TRABAJO FIN DE GRADO

Título:	Desarrollo de un Analizador de Sentimientos basado en Aspectos para Redes Sociales y Aplicación a Reseñas de Productos.
Título (inglés):	Development of an Aspect-based Sentiment Analyzer for the Social Web and Application to Product Reviews
Autor:	Manuel García-Amado Sancho
Tutor:	Óscar Araque Iborra
Ponente:	Carlos Ángel Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	Mercedes Garijo Ayestarán
Vocal:	Álvaro Carrera Barroso
Secretario:	Juan Fernando Sánchez Rada
Suplente:	Tomás Robles Valladares

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DEVELOPMENT OF AN ASPECT-BASED SENTIMENT ANALYZER FOR THE SOCIAL WEB AND APPLICATION TO PRODUCT REVIEWS

Manuel García-Amado Sancho

Julio de 2016

Resumen

Esta memoria es el resultado de un proyecto cuyo objetivo ha sido desarrollar y desplegar un analizador de sentimientos basado en aspectos aplicado a reseñas de restaurantes mediante herramientas de Procesado de Lenguaje Natural (NLP), varias de ellas desarrolladas en el Grupo de Sistemas Inteligentes.

Para hacer esto se ha desarrollado un sistema de extraccion, contextualización y clasificación de aspectos. Concretamente, cada aspecto comentado por un usuario en su reseña de un restaurante.

Para la prueba y evaluación del sistema hemos usado un conjunto de datos compuesto por reseñas de restaurantes que nos ha servido tanto para esto, como para ir implementando cada módulo en nuestro analizador. Primero, se realiza la extracción de los aspectos de cada reseña. Con el siguiente módulo, se busca el contexto de cada uno de estos aspectos. Tras hacer esto, el objetivo de los siguientes módulos es clasificar de dos formas estos aspectos. Primero, se clasifican en base a seis posibles temas sobre restaurantes, y finalmente respecto a su polaridad, esto es, si se valora el aspecto positiva o negativamente.

Además, hemos validado un prototipo con el conjunto de datos y desarrollado una interfaz gráfica basada en Web Components y D3.js para la visualización de los resultados del analizador.

Como resultado, este proyecto nos permitirá aplicar la técnica del análisis de sentimientos basada en aspectos a reseñas de restaurantes, lo que puede conllevar un estudio interesante del mercado de los restaurantes y en un futuro aplicarlo a otras temáticas.

Palabras clave: Aspectos, Análisis de Sentimientos, Reseñas de restaurantes, Datos enlazados, Procesado de Lenguaje Natural, Python

Abstract

This thesis is the result of a project whose objective has been to develop and deploy an aspect-based sentiment analyzer applied to restaurant reviews based on Natural Language Processing (NLP) techniques, most of them developed at Intelligent Systems Group.

To do so, a system that extracts, contextualizes and classifies aspects was developed. Specifically, each aspect commented by the user in his restaurant review.

For testing and evaluating the system we have used a dataset composed by restaurant reviews that allows implementing each module in our analyzer. First, the system realizes the extraction of aspects from each review. With the next module, the context of each aspect is found. Once we have done this, the objective of next modules is to classify in two ways these aspects. First, regarding six possible topics and finally depending on its polarity, that is, if the aspect is valuated positive or negatively

Besides, a prototype has been validated with the dataset and a visualization system has been developed using Web Components and D3.js to show analyzer results.

As a result, this project allows us to apply aspect-based sentiment analysis tasks to restaurant reviews, which can lead an interesting study of restaurants market and lately applying to other topics.

Keywords: Aspects, Sentiments Analysis, Restaurant Reviews, Linked Data, Natural Language Processing, Python

Agradecimientos

Lo primero dar las gracias a mi madre, ella ha sido la que más me ha apoyado para poder terminar esta carrera y la que siempre ha creido en mi, de forma que ha conseguido que realizara este trabajo. Por ahí va eso, gracias Mamá. También resaltar el apoyo de mis hermanos Pablo, María, David e Isabel que ha sido también fantástico, y también a todos mis cuñados, de entre ellos a Moisés que se ha interesado especialmente por este trabajo.

Por supuesto, dar las gracias a mi tutor Óscar, que con su ayuda y apoyo ha ayudado en gran medida al desarrollo de este trabajo. Él ha tenido que aguantar muchas horas de mis dudas, aconsejándome y guiándome en temas de los que yo todavía poco sabía. A esta labor se ha sumado también Carlos Ángel, cuya perseverancia, apoyo y empuje ha ayudado a llevar este trabajo a buen término.

Además dar las gracias a mi amigo Alberto por su inestimable ayuda y a Ganggao, por su generosa contribución a este proyecto también.

Tambíen dar las gracias a mis amigos de la escuela y los de fuera de ella. Ellos han sido los que han aguantado mis quejas, agobios y demás. Especialmente a mis amigos de toda la vida Fran y a Patricia que siempre han estado ahí apoyándome y a Bea, que me ha aguantado y ayudado estos últimos meses.

Por último y más importante, dedicar este trabajo y esta carrera a la memoria de mi padre.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This Chapter introduces the concept of sentiment analysis, one of the most important tasks in Natural Language Processing, providing a brief introduction of its context in both the academic and business environments. Given the context of the sentiment analysis problem and its main approaches, we list the project goals of this thesis and how this document is structured.

1.1 Context

In recent years, there has been a massive increase in social content caused by the proliferation of social networks, blogs or review sites. Millions of people express uninhibited opinions about various product features and their nuances. This generates a constant active feedback to the products, being vital for companies developing them as well as rivals and potential customers.

Sentiment analysis tasks allows tapping this goldmine of information. It retrieves opinions about certain products and classifies them as positive or negative. Sentiment analysis tasks are applicable to different levels: document (whole opinion), sentence (positive or negative sentence) or even aspect, that is the target of this thesis. Aspect-Based Sentiment Analysis (ABSA) determines the sentiment of an opinion regarding certain dimensions or nuances. It aims the extraction and classification of the sentiment and specific opinion about an aspect, that can be a certain entity, a concept, a topic, or in general, each analysis dimension that could be interesting.

The topic we are managing is restaurant reviews, because this is a domain which facilitates the classification of aspects, that is, when there is a restaurant review, the possible topics are severely limited. Also, the vocabulary is singular and specific which allows extracting the aspects easily from the reviews.

1.2 Project goals

In the long term, this project aims at introducing in the aspect-based sentiment analysis developing a prototype that allows extracting and classifying aspects from restaurant reviews. In addition, we will develop a dashboard with interactive widgets to understand the results of the analysis.

Among the main goals inside this project, we can find:

- Learning intelligent systems technologies and data processing.
- Design a system architecture of an aspect-based sentiment analyzer.
- Develop and evaluate a prototype.
- Validate the prototype in a case study.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is the following:

Chapter 1 explains the context in which this project is developed. Moreover, it describes the main goals to achieve in this project.

Chapter 2 provides a description of the main technologies on which this project relies.

Chapter 3 describes the architecture of the project, including the design phase and implementation details.

Chapter 4 presents experimentation and evaluation results.

Chapter 5 describes the system evaluated in a case study.

Chapter 6 discuss the conclusions drawn from this project, problems faced and suggestions for future work.

CHAPTER 2

Enabling Technologies

In the chapter that follows, it will be described the technologies used in the project. Firstly, all the tools and pipes used for the analysis of aspects will be introduced. Secondly, we will give insight into technologies which make possible the visualization of results.

2.1 IXA pipes

IXA pipes[1] is a modular set of Natural Language Processing tools (or pipes) developed by the IXA NLP Group of the University of the Basque Country [1]. Specifically, three of these pipes will be used: Ixa pipe tok, Ixa pipe pos and Ixa pipe nerc.

To use this tool, we will be using the default format provided by IXA pipes, the NAF format [2]. The NAF format is a stand-off, multilayered annotation schema for representing linguistic annotations. NAF uses URIs extensively and can be converted to RDF-NAF, that can be read by RDF parsers. Examples of NAF format will be given later.

2.1.1 Ixa pipe tok

Ixa pipe tok[2] is a multilingual rule-based tokenizer and sentence segmenter written in Java language. Ixa pipe tok outputs tokenized and segmented text in three formats: NAF, One-line and Conll. This tool also provides normalization functions to comply with annotation in corpora such as Penn Treebank for English and Ancora Corpus for Spanish. Besides, this pipe can be used as a server, in a micro-service manner.

Now will explain how ixa pipe tok works. To do this, we will introduce the phrase: "It was good Lasagna" on the standard input. The tool outputs the following NAF document 2.1

Listing 2.1: Example tokenization

```
<?xml version="1.0" encoding="UTF-8"?>
<NAF xml:lang="en" version="v1.naf">
    <wf id="w1" offset="0" length="2" sent="1" para="1">it</wf>
    <wf id="w2" offset="3" length="3" sent="1" para="1">was</wf>
    <wf id="w3" offset="7" length="4" sent="1" para="1">good</wf>
    <wf id="w4" offset="12" length="7" sent="1" para="1">Lasagna</wf>
</NAF>
```

[1] http://ixa2.si.ehu.es/ixa-pipes/
[2] https://github.com/ixa-ehu/ixa-pipe-tok

2.1.2 Ixa pipe pos

Ixa pipe pos[3] is a multilingual Part of Speech tagger and Lemmatizer written in Java language. Ixa pipe pos provides statistical POS (Part of Speech) tagging and lemmatization in several languages. It is based on Perceptron (Collins 2002) and Maximum Entropy (Ratnapharki 1999) POS tagging and Lemmatization models. To do this, ixa pipe pos use the machine learning API provided by the Apache OpenNLP project. As ixa pipe tok, ixa pipe pos can be deployed as a server that listens to client requests.

In the previous section, we introduced the sentence "It was good Lasagna" in the tokenizer. To explain how ixa pipe pos works, we will use the ixa pipe tok NAF document outputted before. Thus, the output of an ixa pipe pos trained with universal morphological models will be like 2.2 where the POS of each token can be seen.

