annotated are not found in the extracted opinion words list. Finally, by summing up the three values (false positive, false negative and true positive), the precision and recall of the system can be computed, thus providing a standard way for evaluating the system.

For the Feature Extraction Evaluation module, the algorithm is the same, but the contents of the list change from opinion words to features.

# Testing and Validation

In this section the results of the evaluation of the solution implemented and the improvements added are presented, by presenting the impact of each of the parameter of the system, one at a time and explaining how it influences the system. The evaluation of the opinion words extraction and target extraction process, meaning the only the results for the Feature-Opinion Pair Identification module are presented here due to the large number of parameters and results which can be generated by using different combinations, additional results being documented in the Experiments Annex and in the [2], where experiments containing the Retriever Service evaluation are predominantly presented and in [3], where the results of the Polarity Aggregation module can be found.

The following notations are used in the graphs and their explanations extensively:

* T – Target (feature)
* OW – Opinion Word
* 6789 seed words – the largest set of reliable seed words available
* 2 Seeds – two seed words (good and bad, which are representative for negative and positive classes)

The 6789 seed words are taken from [14], and represent a set of 2006 positive seed words and 4783 negative seed words. The list was compiled over many years, starting in 2004 and can be considered to contain all relevant opinion words which could appear in a text.

## Data set

The data set used is taken from [14]; a set used in numerous other opinion and target extraction methods since 2004, providing a reliable source for evaluating our solution.

It contains 4 text files, in each of which a large number of reviews are contained (an average of 700 sentences), having the same target but different authors. There is no additional delimitation besides the different files, representing reviews for different products. The products are 2 cameras, and one DVD player and one phone.

Due to the lack of existing opinion word annotations, the annotations used for testing are constructed manually by 3 different persons, which went thought the texts twice, in order to ensure reliable source of evaluation. Any adjective or adverbs which may indicate an opinion are considered opinion words and any pronoun or noun for which an opinion can be inferred in a sentence are considered targets.

The data set contains unstructured text and taken from amazon, meaning it has grammatical errors as well as badly structured reviews, objective reviews or text which cannot be considered a relevant review as it contains irrelevant information. By using this type of data, the testing done proves the solutions viability for real word usage, where noise is common.

## Experiments and results

### Initial Results vs. Final Results

A comparison between the initial results, without any enchantments, and the final results, containing all the enchantments is presented first, in order to better see the starting point, end point and the enchantments value. The comparison is done by using the set of 6789 seed words.

The precision and recall of the opinion words extraction is presented in Figure 6.1, where for each test data used we can see a small decrease in recall, but a huge increase in precision.

Figure 6.1 Initial vs final results of opinion word extraction

Considering the initial results, it is quite obvious that focusing on increasing the precision was mandatory. The recall generally had good values already, so more accurate by reducing the garbage produced with the cost of possibly omitting some good results is acceptable, as long as the end justifies the means. Getting a precision over 50% is mandatory, because otherwise the system has no real value, as it has more chances of giving false opinion than good ones. As can be seen in Figure 6.1, the effect of the enchantments is similar over all test data, meaning their values is not restricted to some specific case, but considering that all the results are overall improved, without exception, their usage is justified.

A similar result to the opinion mining extraction can be seen in Figure 6.2, where the precision and recall of the target extraction is presented.

Figure 6.2 Initial vs Final results of the target extraction

In the case of target extraction, the impact of the improvements is not as accentuated as for the opinion extraction, in the case of test data for Nokia; the enchantments done are actually detrimental. The reason for the behavior of Nokia and the lack of change in precision which is almost inexistent during all the experiments made, lies in the test data itself, where noise is proportionally mixed with the good results, meaning by increasing or decreasing the restriction of the results, we get about the same amount of good data.

### Seed words number influence

The reason we can safely claim the approach is unsupervised lies in the user set data needed to achieve a good performance. The seed words are used to bootstrap the extraction process, meaning a large number of seed words, or a set of seed words which is specific to a domain would move the solution to the area of semi-supervised approaches. The results of the experiments conducted with the seed words number justify that only one or two generally used opinion words are needed in order to achieve a good accuracy.