Listing 2.2: Example POS tagging

```
<terms>
<term id="t1" type="close" lemma="it" pos="Q" morphofeat="PRP">
</term>
<term id="t2" type="open" lemma="be" pos="V" morphofeat="VBD">
</term>
<term id="t3" type="open" lemma="good" pos="G" morphofeat="JJ">
</term>
<term id="t4" type="close" lemma="lasagna" pos="R" morphofeat="NNP">
</terms>
```

2.1.3 Ixa pipe nerc

Ixa pipe nerc[4] is a multilingual Sequence Labeler for tasks such as Named Entity Recognition (NERC), Opinion Target Extraction (OTE) and SuperSense Tagging (SST).

The functionality concerning this project is OTE. The OTE module of this pipe uses models trained on the SemEval 2014 and 2015 datasets[5] and on Brown [3], Clark [4] and Word2Vec [5] clustering features. OTE reads NAF documents previously tokenized and POS tagged via standard input and outputs opinion targets in NAF. To get the necessary input for OTE, it could be done piping ixa pipe tok and ixa pipe pos before. Besides, OTE can run as a server too.

[3]https://github.com/ixa-ehu/ixa-pipe-pos
[4]https://github.com/ixa-ehu/ixa-pipe-nerc
[5]http://alt.qcri.org/semeval2015/task12/

In the same way as before, retaking the sentence "It was good Lasagna", the output of this pipe with the phrase is displayed. Note in 2.3 that the tool correctly detects the opinion target, *Lasagna*:

Listing 2.3: Example OTE

```
<opinions>
  <opinion id="o1">
    <opinion_target>
      <!--Lasagna-->
      <span>
        <target id="t4" />
      </span>
    </opinion_target>
  </opinion>
</opinions>
```

2.2 Context detector system

The module used in order to tackle this problem is an implementation of a graph based algorithm (Mukherjee and Bhattacharyya, 2012) for feature specific sentiment analysis [6]. This tool is part of Intelligent Systems Group (GSI) participation at Universidad Politécnica de Madrid (UPM) in the Sentiment Analysis work-shop focused in Spanish tweets, TASS2015 [7, p. 4]. It allows us to extract sets of words related to an aspect from a sentence, even if this sentence has different aspects and mixed emotions. The context of an aspect is the set of words related to that aspect.

This technology requires a parser to work. For this project, we have used the parsing software included in Stanford Core NLP detailed below. Besides, in this section a detailed description of the context detector system is presented too.

2.2.1 Stanford CoreNLP pipeline

Stanford CoreNLP[6] provides a set of natural language analysis tools. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize and interpret dates, times, and numeric quantities. It can also mark up the structure of sentences in terms of phrases and word dependencies, and indicate which noun phrases refer to the same entities. It was originally developed for English, but it also

[6]http://stanfordnlp.github.io/CoreNLP/

provides support for several languages nowadays. The Stanford CoreNLP code is written in Java and is integrated as a framework, which make it very easy to apply a variety of language analysis tools to a piece of text [8].

For this project we will be using CoreNLP server which provides both a convenient graphical way to interface with an installation of CoreNLP and an API with which to call CoreNLP using any programming language. To deploy this server we will use Docker[7], a technology which automates the deployment of applications inside software containers. Docker provides an additional layer of abstraction and automation of operating-system-level virtualization on Linux. CoreNLP server will be deployed in one of this mentioned containers.

The reason for choosing Stanford Parser is that this parser not only provides us the sentence tokenized and the POS analysis, but it also gives the dependencies among words, fundamental fact for our project.

Example of dependencies parsed by Stanford parser

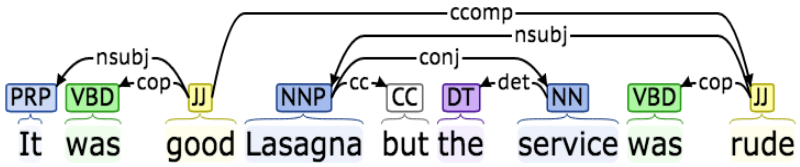

Figure 2.1: Dependency tree of a simple sentence

2.2.2 Graph dependency parser

The graph dependency parser module uses the dependencies provided by the Stanford Parser to express the dependency tree more compactly. Then, creates a matrix in order to correctly express all the relations in the graph. The rows and columns of this matrix represent the words of the sentence and its contents the distance to each other. To obtain this distances, the shortest paths in the dependency graph are computed with Dijkstra's algorithm.

[7] https://www.docker.com/

$$
G = \begin{array}{c} \\ It \\ was \\ good \\ Lasagna \\ but \\ the \\ service \\ was \\ rude \end{array}
\begin{array}{c}
\begin{array}{ccccccccc} It & was & good & Lasagna & but & the & service & was & rude \end{array} \\
\left(\begin{array}{ccccccccc}
0 & 2 & 1 & 3 & 4 & 5 & 4 & 3 & 2 \\
2 & 0 & 1 & 3 & 4 & 5 & 4 & 3 & 2 \\
1 & 1 & 0 & 2 & 3 & 4 & 3 & 2 & 1 \\
3 & 3 & 2 & 0 & 1 & 2 & 1 & 2 & 1 \\
4 & 4 & 3 & 1 & 0 & 3 & 2 & 3 & 2 \\
5 & 5 & 4 & 2 & 3 & 0 & 1 & 4 & 3 \\
4 & 4 & 3 & 1 & 2 & 1 & 0 & 3 & 2 \\
3 & 3 & 2 & 2 & 3 & 4 & 3 & 0 & 1 \\
2 & 2 & 1 & 1 & 2 & 3 & 2 & 1 & 0
\end{array}\right)
\end{array}
$$

2.2.3 Dependency extraction

The context detection algorithm 1 makes use of the graph of dependencies in order to calculate the set of words that express an opinion regarding an aspect. To do this, creates a cluster for each feature which will be the clusterhead. Then, assigns the word to the cluster whose clusterhead is closest to it.

As a result, for detecting the contexts in sentence "It was good Lasagna, but the service was rude", and knowing that the features are "Lasagna" and "service" , the tool computes the graph matrix and then applies this context detection algorithm obtaining the following set of words for each feature:

Lasagna$\rightarrow$ *It was good Lasagna, but*

service$\rightarrow$ *the service was rude*

2.3 Sematch

Sematch[8] is a framework for entity search in the knowledge graph that combines natural language query processing, entity linking, entity type linking and semantic similarity based on query expansion [9]. The system has been validated in a dataset and a prototype has been developed which translates natural language queries into SPARQL[9]. There is also an available on-line demo[10] for testing.

[8] https://github.com/gsi-upm/sematch

[9] https://www.w3.org/TR/rdf-sparql-query/

[10] http://demos.gsi.dit.upm.es/sematch/

The Sematch tool kit aims to provide a framework for matching required information from public Knowledge Graphs (KG) semantically. It provides core tools for these tasks:

- Semantic similarity of concepts in KG.
- Entity linking and disambiguation with KG.
- Semantic parsing.
- Question answering based on knowledge graph.

The great potential of Sematch for our project lies in the capability to find the similarity between the feature and the possible thematic category it belongs. To accomplish this, Sematch make use of technologies such as WordNet and NLTK. Following, these technologies are briefly explained.

2.3.1 WordNet

WordNet is a large lexical database for the English language[11]. It groups English words into sets of synonyms called synsets, each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. Moreover, Wordnet provides short definitions and usage examples, and records a number of relations among these synonym sets or their members [10].

Wordnet can be used to determine the similarity between words. Various algorithms have been proposed. One of them is measuring the distance among the words and synsets in WordNet's graph structure, such as by counting the number of edges among synsets. Another is with the intuition, that is based in the following hypothesis: the closer two words or synsets are, the closer their meaning. This second algorithm is used in our project. A number of WordNet-based word similarity algorithms are implemented in Python package called NLTK, that we explain below.

2.3.2 NLTK

NLTK[12] is a leading platform for building Python programs to work with human language data. It is developed by the University of Pennsylvania and in the current version provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a

[11] https://wordnet.princeton.edu/
[12] http://www.nltk.org/

suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, etc. With NLTK, it is possible to easily use the Wordnet corpus explained before.

2.4 Senpy and Linked Data

Senpy[13] is an open source reference implementation of a linked data model for sentiment and emotion analysis services based on semantic vocabularies NIF, Marl and Onyx. Senpy can use several formats such as turtle, JSON-LD and XML-RDF.

JSON-LD[14] is a method of encoding Linked Data using JSON. The data is serialized in a way that is similar to traditional JSON. JSON-LD is designed around the concept of a *context* to provide additional mappings from JSON to an RDF model. The *context* links object properties in a JSON document to concepts in an ontology, as Marl or NIF.

RDF[15] is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata model for information interchange on the Web. RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link.

Before proceeding to explain the use of Senpy, it will be necessary to introduce briefly the semantic vocabularies used by this framework:

- Marl[16], a vocabulary designed to annotate and describe subjective opinions expressed on the web or in information systems.
- Onyx[17], which is built one the same principles as Marl to annotate and describe emotions, and provides interoperability with Emotion Markup Language.
- NIF 2.0[18], which defines a semantic format and APO for improving interoperability among natural language processing services.

Senpy proposes a modular and dynamic architecture that allows:

- Implementing different algorithms in a extensible way, yet offering a common interface.

[13] https://github.com/gsi-upm/senpy
[14] http://json-ld.org/
[15] https://www.w3.org/RDF/
[16] http://www.gsi.dit.upm.es/ontologies/marl/ns#
[17] http://www.gsi.dit.upm.es/ontologies/onyx#
[18] http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#

- Offering common services that facilitate development.

The framework consists of two main modules: Senpy core, which is the building block of the service, and Senpy plug-ins, which consists on a number of NLP algorithms. 2.2 depicts a simplified version of the processes involved in an analysis with the Senpy framework. The tool extracts the parameter from a NIF HTTP query and executes the code of the plug-in selected with the inputted parameters previously validated. Then, use models to output a linked data publication in the desired format.