In Figure 6.3 and 6.4, the effect of choosing an enormous amount of seed words versus the smallest set possible, two seed words, each of which represent a negative or positive polarity can be seen.

Figure 6.2 Seed words influence on opinion word extraction

Figure 6.3 Seed words influence on target extraction

This behavior is explained by the following two facts: the number of reviews is sufficiently large; there is a high probability that the two – very common – words are used at least once to describe a product or one of its features. After at least one target is extracted, the iterative algorithm finds all the opinion words associated with it. The number of opinion words extracted in this case is close to the one found by using a very large set of seed words. Following this reasoning, we can safely state that this approach is unsupervised.

Despite the low difference in the results induced by the number of seed words, there is a large difference in the extraction times. As can be seen in Figure 6.5, where we have the execution times for the Feature-opinion pair identification component, the number of seed words dramatically increase the extraction time. This is caused by the excessive number checks made by each rule on each possible opinion word. As an example, for extracting 400 opinion words, using 2 seed words we have a maximum of 402 comparisons, but by using 6785, we have over 7000.

Figure 6.4 Seed words number influence on the extraction time

From Figure 14 also the performance of the Feature-Opinion Pair Identification module can be seen very clearly, by considering the usage of 2 seed words, the extraction time is just over 1000 miliseconds, which is even faster than loading most web pages and can be considered almost real-time. Of course the extraction is directly proportional to the size of the text, but considering that it takes 1 second to process about 500 sentences, we can safely say that it would perform reasonably well for any data size.

### Polarity Threshold influence

The polarity threshold is used for determining the adjectives and adverbs which are actual opinion words, by searching them in SentiWordNet and comparing their polarity with a predefined minimum value. This means that increasing the minimum required values will obviously translate in an increase of precision and possible decrease of recall. The problem then become finding the ideal values, which offers the best precision-recall ratio.

The most relevant values are used for displaying the evolution of the results based on the polarity threshold influence, for opinion word extraction in Figure 6.6.

Figure 6.5 Polarity theshold influence on opinion word extraction

As can be seen Figure 15, up until a threshold value of 0.07, the precision increase outweighs the drop of the recall, but the best results are observed at a threshold value of 0.01. This is due to the fact that with an increase in the threshold value there will be an increase in the number of opinion word omissions, and we actually just need to check if a word is an opinion word, meaning just finding it in SentiWordNet is enough.

The threshold can be used for tuning the system for the retrieval of only the most reliable opinion words, or for the extraction of as many as possible, offering a lot of flexibility for the user.

### The X-Filter influence

As mentioned during the implementation description, additional logic was implemented for the rules containing ”X words”, but we need to justify and show the results of using it. In Figures 6.7 and 6.8, the effect of using it can be seen on both the opinion word and target extraction process.

Figure 6.7 X-filter inflience on opinion word extraction

In the case of opinion word extraction, the difference is not dramatic, usually filtering process efect can be seen to produce a small increase of precision and small decrease in recall, with small variation depending on the dataset. There is no real improvement in terms of precision and recall ration, but keeping in mind the intial results, where the precision must be imroved, this filter offers a step in that direction, justifying its value in the implementation.

Figure 6.8 X-filter influence on target extraction

For the target extraction process, the filter offers a deffinite improvement, as the precision increases drematically, while the decrease in recall is much more insignificant. This combined with the need for an precision increase in both extraction processes leads

### POS tags influence

The use of the types POS tags provided by Stanford, rather than just the NN, JJ, RB and PR part of speech tags helps with a better extraction process, as can be seen in Figures 6.9 and 6.10.

Figure 6.9 POS tags influence on opinion word extraction

Considering the imense precision gain with just a small cost of recall, it is obvious that searching for specific adjectives and adverbs helps when extracting opinion words. The reason behind this change is due to the common usage of comparative and superlative adjectives and adverbs for expressing opinions, like *larger, better, best*. So by considering only these types, we can have a better opinion mining system.

Figure 6.10 POS tags influence on target extraction

In the case of target extraction, the POS tags extracted by Stanford CoreNLP do not offer suffient separation between features and non-features, as in general, almost any noun can be a feature. The inprovement in recall which is almost directly proportional to the decrease in recall is due to the filtering of pronouns, as possesive pronouns are not valid features.

### Domain Independence Evaluation

Domain independence was first tested between product reviews from different sources (cameras, mp3 players etc.) and then tested between product reviews taken in general and movie reviews. In Figure 6.11, the first column from each of the four-set clusters represents the results from product reviews when 6785 seed words were used. The second one corresponds to the same amount of seed words being used for the movie reviews. The equivalent columns which use 2 seed words are the last two of each cluster. Note that the same configuration was used for both the product reviews and the movie reviews, i.e. the *final* configuration presented in the previous section.

Figure 6.11 Cross domain comparison of opinion words and targets

There are generally two types of subjective texts, one which contains only text on topic, like product reviews, and another which is more descriptive in nature, like movie reviews which also describe the plot. In the description, opinions unrelated to the actual target of the subjective text can be conveyed, which affect the extraction process. This behavior can be seen in Figure 6.11 on the extraction of movie reviews using 6000+ seed words. The usage of only two seed words prevents this unwanted behavior, as the propagation is generally limited to related targets.

The experiments done for evaluating and improving the system are quire numerous, but only the ones most relative to the work presented in this thesys are contained in this chapter, additional results being contained in the works done by my colleagues in [2] and [3].

### Polarity Assignment Evaluation

In Figure 6.12, the results of the polarity assignment are presented. They are presented using different values for the score theshold, which determines the tolerance for polarity assigment errors. The polarity threshold was kept to 0.4 because this was the value for which one data set conveyed 100% precision, target frequency was set to 2 and we used the maximum number of seed words. The most optimal value was found to be 0.4, meaning a classification of neutral, weak and strong opinion words can be made by the proposes solution, leading to a suffient classification, because determining the polarity of opinion words is also a subjective task.

Figure 6.12 Polarity Assignment scores

# User’s manual

In this chapter, a step-by-step guide to installing both the application and all of its prerequisites will be presented. It starts off with a short description of how the system was thought of followed by a guide to installing it and its requirements and a main success flow.

This chapter focuses on presenting the steps needed to install the application and the tools it needs. First a short description of the installation of the solution is done, followed by its usage details.

## Generalities

The system was implemented completely in Java, due to the ease of integration with existing opinion mining tools and it’s free to use license. The versioning system chosen is GIT, for the ease of use and reliability.

## Prerequisites

This subchapter will present all the things which have to be installed or simply downloaded in order to run the system which include Java JDK and Eclipse IDE.

### **Java JDK**

Even though most computers have the Java Runtime Environment already installed, it’s important to download and install the Java Development Kit (JDK) instead. This is required for compiling and successfully running the application from the Eclipse IDE.

The download link can be found at <http://www.oracle.com/technetwork>/ java/javase/downloads/jdk7-downloads-1880260.html or in the Appendix section. You are there required to choose the corresponding operating system, preferably a version of Windows, where the solution was thoroughly tested. After successfully downloading and installing the Java JDK, we can move on to downloading the Eclipse IDE.

### **Eclipse IDE**

The IDE used for writing, compiling and running the code/application is Eclipse. It can be fully integrated with Maven and GIT and it offers great intellisense for java and is overall viewed as being a very power IDE. It can be downloaded from at http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/heliossr1. There are plenty of packages to choose from, each one being slightly different however the differences are irrelevant to our needs so any of them can be chosen.

## Additional requirements

The additional settings which need to be made for a successful installation will be presented in this subchapter, including the GIT cloning process, project import and IDE settings.

### **GIT and GIT Cloning**

For the whole development process the code versioning system used was GIT. The repository is public thus it can be cloned without any additional permissions. Of course, for pushing or branching of any sort which is not to be done locally, permissions from our side have to be granted. Please contact us for any such permissions.