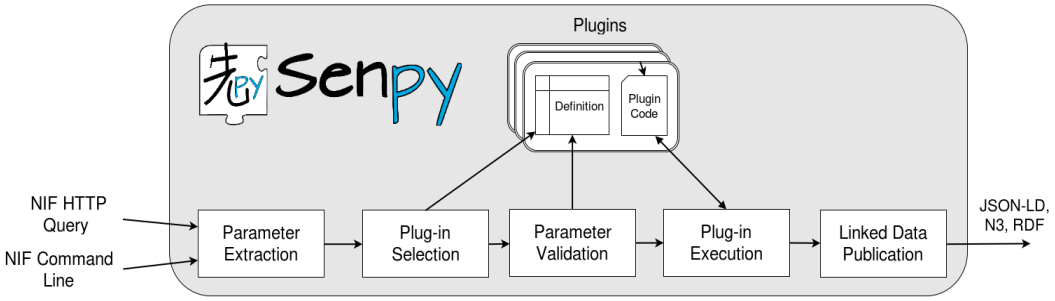

Figure 2.2: Senpy framework architecture

2.5 Sefarad

Sefarad is a web-based data visualization and browsing application implemented by Intelligent Systems Group (GSI) at Universidad Politécnica de Madrid (UPM) for consultation and visualization of semantic data [11]. It is developed to explore linked data by making SPARQL queries to the chosen endpoint without writing more code. In this way, it provides a semantic front-end to Linked Open Data.

It allows the user to configure his own dashboard with many widgets to visualize, explore and analyze graphically the different characteristics, attributes and relationships of the queried data.

Sefarad is based on Web components. Web Components[19] are a set of standards currently being produced by Google engineers as a W3C specification that enables the creation of reusable widgets or components in web documents and web applications. The intention behind them is to bring component-based software engineering to the World Wide Web. The component model enables encapsulation and interoperability of individual HTML elements.

[19] http://webcomponents.org/

In this project, we will be using a dashboard developed from Sefarad 3.0 [12] for sentiment analysis based on Polymer and D3.js. Polymer is an implementation of previously mentioned Web Components and D3.js is a JavaScript library for manipulating documents based on data.

CHAPTER 3

Architecture

This chapter describes the architecture of this project, including the design phase and implementation details. Firstly, we will present a global vision about the project architecture, identifying the different modules that integrates the system. Secondly, we will describe the modules which compose the sentiment analysis prototype, showing its functionality in depth and how they have been put together to set up the analyzer. Finally, we will explain how all modules are implemented in a sentiment analysis pipeline with a Senpy plug-in and the performance of visualization system.

3.1 Overview

First of all, throughout this chapter we will refer to some terms recursively that would be useful to specify. The term aspect, also called feature, is used when referring to different nuances of a sole opinion. Furthermore, it is important to realize that every aspect is contained in one or more words which has the core of aspect information. This system is focused on extracting and analyzing these aspects. Once this term has been explained, we turn to describe the system architecture. The system is composed of the following modules.

- ***Opinion target extraction***: this module is in charge of finding the target of an opinion. The opinion will be extracted from a product review.
- ***Context finder***: once we have the target, we must obtain the context of target.
- ***Topic Detector***: in order to analyze the aspects, this module classifies the contexts from Context finder adhering to six categories given.
- ***Sentiment analyzer***: this module is responsible for determining the polarity of aspects.
- ***Visualization system***: this part handles the results of Aspects analysis and let us to visualize them in a dashboard.

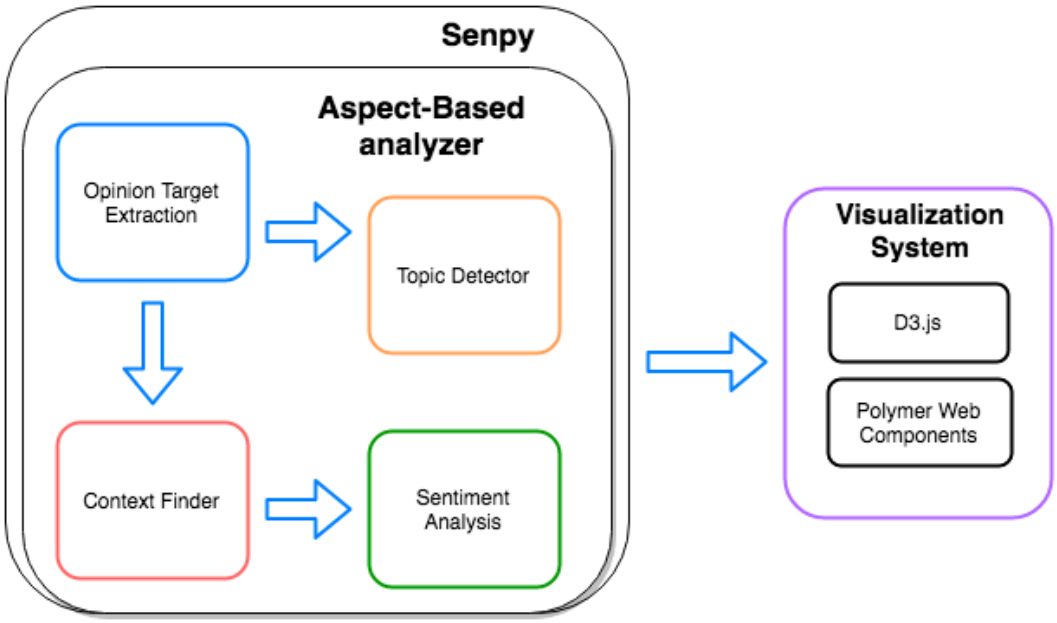

Figure 3.1: System general architecture

To implement this system, we have used Python[1]. As a scripting language with module architecture, simple syntax and rich text processing tools, Python is often used for natural language processing tasks so is proper to our project.

3.2 Opinion target extraction

The main goal of this module is to identify the targets of a product review. To do so, we use tools called IXA pipes [1]. Specifically, we use one called Ixa pipe nerc [1, p. 3]. This module is the starting point of the project and the most important because it is the key to identify aspects. Henceforth, we will call this module with the acronym OTE (Opinion Target Extraction).

The procedure to extract the target requires a model to identify the aspects of entities and the sentiment expressed for each aspect. The one used in this prototype is a model provided by the tool and trained on SemEval 2015 Task 12 corpora for the restaurant domain [13].

Moreover, Ixa pipe nerc use a special format called NAF (Newsreader Annotation Format). The NAF format is a linguistic annotation format designed for complex NLP pipelines. Hereby, other modules are required to provide ixa pipe nerc an appropriate document (NAF) to work with. The reason to this is that the Ixa pipe nerc needs tokenized and morphological analyzed text, not plain text.

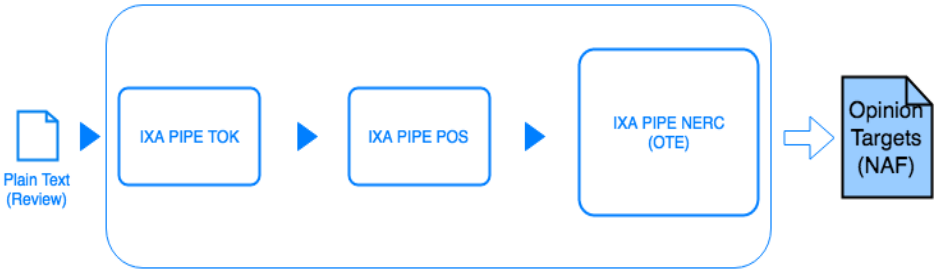

Figure 3.2: Opinion targets extraction process

Figure 3.2 shows the architecture of Opinion target extraction module. The process performed in the module begins with the input of a restaurant review. This text input will be processed by the ixa pipes:

- Ixa pipe tok will be in charge of tokenizing and segmenting the text from review providing us information about where the word is located in the text. This is important

[1]https://www.python.org/

information in order to carry out the implementation with other modules.

- Ixa pipe pos will tag each token providing information about lemma and morphology.
- Ixa pipe nerc is the main pipe of the pipeline and uses all the information provided from the previous pipes to extract the aspect of a sentence.

Regarding the NAF format provided by IXA, it is important to know that is based on XML, which facilitates the extraction of information. Because of this, the following modules will be able to parse the NAF file to easily obtain the opinion target for each aspect of the review.

Having defined what is meant by IXA pipes, we will now move on to explain how this module can be deployed. All the IXA modules can be set up as a TCP socket so they can be used in a micro-service environment. The server returns a NAF document with the opinion targets. This file serve as input to the context finder module. To compose the request, we have used the Socket module for Python language[2]. In figure 3.3 we can see the communication process with the pipes.

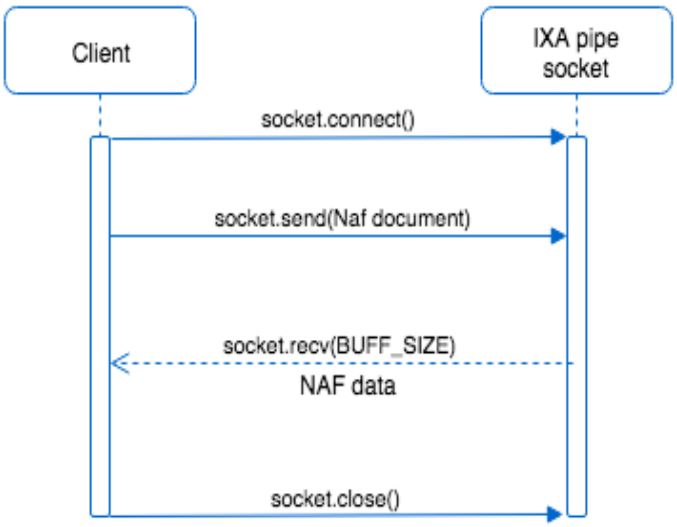

Figure 3.3: IXA pipe socket communication process

To test and implement the module, we use the restaurant reviews from SemEval 2015 Train Corpora [13]. This tagged corpora allows us to know which is the opinion target for each sentence. Every sentence in the Corpus is parsed and passed as a request to the server in order to estimate the accuracy of Ixa pipe nerc. In the next section, it will be argued in depth with the experiment results.