First of all, one has to download and install GIT. The link can be found at http://git-scm.com/downloads. After this, even though the GIT bash can be used to pull the repository, a more elegant and intuitive way will be presented by using Tortoise GIT. Tortoise GIT is an extension to GIT which offers a more “*click and do*” user friendly approach to operating the GIT commands. It’s an alternative to writing bash commands which can sometimes get tricky. The download link can be found at https://code.google.com/p/tortoisegit/wiki/Download.

After installing these two in this exact succession, when right-clicking anywhere in the file system, one should get additional menu items as seen in Figure 7.1.



Figure 7.1Menu Items

Clicking the first item, *Git Clone…* will result in a window very similar to the one in Figure 7.2. Simply copy the following text in the *URL* field and click ok.

A slight warning, a folder will be created inside the folder in which the *right click* was made.

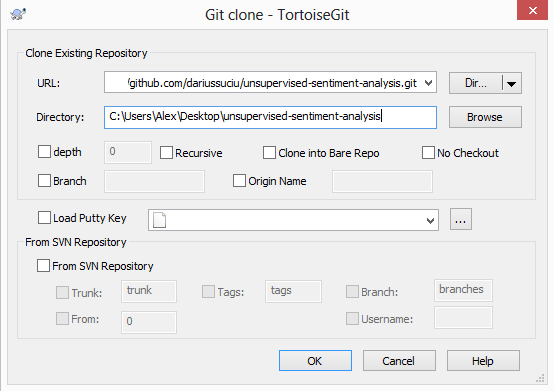


Figure 7.1 Git Clone Menu

### **Maven in Eclipse**

Not all versions of Eclipse come with Maven preinstalled therefore a short tutorial might come in handy. After starting Eclipse and arriving at the (most probably) new and empty Workspace, go to *Help* 🡪 *Install new software…* in the above menu-bar. In the *Work* *with* field from the dialog which appears, copy paste the following: <http://download.eclipse.org/technology/m2e/releases>. After checking the newly appeared checkbox, the dialog should look like in Figure 7.3.

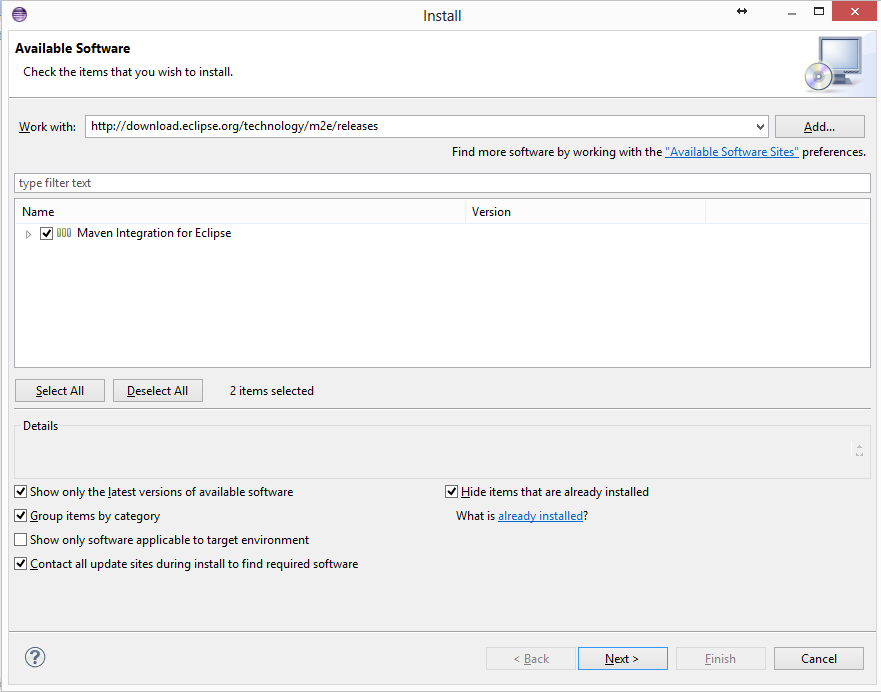


Figure 7.2 Install Menu

Clicking Next, Next, accepting all the terms and conditions etc. will eventually install the Maven plugin for Eclipse.

### **Importing the project**

Simple yet necessary operation. Right click anywhere in the left package-explorer view in Eclipse and click on *Import…*. Select the Maven dropdown list, existing Maven Projects and in the *Root Directory* field browse to the folder where the GIT Cloning was done. The dialog should appear as in Figure 7.4.

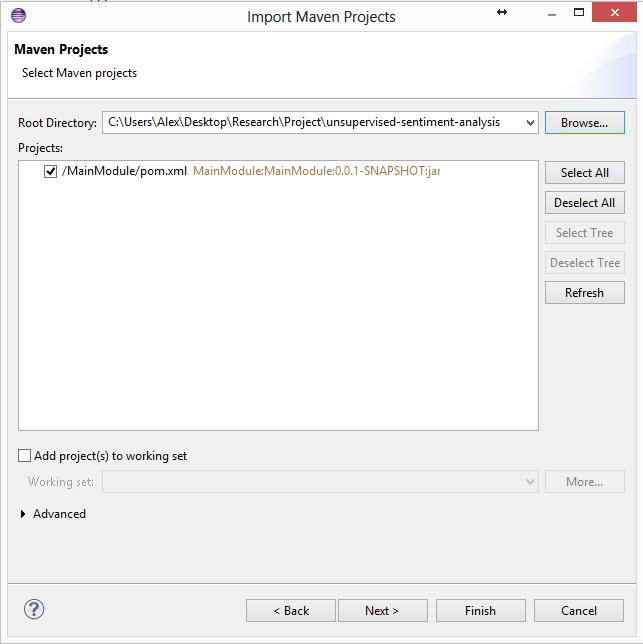


Figure 7.3 Maven Project Import Menu

After clicking *Finish* and waiting for all the processes to finish, the application should be imported fully into the workspace, including all additional maven dependencies which should automatically be downloaded.

## Running and testing the project

This subchapter will cover all the different configurations which have to be initially changed before attempting to run the project and a quick guide on how to run it.

### **Configuration management and initial settings**

The first thing that has to be done is to copy the configuration model which is present in one of the folders which was pulled from GIT. The folder name is *configModel* and the *config.xml* file inside should be copied to the *MainModule* folder. There are only a few mandatory modifications which have to be done in the *config.xml* file.

Please modify the *<inputDirectory>* and put the path to the folder …/ProjectRoot/TestReviews/Annotated.

Also modify the <SWNPath> to point to the ProjectRoot/SWN/SentiWordNet.txt file.

The file paths for the <negativeSeedWordsFile> and <positiveSeedWordsFile> must also be changed to the ones corresponding to the files in the ProjectRoot/SeedWords/\*-words.txt.

The output folders can be chosen at will. The rest of the configuration parameters are not mandatorily changed however are thoroughly explained in chapter **Error! Reference source not found.**.

### **Running the project**

After all the previous steps were finally done, the last step would be to run the application in all its glory. In the IDE package explorer, simply right click the project and select *Run As 🡪 Java Application.* The project will then start. As a small warning, the initial runtime is of about 1-2 hours depending on the processor power.

# Conclusions

### **Summary and contributions**

The domain of opinion mining is rapidly evolving, with the problem of extracting opinion being handled at various granularity levels, and using fundamentally different methods. One of the methods relies on a supervised approach, which after having reached its apex in terms of possible accuracy, lead to the increased interest on unsupervised approaches, which offer the much needed domain independent solutions, an interest which led to the creation of the solution proposed.