[2]https://docs.python.org/2/library/socket.html

3.3 Context finder

Once the opinion targets have been extracted, we need to find the context of each target in order to delimit the scope of aspects. This module is in charge of detecting the context of the aspects in the review documents.

Before proceeding to dissect the performance of this module, it is necessary to cover the idea of dependency tree of a sentence. Dependency parsing represents the syntactic relationships between words in a sentence. Figure 3.4 shows an example of a result of dependency parsing and POS tagging. In order to extract the dependency trees of the dataset, we have used the Stanford parser.

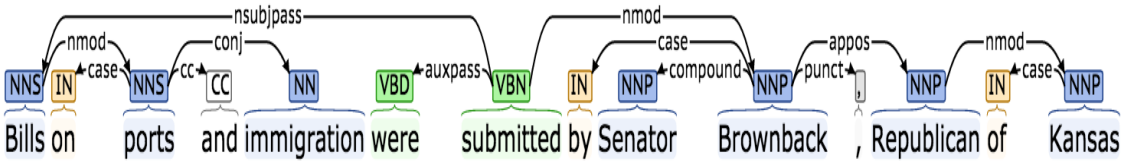

Figure 3.4: Parsed dependencies of a simple sentence

3.4 is not representative enough to illustrate what the dependency tree is. To exemplify it properly, in figure 3.5 the dependency tree extracted from the parsed dependencies of the previous sentence is shown[3]. It can be seen that the parser spreads the links between words basing on the POS tagging to form the complete tree.

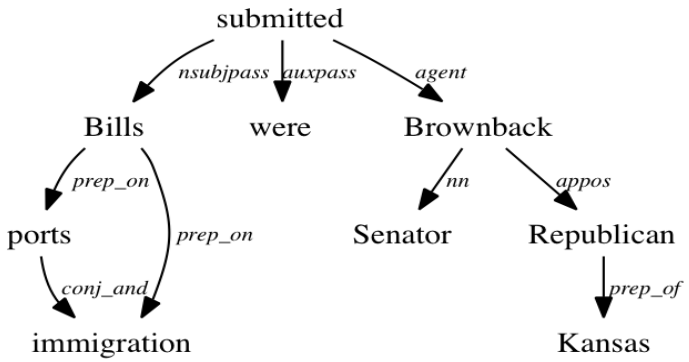

Figure 3.5: Dependency tree

[3]http://nlp.stanford.edu/software/stanford-dependencies.shtml

Turning now to the behaviour of this module, the first task done by it will be parsing the dependencies mentioned before. This task will be executed by Stanford CoreNLP Parser. This parser has been implemented in the module deploying CoreNLP Server through a dockerized image of it.

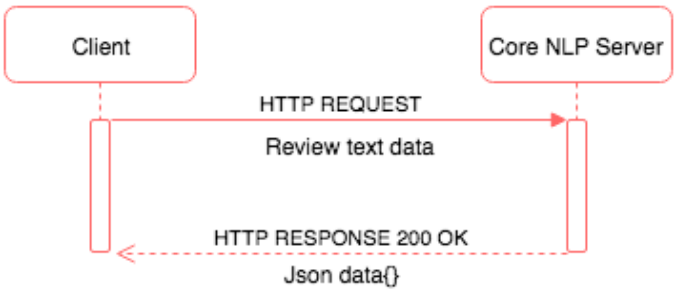

Figure 3.6: CoreNLP server communication process

A parameter *properties* is given to the parser to let it know which tools from CoreNLP we want to use, in our case, the parser. Besides, it can be seen that the output format selected is JSON.

Context detector uses the collapsed dependencies provided by the parser (An example of these dependencies type provided by Stanford parser is given in figure 3.4), but the module needs more information to work properly, the words associated with each aspect. To provide the context detector with each aspect's words, the JSON output from stanford parsed has been modified to shows if a word is associated with an aspect or not. To do this, we use the output of the previous module, OTE. The NAF document outputted by the OTE module is parsed to learn which words are an aspect in the review document comparing the offsets of tokens parsed by Stanford with the offset of each aspect word detected by IXA.

With the modified collapsed dependencies, the module creates a matrix as that illustrated in section 2.2.2. This matrix offers a measure of distance between words attending to number of arcs that link each word in the dependency graph, when going from one word to another, that is, the distance $d(w_i, w_j)$ between two words w_i and w_j in the shortest path.

At this step, the module has a graph with the extracted dependency tree and a set of aspects defined by his aspect word or set of words. We need to detect which word belongs to every aspect context. As mentioned in section 2.2.3, to do this, in this project, we use the implementation of an algorithm for feature specific sentiment analysis [6].

This implementation is done in this Master Thesis [14]. As detailed in that work, if there are n aspects of a product in a sentence, then the word that are more close (in terms of these dependency distances) to an aspect i will come together to express some opinion

about it, rather than about another aspect j.

In general, as explained in [6], there are n aspects where n is the dimension of A. The algorithm for extracting the set of words $w_i \in S$ that express any opinion about the target aspect at proceeds as follows:

Algorithm 1 Dependency extraction algorithm

(i) Initialize n clusters $C_i \forall i = 1..n$

(ii) Make each $a_i \in A$ the clusterhead of C_i. The target aspect a_t is the clusterhead of C_t. Initially, each cluster only consists of the clusterhead.

(iii) Assign each word $w_j \in S$ to cluster C_k s.t.
$k = arg\ min_{i \in n} d(w_j, a_i)$

(iv) Merge any cluster C_i with C_t if $d(a_i, a_t) < \theta$ where θ is some threshold distance.

(v) The set of words $w_i \in C_t$ expresses the opinion regarding the target aspect a_t.

In other words, n clusters are initialized, each cluster C_i corresponding to each feature $a_i \in A$, being a_i the clusterhead of C_i. Then, each word $w_i \in S$ is assigned to the cluster whose clusterhead is closest to it. After this, any cluster is merged with the cluster whose clusterhead is the target aspect if the distance between their clusterheads is lower than a threshold θ. Finally, the set of words in the cluster C_t give the opinion about the aspect a_t.

As explained earlier, for our project we use a tool developed in the Master Thesis mentioned before that implements this algorithm. The tool will use the extracted dependencies matrix and the aspects extracted from OTE module. Thus, the tool goes through objects in the array of aspects provided by the previous module and assign each word to the aspect or feature cluster with the method "extract_dependencies" [14]. The distance between the aspect context and another aspect context is compared with the threshold value. Depending on this, these two contexts will be joined or not. By default, the threshold will be 2.

So far this section has focused in describing the behaviour of this module but there are another important issues concerning the integration with first module that are mentioned below.

The principal problem found in facing integration was that aspects could be composed by more than one word, for instance, in the sentence "Simply some good tasting Chinese food at incredible prices". The aspect in this example will be "chinese food", not only "food". Indicated this problem, it can be observed that conversely, the context detector

processes one word per aspect, not a set of words. In spite of this behaviour, we have taken advantage of using OTE. Thus, we propose a solution to this problem, consisting on joining the words that forms the same aspect and merging their contexts.

However, the implementation of this solution leads to comparing the words whose position in the text is not known, so it is not able to detect and merge aspects composed by more than two equal words in the same document. Additionally, another problem is presented. That is, the way of proceeding in tokenizing texts in Ixa pipe tok is significally different to that of the Stanford module. Concretely, the parsing of a regular word merged with posterior punctuation. To fix this lack of correspondence, the output from Stanford parser is cleaned, removing the punctuation symbols such as "?.¿!".

To sum up, we have seen in depth the proficiency of context detector module and his aggregated tools. To continue explaining the architecture of this project, is important to know that at this step, as shown in 3.1, the corresponding output of this module will be the scoped aspects defined by the words which compose its context and its word or set of words.

Listing 3.1: Output of Context Finder example

```
{
"context": ["that","no","one","in","the","restaurant","has","any","idea","about","or","
    experience","with","Japanese","cuisine."],
"feature": "Japanese cuisine"
}
```

3.4 Topic detector

This part of the system is in charge of detecting the aspect's topic. In principle, the idea of detecting the topic of an aspect which is not referred to a particular concept may seem complicated, however, detecting topics among a few of characteristics from a concrete object is possible.

This project is focused on a particular dimension of topics from the restaurant domain. As a result, the task of classifying aspects by their topics is restricted to a number of categories. For this, we have used a solution to this issue based on tool Sematch [9]. The solution is a standalone module[4] based on this tool that is capable of detecting the topic among six categories, "Restaurant", "Food", "Drinks", "Ambience", "Location" and

[4]https://github.com/gsi-upm/sematch/

"Service". This six categories are extracted from the dataset of SemEval 2015 used in the project [5].

To choose between these categories the module needs to know what is the word that represents the aspect so that this is the data passed to the module, the opinion targets extracted by OTE module. However, the module also needs a dictionary with the keywords of each category to have a general notion of which words belongs to each category. The dictionary has been trained specifically with words from the SemEval 2015 corpora for this project.

It is interesting to briefly report the formation of this dictionary to understand how the topic detector choose among six categories. To train it, the Nltk library is used to eliminate from the sentences of dataset the noisy words that not contribute at all to the topic, that is to say, the stopwords. The library Corpus[6] from Nltk provides a good dictionary of stop words that is used to keep only the words with a concrete meaning.

Once we have extracted from the dataset only the significant words of each category, the next step is to find the most common among these words. Nltk provides the necessary to achieve this, calculating the frequency of each word in a list. Then, the only remaining is choosing between the most common words, the ones which only appears in one category. We have selected this approach because we want to choose between the categories, so, the words that appears in several categories, are not a characteristic word of the category and does not a support in the dictionary.