The solution is designed in this thesis handles the increasing demand in extracting the important opinions from the enormous datasets feely available, offering increased performance, accuracy and most importantly domain independence. It breaks down the domain barriers that exist in most similar approaches, by using the grammar and semantic rules that are part of written text and a smart approach that produces most of its necessary data itself. The solution is designed to adapt to the various user demands, which can focus on accuracy or coverage, by using a set of parameters, which by tuning can dramatically affect the behavior, tuning done by varying the number of seed words, the required minimum polarity for opinion words or frequency of accepted targets. Considering the vast domain of opinion mining and its multitude of applications, the solution is designed for both independent usage in extracting opinion, but at the same time a great deal of focus was dedicated for its modularity, reusability and flexibility, in order to achieve the easiest integration possible for additional modules designed for handling additional tasks, like ranking targets or complex negation handling as well for the integration of the entire solution of any of its components independently in a larger system.

The contributions added are focused on achieving domain independence by enhancing solutions proposed to solve semi-supervised opinion mining, eliminating the domain constraints, integrating newer and better tools and improving the noise removal techniques, with the purpose of achieving a better accuracy. The noise removal techniques focus on the extraction process of both opinion words and targets, each of which directly affect the other. For opinion words, improvements are done on filtering adjectives, by using their polarity, the usage of additional or enhanced rules for the extraction process, by using the more specific part of speech tags, which have become available and the consideration of opinion expressing adverbs. For targets, the usage of their frequency, usage of pronouns and enhanced specific part of speech tags for the extraction process. In the end, with the addition of these features on top of a relatively new and smart unsupervised approach, a viable domain independent, unsupervised solution is created, that can be freely configured for each user needs.

### **Experiments and results conclusions**

The large number of experiments conducted helped shaped the final, enhanced form of the solution, but at the same time offered the following very important insight: The numbers of seed words do not affect the performance of the system

This gives us the possibility of using the system in any domain, without the need for any user input, as only data needed consists of the two opinion words, *good* and *bad*, opinion words that are consistent with any domain, while at the same time the usage of smaller helping data improving the system performance immensely, allowing it to perform in almost real time.

The results show relatively high system accuracy, represented by a precision of over 55% and a recall of exceeding 65%, results which do not exceed the one offered by the existing supervised approaches, but are sufficient for a reliable opinion extraction, the value added by the domain independence and system performance compensating for the difference. The usefulness of the parameters created is also visible in the results, as a setting for achieving a precision of 90% can be easily made, unfortunately with a big cost in recall, and vice versa, but which can satisfy the various user demands.

Considering the initial and final results presented, starting from an initial useless tool, in which the garbage extracted exceeds the opinions in the output, a reliable solution that can be easily be used for finding the relevant opinions was derived, which uses the knowledge and tools currently available.

### **Future Development**

The system is designed to be as extensible as modular as possible; all for the purpose of the addition of future work, as the opinion mining domain have become huge in terms of possibilities and direction of research. The work done in this thesis, only when combined with the work done by my other two colleagues manages to take a stab at a shallow implementation of extracting opinions in an unsupervised ways. The issues are not all handled, as can be seen in the results, where the room for improvement has been seen.

The improvement of the system can be approached in numerous ways, one being in the continuation of the direction followed in the improvement of the solution, doing experiments and seeing patterns in the data which can be used to get a better accuracy. This an approach used extensively during the evaluation phase, leading to the discovery of the filtering processes added to the system and which does not yet shown exhaustion in terms of possibilities. Another very important approach can be the integration of modules which handle the issues that are too big to be properly treated in the development of the solution, considering the time constraint, issues which include the handling of negation, a very important process needed by the system, but which does not consist on solely finding the “not” words, but handles all the possible forms of negation that exist in written text. A similar issue that when handled would lead to better opinion mining is understanding sarcasm, an issue whose approach is tremendously difficult, but somewhat necessary when dealing with user created data. Many more such issues still remain unsolved, because as mentioned, this domain is vast and will become even more so in the future. By considering the results and their applicability, another direction become visible, which consists on expanding the scope of the solution, as the pairs of target-opinion words can be used to detect sentence or document level sentiments, possibly achieving better results than existing solutions, due to the higher degree of granularity actually used. The analysis of the results be used in predicting behavior in stock market for example or political elections, meaning the creation of a bigger system, in which opinion mining is an important component can be also considered as possible future work.