To conclude, it is highlighted that the dictionary trained has about one hundred words and to improve the system's performance, it could be convenient to add more but this task will be shown in the next chapter.

Now we will explain the behaviour of the module. Once the aspect word or words are introduced, Sematch module extracts the lemma of every word that forms the aspect using a WordNet Lemmatizer. To use it, the tool takes advantage of Nltk core tools, specifically, Nltk stem[7], a library for executing tasks as extracting lemma of words. After doing this, the next step is to find the similarity between every word from the dictionary with the characteristic words of each category mentioned earlier and the aspect.

Reducing the problem to a minimum, the necessary task performed is to find the similarity between two words. Sematch achieves this by using Wordnet corpora which provides

[5] http://alt.qcri.org/semeval2015/task12/data/uploads/semeval2015_absa_restaurants_annotationguidelines.pdf
[6] http://www.nltk.org/api/nltk.corpus.html
[7] http://www.nltk.org/api/nltk.stem.html

synsets[8]. Synsets are sets of synonymous terms that the tool finds, if there is not a synset stored in Wordnet for specific word, it stems the word with Porter algorithm[9], another tool provided by Nltk stem, and tries again to find a Wordnet Synset. Consequently, it is necessary to find synsets for the two words, and then, paths the similarity between then. Finally, this is an easy task to solve, since Wordnet provides a method to find similarity between its synsets.

Knowing the similarity between aspect and one word from the dictionary, in the same way, is easy to know this for each word from a category. The category that has more words with more similarity with the aspect, will be the chosen one.

To implement Sematch tool in our project, we use a Python class called Aspect [10] [9]. With this class, aspect objects can be created. This objects allow finding the similarity mentioned before and returns one of the six possible topics. To run the search of similarity, this objects need the aspect feature to be inputted. Of course, every aspect object created has loaded the dictionary of words from SemEval corpora which permits improve his performance. We provide a diagram to illustrate this process.

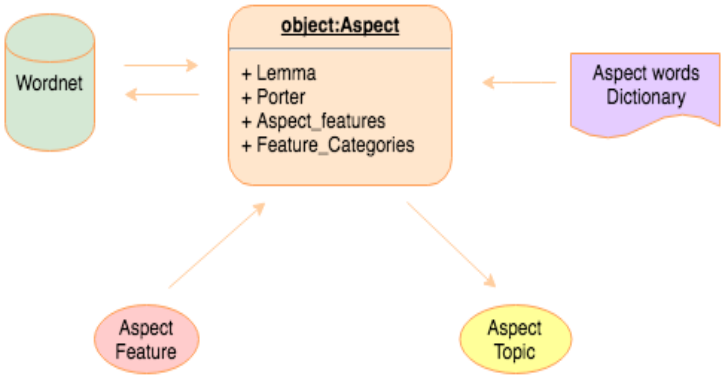

Figure 3.7: Topic detector architecture

3.5 Polarity analyzer

This module has been implemented using Senpy[11], a server for sentiment and emotion analysis in Python developed by the Intelligent Systems Group. For this module we have used a plug-in developed by GSI [15], sentiText plug-in, which distinguishes between positive, neutral or negative sentiment. We introduce the plug-in performance easily explaining the

[8] https://wordnet.princeton.edu/
[9] https://tartarus.org/martin/PorterStemmer/
[10] https://github.com/gsi-upm/sematch/blob/master/sematch/semantic/aspect.py
[11] https://github.com/gsi-upm/senpy

data flow:

- ***Input***: we want to analyze every aspect on each review document, so the input of the Senpy plug-in will be the contexts extracted in the Context finder.
- ***Output***: the plug-in, showing the output in semantic language, decides if each aspects as a *marl:hasPolarity*[12] positive or negative depending of the *marl:polarityValue*.

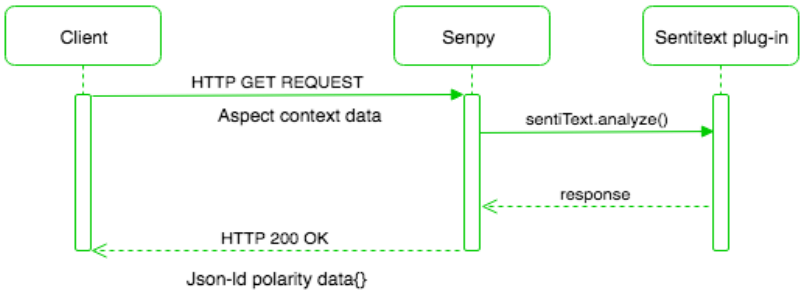

Figure 3.8: SentiText data flow example

3.6 Senpy plug-in

The purpose of this section is to approach the implementation of the previous four sentiment analysis modules in one service entirely functional. Senpy allows the creation of plug-ins that can be deployed in a Senpy server. In this project, we have created one plug-in to execute our system as a service.

To develop the plug-in, a Python module is created gathering the necessary tools to execute each module. To do that, we create a pipeline with OTE, Context finder, Topic detector and Polarity analyzer modules. These modules require libraries and servers to perform. All these procedures, that have been explained in previous sections,are included in the plug-in module.

The plug-in developed use semantic vocabularies Marl and NIF to gather all the information reached by the modules. As a result, the data outputted by Senpy server is JSON-LD format. The context of the JSON-LD data includes the URIs that Senpy provides for the necessary ontologies, including Marl and NIF. Also it is important to know that in the JSON-LD there are other URIs, that links with DBpedia. DBpedia[13] is a project aiming to extract structured content from the information created as part of the Wikipedia.

[12]http://www.gsi.dit.upm.es/ontologies/marl/
[13]http://wiki.dbpedia.org/

This dataset allows users to semantically query relationships and properties associated with Wikipedia resources.

With regard to the information of the aspects, is stored in an array of *Sentiments*, an object provided by the model of Senpy, in the JSON-LD. Each element of this array has the following fields:

- ***marl:describesObject***: this field contains the word or set of words that defines the aspect.
- ***marl:describesObjectFeature***: we associate the topic of the aspect to this Marl concept.
- ***marl:forDomain***: with this Marl item we link the topic of the aspect to a entity from DBpedia.
- ***marl:hasPolarity***: indicates the polarity of the aspect.
- ***marl:polarityValue***:this field provides the polarity value of the aspect.
- ***nif:anchorOf***: NIF item with the words that forms the context of the aspect.

Also, the JSON-LD output contains a field "nif:isString" with the text parameter introduced to Senpy server. To illustrate all this information, an example for the text with some aspects "The meat is fresh, the sauces are great, you get kimchi and a salad free with your meal and service is good too." is given in 3.2.

Listing 3.2: Output of Context Finder example

```
{
  "@context": "http://senpy.cluster.gsi.dit.upm.es/api/contexts/Results.jsonld",
  "@id": "Results_1467760011.061031",
  "analysis": [
    {
      "@id": "aspy",
      "@type": "marl:SentimentAnalysis",
      "author": "Manuel Garcia-Amado",
      "description": "Sentiment Aspect analyzer for restaurants",
      "extra_params": {
        "language": {
          "aliases": ["language","l"],
          "default": "en",
          "options": ["en"],
        "required": true
      }
```

```
    },
  "info": {...},
  "entries": [
    {
      "@id": "Entry0",
      "nif:isString": "The meat is fresh, the sauces are great, you get kimchi and a
          salad free with your meal and service is good too.",
      "sentiments": [
        {
          "@id": "Aspect0",
          "marl:describesObject": "meat",
          "marl:describesObjectFeature": "food",
          "marl:forDomain": "http://dbpedia.org/page/Food",
          "marl:hasPolarity": "marl:Positive",
          "marl:polarityValue": "1",
          "nif:anchorOf": "The meat is",
          "prov:wasGeneratedBy": "aspy"
        },
        {
          "@id": "Aspect1",
          "marl:describesObject": "sauces",
          "marl:describesObjectFeature": "food",
          "marl:forDomain": "http://dbpedia.org/page/Food",
          "marl:hasPolarity": "marl:Positive",
          "marl:polarityValue": "1",
          "nif:anchorOf": "fresh, the sauces are great, you get",
          "prov:wasGeneratedBy": "aspy"
        },
        {
          "@id": "Aspect2",
          "marl:describesObject": "meal",
          "marl:describesObjectFeature": "food",
          "marl:forDomain": "http://dbpedia.org/page/Food",
          "marl:hasPolarity": "marl:Positive",
          "marl:polarityValue": "1",
          "nif:anchorOf": "kimchi and a salad free with your meal and",
          "prov:wasGeneratedBy": "aspy"
        },
        {
          "@id": "Aspect3",
          "marl:describesObject": "service",
          "marl:describesObjectFeature": "service",
          "marl:forDomain": "http://dbpedia.org/page/Service",
          "marl:hasPolarity": "marl:Positive",
          "marl:polarityValue": "1",
          "nif:anchorOf": "service is good too.",
          "prov:wasGeneratedBy": "aspy"
        }
      ]
    }
  ]
}
```

3.7 Visualization system

The main goal of this module is to show restaurant reviews sentiment analysis, specifically the sentiment analysis of the aspects of a review. This visualization server is based on Polymer Web Components Library. This library allows the creation of widgets in web applications [16]. To do this, we have based on the work realized in [12] and developed a simple reduced version of it.

3.7.1 Mock-up

To make a mock-up of the dashboard desired we use the on-line tool draw.io [14]. This mock-up is realized with the aim of achieving an easy not overloaded interface that may give insight into the analysis realized with the aspect-based analyzer.

In figure 3.9 it can be seen the mock-up containing the main tab of the possible dashboard. In this tab we will show information about the review analysis. Also, there is another tab explaining the purpose of this dashboard.

The main tab contains three appreciable sections. One, at the top left of the image would be a graphic with the number of aspects classified by topic information. On the right of the previous, a graphic with the polarity of each aspects, also classified by its topic. Finally, at the bottom of the mock-up we find the review documents where we can recognize each aspect highlighted depending the topic, and the aspect indicated in bold.

3.7.2 Widgets

To make the widgets, we have based on widgets done for Sefarad [11] and Football analysis [12], which are done with D3.js technology. They have been adapted to our data. To implement the widgets, we provide to the visualization server JSON-LD[15] data with our aspect-based sentiment analysis. In section 3.6 we explained how obtaining this semantic data in JSON-LD format.

Each widget provides us different information of the aspects in reviews data. These widgets are described below.

[14] https://www.draw.io/
[15] http://json-ld.org/

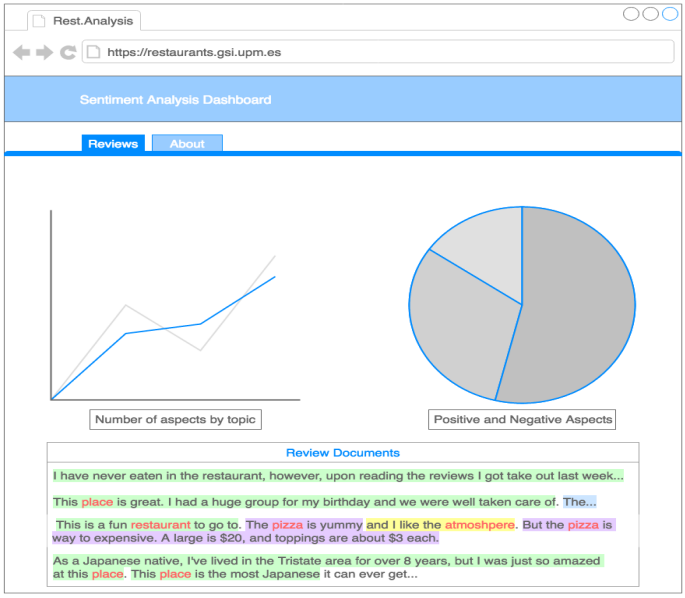

Figure 3.9: Dashboard mock-up

3.7.2.1 Topic Radar

This widget is designed to study the topics present in reviews. The axis in radar counts every analyzed review. The colored area shows the amount of aspects pertaining to different topic.

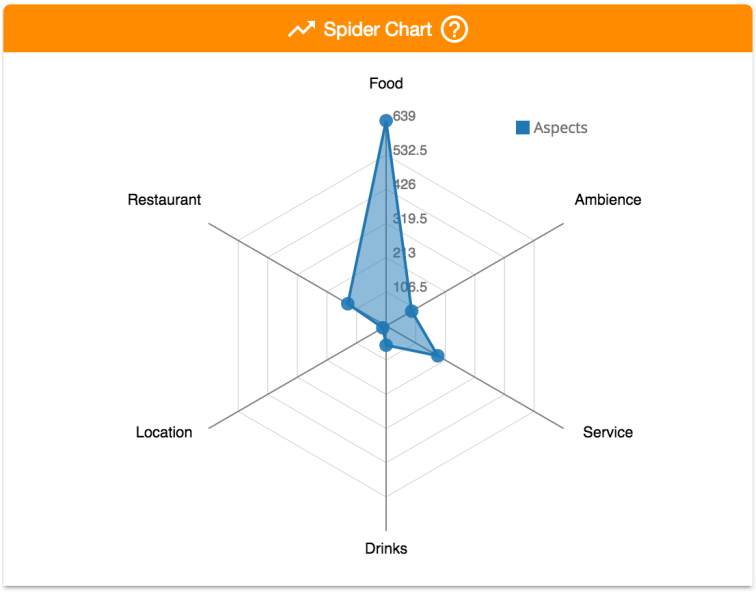

Figure 3.10: Topic Radar widget

3.7.2.2 Aspects Sentiment Wheel

This widget is designed to show the aspects according to its polarity. Moreover, we can see the aspects also classified by his topic. With the inner wheel, we choose between the six topics. In figure 3.11 we can see an example to clarify it, we have chosen the light green color that represents ambiance aspects.

Then, with the outer wheel it can be seen the text of aspect moving around the wheel, because aspects are scattered in the wheel classified by his polarity, positives on green, and negatives on red. All this data is indicated in the center of wheel.

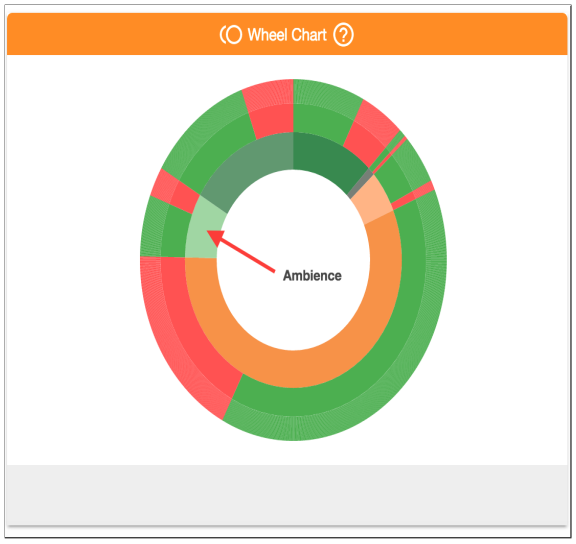

Figure 3.11: Aspect Polarity inner wheel

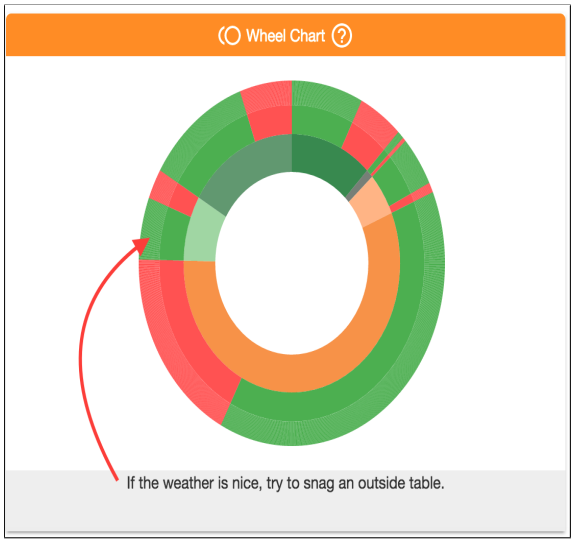

Figure 3.12: Aspect Polarity outer wheel

3.7.2.3 Reviews chart

This widget is used for showing the analyzed reviews. We have put the aspects of each review together and classify it by his topic, indicating which is the aspect in bold. As a result, each aspect has a different color and a word or set of words in bold.

Figure 3.13: List of analyzed reviews chart

CHAPTER 4

Experimental Study

The purpose of this chapter is to describe the process realized to test the system in order to achieve the best behaviour of it. It begins by analyzing the behaviour of each module and then comparing this behaviour with different module configurations. Finally, we discuss the improvement in the results from the aspects' analysis.

4.1 Evaluation data and measures

Once we have defined our system architecture, we need to test the aspect-based analyzer. Some of these tests have been done whereas we were developing the system but also we have done other evaluations to improve the basis behaviour of the system. These evaluations are possible due to the fact that some modules of the analyzer have different ways of working. Taking advantage of this, in those modules we have tested its different performances and looked at the general system improvements or the improvements in next modules of the pipeline.

To test and compare every module behaviours, we use the annotations from the SemEval 2015 dataset used for implement the analyzer [13]. This annotations include the topic, the polarity, the target and the scope of the aspects. A chunk of this dataset is presented in 4.1.

Listing 4.1: Evaluation dataset example

```
<Review rid="1004293">
  <sentences>
    <sentence id="1004293:0">
      <text>Judging from previous posts this used to be a good place, but not any longer
          .</text>
      <Opinions>
        <Opinion target="place" category="RESTAURANT#GENERAL" polarity="negative"   from
            ="51" to="56"/>
      </Opinions>
    </sentence>
    <sentence id="1004293:1">
      <text>We, there were four of us, arrived at noon - the place was empty - and the
          staff acted like we were imposing on them and they were very rude.</text>
      <Opinions>
        <Opinion target="staff" category="SERVICE#GENERAL" polarity="negative" from="75"
            to="80"/>
      </Opinions>
    </sentence>
    <sentence id="1004293:2">
      <text>They never brought us complimentary noodles, ignored repeated requests for
          sugar, and threw our dishes on the table.</text>
      <Opinions>
        <Opinion target="NULL" category="SERVICE#GENERAL" polarity="negative" from="0"
            to="0"/>
      </Opinions>
    </sentence>
    <sentence id="1004293:3">
      <text>The food was lousy - too sweet or too salty and the portions tiny.</text>
      <Opinions>
      <Opinion target="food" category="FOOD#QUALITY" polarity="negative" from="4" to
```

```
          ="8"/>
        <Opinion target="portions" category="FOOD#STYLE_OPTIONS" polarity="negative"
            from="52" to="60"/>
      </Opinions>
    </sentence>
      <sentence id="1004293:4">
      <text>After all that, they complained to me about the small tip.</text>
      <Opinions>
        <Opinion target="NULL" category="SERVICE#GENERAL" polarity="negative" from="0"
            to="0"/>
      </Opinions>
    </sentence>
      <sentence id="1004293:5">
      <text>Avoid this place!</text>
      <Opinions>
        <Opinion target="place" category="RESTAURANT#GENERAL" polarity="negative" from
            ="11" to="16"/>
      </Opinions>
    </sentence>
  </sentences>
</Review>
```

In some cases, the data outputted by the modules is difficult to match with the annotations from the dataset but in general, we compare more than eighty percent of the aspects, sufficient quantity to experiment with the tests.

With regard of the evaluation measures, to measure the sentiment analysis realized, we use F1-Score. This metric is the harmonic mean of two values, *precision* and *recall*. For defining these two values, the analysis task must be understood.