In conclusion, the possibilities the solution offers are endless, but each of them needing additional time and interest as payment for achieving.

# Bibliography

|  |  |
| --- | --- |
| [1] | W. J. Frawley, G. Piatetsky-Shapiro and C. J. Matheus, "Knowledge Discovery in Databases: An Overview," *AI Magazine,* vol. 13, no. 3, pp. 56-57, 1992. |
| [2] | A. C. Cosma, "Preprocessing and multi-dimensionality aspects in a unsupervised extraction system," 2014. |
| [3] | V. V. Itu, "Cross-domain polarity assignment by minimal interaction with external resources," 2014. |
| [4] | B. Liu, "Sentiment Analysis and Subjectivity," in *Natural Language Processing*, 2010, pp. 627-666. |
| [5] | J. Yi, T. Yi and Nasukawa, "Sentiment analysis: Capturing favorability using natural language processing," in *In Proceedings of the 2nd international conference on Knowledge capture*, 2003. |
| [6] | B. Pang, L. Lee and S. Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques," in *In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, 2001. |
| [7] | L. Zhuang, F. Jing and X.-Y. Zhu, "Movie Review Mining and Summarization," in *Proceedings of the 15th ACM international conference on Information and knowledge management*, Virginia, 2006. |
| [8] | G. A. Miller, R. Beckwith, C. D. Fellbaum, D. Gross and K. Miller, "Wordnet: An online lexical database," in *International Journal of Lexicography*, 1990. |
| [9] | V. Hatzivassiloglou and K. McKeown, "Predicting the semantic orientation of adjectives," in *In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, 1997. |
| [10] | P. D. Turney, "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews," in *In Proceedings of the 40th annual meeting on association for computational linguistics*, 2002. |
| [11] | L. Zhang, B. Liu, S. H. Lim and E. O`Brien-Strain, "Extracting and Ranking Product Features in Opinion Documents," in *International Conference on Computational Linquistics*, 2010. |
| [12] | B. Liu, G. Qiu, J. Bu and C. Chen, "Opinion word expansion and targets extraction though double propagation," in *Computational linguistics 37*, 2010. |
| [13] | S. S. Htay and K. T. Lynn, "Extracting product features and opinion words using pattern knowledge in customer reviews," in *The Scientific World Journal*, 2013. |
| [14] | B. Liu and M. Hu, " In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining," in *International Conference on Knowledge Discovery and Data Mining*, 2004. |
| [15] | A.-M. Popescu and O. Etzioni, "Extracting product features and opinions from reviews," in *Conference on Empirical Methods in Natural Language Processing*, 2005. |
| [16] | D. Lin, "Dependency-based evaluation of Minipar," in *Treebanks*, Springer, 2003. |
| [17] | "Stanford NLP Group," [Online]. Available: http://nlp.stanford.edu/software/. |
| [18] | K. Toutanova, D. Klein, C. D. Manning and Y. Singer, "Feature-Rich Part-Of-Speech Tagging with a Cyclic Dependency Network," in *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 2003. |
| [19] | M.-C. de Marneffe, B. MacCartney and C. D. Manning, "Generating Typed Dependency Parses from Phrase Structure Parses," in *LREC*, 2006. |
| [20] | S. Baccianella, A. Esuli and F. Sebastiani, "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining," in *Seventh conference on Iternational Language Resources and Evaluation*, 2010. |
| [21] | N. Jindal and B. Liu, "Mining Comparative Sentences and Relations," in *National Conference on Artificial Inteligence*, Boston, 2006. |
| [22] | S.-M. Kim and E. Hovy, "Determining the sentiment of opinions," in *In Proceedings of the 20th international conference on Computational Linguistics*, Stroudsburg, 2004. |
| [23] | M.-C. de Marneffe and C. D. Manning, "Stanford typed dependencies manual," 2008. |