- ***Precision***: number of correct positive results divided by the number of all positive results.
- ***Recall***: number of correct positive results divided by the number of positive results that should have been predicted.

The F1-Score can be interpreted as a weighted average of these two values, reaching its best score at 1, and worst at 0. That is the measure used in this work.

4.2 Context finder

This section analyses and discusses the tests realized with the second module of the system, the context finder. We test the context finder module in an indirect manner, in order to simplify the measurement process. So, the way of proceeding to measure the behaviour of

this module is looking at the polarity analyzer module which uses sentiText plug-in [15]. In fact, context finder behaviour is studied in depth in [14], so we only give insight into a particular performance of the tool implemented in our system, that is, we measure the behaviour of both polarity analyzer and context finder together.

4.2.1 Context finder threshold

The algorithm behind context finder module uses a threshold to decide if merging two contexts. To do this, the distance between the aspect of each contexts must be less than mentioned threshold. In the module, this threshold is configurable. We have implemented the first version of the system with a threshold value of 2, but in order to find an improvement in the behaviour of polarity analyzer, we have experimented with other threshold values. In 4.1 it can be seen the different threshold values with its calculated F1 score.

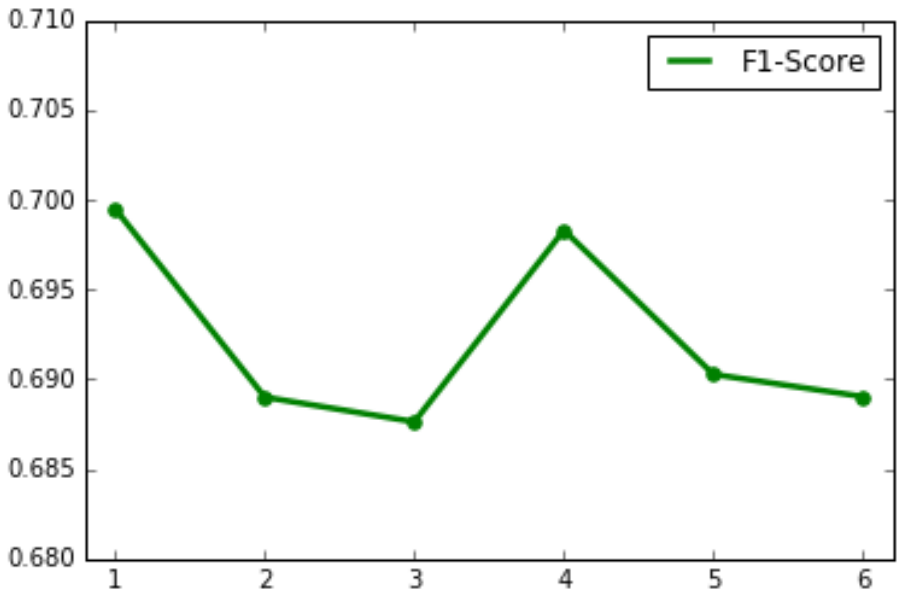

Figure 4.1: F1 score of different threshold values

At a first glance, looking at the graph, when threshold is 1 or 4, we achieve better performance in polarity analyzer. As a result of this analysis, we choose the threshold value of 1. In Table 4.1 it can be seen the data of measuring F1-Score with a threshold value of 1.

	Precision	Recall	F1-Score	Support
Negative	0.38	0.58	0.46	191
Positive	0.84	0.75	0.79	695
Avg / Total	0.72	0.69	0.70	886

Table 4.1: F1-score of polarity analyzer with 1 threshold value

4.2.2 Window experiment

OTE module gives the opinion targets of a sentence or aspect. The window span used to scope the aspect of this opinion target is given by the context finder. However, there is a simpler and primary way to do this task. It consists in defining a window span around a given opinion target. We have experimented with several sizes for this window to find the best results in polarity analyzer module. In 4.2 it can be seen the different window sizes with its calculated F1 score.

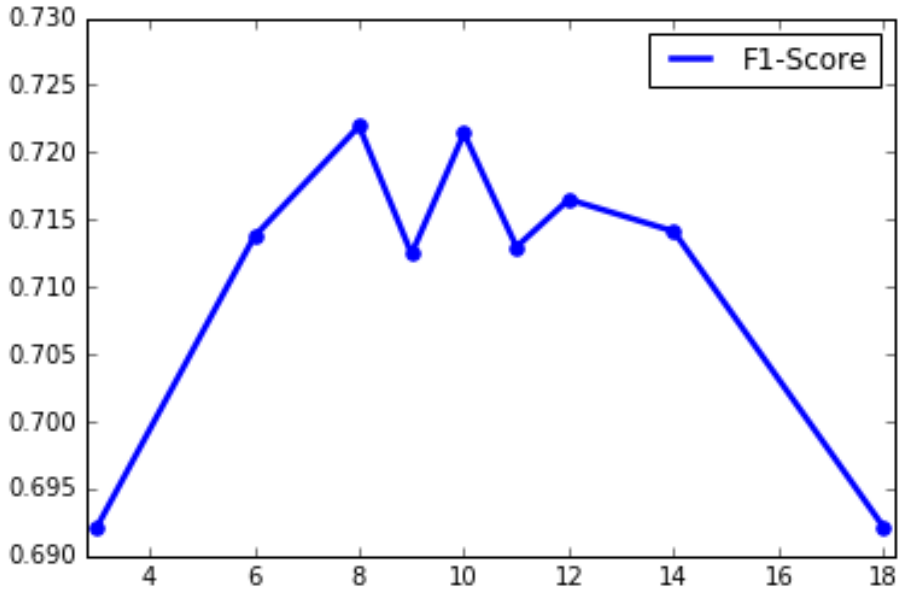

Figure 4.2: F1 score of different window sizes

We observe that the results are favorable with values of window close to ten. That could be the approximate maximum value of the polarity analyzer performance. The polarity analyzer behaves well with this window but the contexts obtained, in several occasions lose

their sense. In Table 4.2 it can be seen the F1-Score disaggregated to compare the results of this experiment with performance of context finder described in the previous section.

	Precision	**Recall**	**F1-Score**	**Support**
Negative	0.40	0.48	0.44	193
Positive	0.83	0.82	0.83	731
Avg / Total	0.72	0.73	0.72	924

Table 4.2: F1-score of polarity analyzer with an 8 window size

The results are very similar that the obtained with the context finder, except when the window are near to ten, that obtains a little more performance. So this experiment has concluded that the contexts found with the context finder are good because the performance of polarity analyzer with our context finder is near to its maximum.

4.3 Topic detector

The purpose of this chapter is to describe and discuss the experimentation with topic detector module. In this case, it is easy to compare the topics provided by this module with the annotations of the SemEval 2015 dataset. Despite of this, we can't compare the whole dataset but, nevertheless, we have obtained enough results with which compute a robust metric.

This module counts with a support dictionary to improve its performance. Once we had the dictionary made we try to improve its performance adding these words that are important for each topic. Then, we definitely established this dictionary and evaluated the two algorithms provided by Sematch to find similarity between words. This two methods are described below:

- ***Max similarity***: uses only the method provided by Wordnet to path similarity between the synsets and the words from the dictionary. With this algorithm, the F1-Score obtained is illustrated in 4.3
- ***K-NN classify***: uses k-nearest neighbors algorithm. The input for this algorithm consists of the k closest training examples in the feature space, that is, the aspect word

dictionary. The output is reached as describes. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. With this algorithm, the F1-Score is shown in 4.4.

	Precision	**Recall**	**F1-Score**	**Support**
Ambience	0.90	0.52	0.66	121
Drinks	0.89	0.95	0.92	44
Food	0.96	0.92	0.94	437
Location	0.40	0.60	0.48	10
Restaurant	0.71	0.62	0.66	128
Service	0.67	0.98	0.80	174
Avg / Total	0.85	0.83	0.83	914

Table 4.3: Topic detector Max similarity algorithm F1-score

	Precision	**Recall**	**F1-Score**	**Support**
Ambience	0.91	0.50	0.65	121
Drinks	0.98	0.95	0.97	44
Food	0.91	0.95	0.93	437
Location	0.60	0.60	0.60	10
Restaurant	0.71	0.63	0.67	128
Service	0.71	0.63	0.67	174
Avg / Total	0.85	0.84	0.84	914

Table 4.4: Topic detector K-NN classify algorithm F1-score

As a result of this analysis, we conclude that the two methods have similar behaviours, but using *K-NN classify* algorithm, we have better results.

To sum up, the experimental study has shown how to optimize the system, specifically, context finder and topic detector modules. On the other hand, we have seen that the polarity analyzer performs well when used in conjunction with our context finder, because the F1-Score is very close to the F1-Score measured with the test windows. Besides, this experimental study could be the beginning of a deeper improvement of the system in the future.

CHAPTER 5

Case Study

This Chapter describes the case study oriented to aspect based sentiment analysis on the domain of restaurant reviews. We present the process followed for analyzing data, the results obtained, the visualization implementation and how does the final system work.

5.1 Analyzing data

Our project is focused in restaurant reviews. This chapter explains the process executed to analyze restaurant reviews. The data with restaurant reviews used in this project is SemEval 2015 dataset. The dataset is part of the Task 12: Aspect Based Sentiment Analysis from SemEval 2015 [13]. The evaluation dataset described in section 4.1 is the same used to set up the final system, we can see an example of this in 4.1. Moreover, we have increased this dataset with the test data from SemEval 2015, only useful once we have evaluated the system. Finally, the dataset used has around 350 restaurant reviews for the restaurant domain.

We have analyzed each review from this dataset with our aspect-based sentiment analyzer deployed as a Senpy plug-in, as explained in 3.6. With these reviews, we obtained 1568 aspects. The most common words that define these aspects are "food", "service","place","restaurant","staff", "sushi","pizza","atmosphere","decor" and "wine" among others. With regard of the topic and polarity classification done of these aspects, in table 5.1, the results are illustrated.

Aspects	**Positive**	**Negative**	**Total**
Ambiance	126	58	184
Drinks	62	11	73
Food	584	247	831
Location	9	3	12
Restaurant	124	85	209
Service	168	91	259
Total	1073	495	1568

Table 5.1: Restaurants reviews aspects classification

All this data, provided by our Senpy plug-in in JSON-LD format as shown in 3.2 is collected in a JSON array to be easily loaded by the visualization system.

5.2 Displaying data

Once the data has been analyzed, the visualization module displays data using a dashboard based on tabs and widgets. The main tab showed when a user opens the site is the reviews tab. The main purpose of this tab is to show the aspect-based sentiment analysis realized. In figure 5.1 we can see the reviews tab screen.

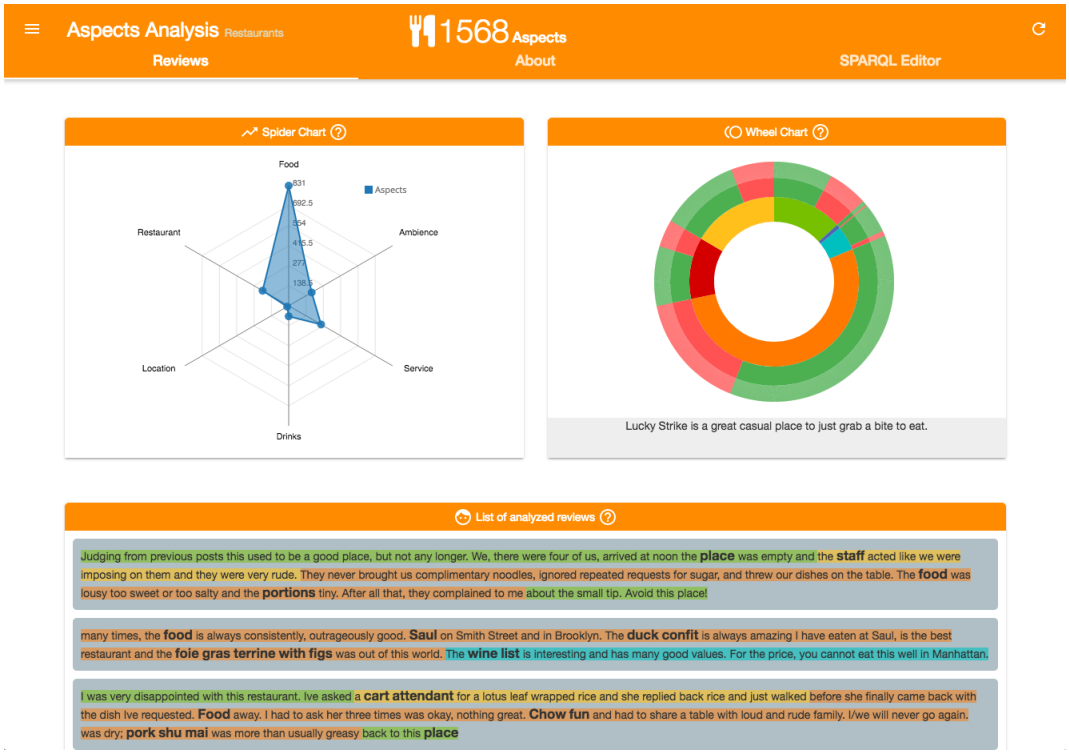

Figure 5.1: Reviews dashboard

The ***topic radar*** chart has been created to show the number of aspects that belongs to each topic. The data from topic classification are put here. As a result, we can see that food is the most common topic as it was expected. Near this chart, we can see the ***sentiment wheel*** chart. This chart shows the polarity of each aspect analyzed. This chart allows finding aspect classified by his topic and his polarity and shows the aspect context below.

Finally, we have the ***list of analyzed reviews*** chart in the bottom. This chart contains all the reviews analyzed. It shows the aspects texts colored according to its topic and the extracted aspect target in bold.

5.3 Conclusions

After this chapter, we can conclude that the system works properly. Besides, we have extracted interesting information from the reviews analyzed. People usually criticize more the restaurant in general if they write a review of it. Drinks are the topic more positive with 62 aspects positive of 73. Also, we can observe that the service is the topic most commented after food, 35% of the times, negatively.

CHAPTER 6

Conclusions and future work

This final chapter presents the conclusions of this thesis, as well as the achieved goals and the future work that can be initialized from this work.

6.1 Conclusions

In this project we have developed an aspect-based sentiment analyzer for restaurant reviews using different technologies, most of them, developed at Intelligent Systems Group. Also, this analyzer has been deployed as a Senpy plug-in and we have designed a dashboard based on Web Components and D3.js to make analyzed data accessible and interactive.

This project has a modular architecture composed by five modules. The aspect-based sentiment analyzer consists of four of these five modules. An opinion target extractor, a context finder, a topic detector and a polarity analyzer. The fifth module of the prototype is the visualization system.

Added to this, this thesis evaluates the analyzer developed and optimize his performance, giving positive results with regard of the extraction and the classification of aspects and its context.

6.2 Achieved goals

In the following section we will explain the achieved goals that are available in this project.

Learning intelligent systems technologies and data processing. The first step in this project has been learning natural language proccesing tasks, with a special emphasis on sentiment analysis, dependency parsing, linked data and the concept of semantic web and vocabularies, among other technlogies.

Design a system architecture of an aspect-based sentiment analyzer. This was the main goal of the project, We designed a sentiment analyzer, implemented all the sub-modules, including a visualization system.

Develop and evaluate a prototype. We have developed a prototype for restaurant reviews and evaluated with a dataset giving satisfactory results.

Validate the prototype in a case study. The prototype has been validated in a case study for restaurants reviews analyzing more than 1500 aspects from a public and available dataset of reviews.

6.3 Future work

This section details the possible new features or improvements that could be added to the project.

Design another topic detector. We want to explore the possibility of implementing an alternative topic detector, by means of the Latent Semantic Analysis (LSA) algorithm. Also, we could refine the classifying task using not only the aspect, but also its context, implementing a new feature in Sematch.

Detecting subtopics. We could get down a category level to include the classification of subtopics like food-prices, food-quality or drinks-style_options. To do this, we had to improve or topic detector module.

Expand the sentiment range. In order to increase the sentiment analysis spectrum by adding the neutral polarity to the analysis, several modifications can be made on the sentiment submodule.

Adding more domains. This system could be a starting point to develop a better system for different domains, such as sports, or even mixed domains. To do this, we need to train new models for some modules of the system.

Improving visualization system. Include a search and indexing system as Elasticsearch and add SPARQL support. To achieve the last one, an integration of the aspects' concepts with Dbpedia entities can be made.

Bibliography

[1] J. B. Rodrigo Agerri and G. Rigau, "Ixa pipeline: Efficient and ready to use multilingual nlp tools," *Proceedings of the 9th Language Resources and Evaluation Conference (LREC2014),Reykjavik,Iceland*, 2014.

[2] A. Fokkens, A. Soroa, Z. Beloki, N. Ockeloen, G. Rigau, W. R. van Hage, and P. Vossen, "Naf and gaf: Linking linguistic annotations," in *Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, 2014, pp. 9–16.

[3] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Comput. Linguist.*, vol. 18, no. 4, pp. 467–479, Dec. 1992. [Online]. Available: http://dl.acm.org/citation.cfm?id=176313.176316

[4] J. W. H. Alan L. Ritter and P. A. Nelson, "Distributional word clustering in parallel," 2003. [Online]. Available: https://aritter.github.io/cluster.pdf

[5] G. C. Tomas Mikolov, Kai Chen and J. Dean, "Efficient estimation of word representations in vector spac," 2013. [Online]. Available: http://arxiv.org/pdf/1301.3781v3.pdf

[6] P. B. Subhabrata Mukherjee, "Feature specific sentiment analysis for product reviews," 2012.

[7] O. Araque, I. Corcuera-Platas, C. Román-Gómez, C. A. Iglesias, and J. F. Sánchez-Rada, "Aspect based Sentiment Analysis of Spanish Tweets," in *Proceedings of TASS 2015: Workshop on Sentiment Analysis at SEPLN co-located with 31st SEPLN Conference*, vol. 1397, September 2015, pp. 29–34. [Online]. Available: http://ceur-ws.org/Vol-1397/gsi.pdf

[8] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

[9] G. Zhu and C. A. Iglesias, "Sematch: Semantic Entity Search from Knowledge Graph," in *SumPre 2015 - 1st International Workshop on Summarizing and Presenting Entities and Ontologies Co-located with the 12th Extended Semantic Web Conference*, June 2015. [Online]. Available: http://km.aifb.kit.edu/ws/sumpre2015/paper4.pdf

[10] G. A. Miller, "WordNet: A lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, November 1995.

[11] E. Conde-Sánchez, "Development of a Social Media Monitoring System based on Elasticsearch and Web Components Technologies," Master's thesis, ETSI Telecomunicación, June 2016.

[12] A. Pascual-Saavedra, "Development of a Dashboard for Sentiment Analysis of Football in Twitter based on Web Components and D3.js," Master's thesis, June 2016.

[13] "Semeval-2015 task 12: Aspect based sentiment analysis," http://alt.qcri.org/semeval2015/task12/, accessed March 26, 2015.

[14] O. Araque, "Prototype of a Sentiment Analysis System Based on Ensemble Algorithms for Combining Deep and Surface Machine Learning Techniques," Master's thesis, June 2016.

[15] I. Corcuera-Platas, "Development of an Emotion Analysis System in a University community," ETSI Telecomunicación, July 2015. [Online]. Available: http://gsi.dit.upm.es/administrator/components/com_jresearch/files/publications/nachoTFG.pdf

[16] J. Overson and J. Strimpel, *Developing Web Components: UI from jQuery to Polymer*. O'Reilly Media.