UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN


GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

# Design and Development of a Stylometry Library for Texts in Spanish and English. Application to Terrorist and Radical Texts

ÁLVARO DE PABLO MARSAL

2019

## TRABAJO DE FIN DE GRADO

| | |
|---|---|
| **Título:** | Diseño y Desarrollo de una Biblioteca de Estilometría para Textos en Español e Inglés. Aplicación a Textos Terroristas y Radicales |
| **Título (inglés):** | Design and Development of a Stylometry Library for Texts in Spanish and English. Application to Terrorist and Radical Texts |
| **Autor:** | Álvaro de Pablo Marsal |
| **Tutor:** | Carlos Ángel Iglesias Fernández |
| **Departamento:** | Departamento de Ingeniería de Sistemas Telemáticos |

## MIEMBROS DEL TRIBUNAL CALIFICADOR

| | |
|---|---|
| **Presidente:** | — |
| **Vocal:** | — |
| **Secretario:** | — |
| **Suplente:** | — |

## 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 DE FIN DE GRADO

# Design and Development of a Stylometry Library for Texts in Spanish and English. Application to Terrorist and Radical Texts

**Álvaro de Pablo Marsal**

Junio 2019

# Resumen

Cada una de las personas del mundo escribe con un estilo distinto a cualquier otra persona. Este estilo, además de estar intrínisicamente relacionado con la misma persona que lo escribe, también tiene que ver con el ámbito o el propósito para el que se escribe. El campo que estudia este estilo es la Estilometría.

La Estilometría se basa en el estudio del estilo a través de diferentes métricas. Entre otras, destacan el Índice de Legibilidad, la Riqueza de Vocabulario, la Formalidad y la Coherencia. A su vez, estas métricas se pueden medir e interpretar de distintas formas.

En este proyecto, se ha diseñado y desarrollado en Python una biblioteca de Estilometría capaz de medir el estilo de un texto utilizando diferentes algoritmos basándose en las métricas mencionadas anteriormente. Esta biblioteca permite analizar el estilo tanto de textos escritos en español como en inglés, siendo algunas de las métricas diferentes para cada uno de los idiomas debido a las características de cada uno de ellos (longitud de las palabras, longitud de las frases...).

Posteriormente, para la visualización de los datos se ha desarrollado un Dashboard basado en web components donde poder seleccionar un texto y poder ver de una forma clara y cómoda cómo es el estilo de dicho texto.

Esta herramienta podría usarse con diferentes objetivos: comprobar que un texto tiene un estilo que se ajusta a las características del público que va a leerlo, comparar el estilo de dos individuos (políticos, escritores, personalidades influyentes...), conocer el origen de un texto así como identificar su autoría y muchos más.

En concreto, en este proyecto se ha enfocado el uso de la biblioteca en la comparación de noticias que hablan de terrorismo con manifiestos y comunicados de grupos terroristas como ETA o el ISIS. Así, si fuera posible, podrían identificarse con anterioridad a su publicación o consumo por parte de usuarios de Internet y redes sociales.

En definitiva, esta biblioteca de Estilometría puede ser utilizada con diferentes propósitos basados en el análisis del estilo de los textos.

**Palabras clave:** Estilo, Estilometría, Texto, Índice de Legibilidad, Riqueza de Vocabulario, Formalidad, Coherencia, Dashboard, Python, Terrorismo, PLN, Métrica

# Abstract

Each person writes in a different style to any other person. This style, in addition to being intrinsically related to the same person who writes it, is also related to the scope or purpose for which it is written. The field that studies this style is called Stylometry.

Stylometry is based on the study of the style through different metrics. Among others, highlight the Readability Index, Vocabulary Richness, Formality and Coherence. At the same time, these metrics can be measured and interpreted in different ways.

In this project, a Python Stylometry library able to measure the style of a text has been designed and developed using different algorithms based on the previous metrics. This library allows us to analyze the style of texts written in Spanish and English, depending some of those metrics on the language due to the characteristics of each one of them (length of words, length of sentences ...).

Later, for the visualization of the data, a Dashboard based on web components where you can select a text and be able to see in a clear and comfortable way how is the style of that text has been developed.

This library could be used to achieve different objectives: to check that a text has a style adapted to the characteristics of the audience that is going to read it, to compare the style of two different people (politicians, writers, influential people...), to know the source of a text as well as to identify its authorship and more.

In particular, this project has been focused on the use of the library for the comparison between news that talk about terrorism and statements made by terrorist groups like ETA or ISIS. Thus, if this would be possible, these radical texts could be identified and removed before its publication or consumption by Internet users and social networks users.

Definitely, this Stylometry library can be used for different purposes based on the analysis of the style of the texts.

**Keywords:** Style, Stylometry, Text, Readability Index, Vocabulary Richness, Formality, Coherence, Dashboard, Python, Terrorism, NLP, Metric

# Agradecimientos

Quiero agradecer a todos los que me han apoyado en la realización de este proyecto. Especialmente a mi tutor Carlos Ángel Iglesias y a la gente del GSI por su ayuda a la hora de sacar el trabajo adelante.

Muchas gracias a todos.

# Contents

# List of Figures

# List of Tables

CHAPTER 1

# Introduction

## 1.1 Context

Throughout the history of humanity, and, particularly, in the wake of the birth of Writing around 3000 B.C., different writing styles of each author have been studied. Already in the Old Testament, one of the first references in history to the ability of distinguish one style from another is made [1].

At the same time, and always due to the style each author intrinsically has and uses in his works, has been achieved on numerous occasions to attribute to different authors works that initially were anonymous (or works with a questionable authenticity) based mainly on that author's style study. The first historic reference that we actually have about the use of this kind of studies dates back to 1439, when Lorenzo Valla analyzed the *Donation of Constatine* [12] and he deducted that it was a forgery. Another moment where this analysis has gained special relevance was the verification that 12 of the so-called *Federalist Papers* were written by John Madison, when initially these papers were published anonymously. The science that studies and determines results about the style is called **Stylometry**.

According to the Oxford dictionary, **Stylometry** is "*the statistical analysis of variations in literary style between one writer or genre and another*" [2]. This definition could be extended to other fields beyond writing, such as painting, music or even speech. **Stylometry** can also be used to avoid plagiarism or even to analyze different personality traits of the

works of the authors. In any case, this text will be focused on analyzing writing styles.

**Stylometry** is based on statistical analysis of texts and on the analysis of different kinds of metrics present in texts, like verbal forms, disused words, vocabulary, ways of using punctuation signs, richness of language, the frequency of use of different words... The foundations of modern **Stylometry** were defined by the philosopher Wicenty Lutoslasky in his book *Principes de stylométrie* [22]. In fact, the word "**Stylometry**" owes its existence to this author.

Whereas long ago stylometric analysis was performed manually, thanks to the huge advance of the new technologies that analysis has been made much more efficient. The rise of technologies based on Machine Learning and Artificial Intelligence since the late twentieth century, has managed to automate the study obtaining very high levels of accuracy. Furthermore, the existence of large databases favors the viability of stylometric studies, expanding the range of possibilities.

The widespread use of social networks such as Twitter makes the **Stylometry** the perfect science for the study of the different variables attributable to the users of these applications. In fact, there are several research articles that, using different psychological indicators, such as MBTI, manage to extract personality traits from tweets [44]. Following with the use of the Stylometry in areas related to the Internet, it can be used for compare the readability of different blogs, web pages, know about the subject of articles posted on the network, know the cultural richness of these, obtain information about the author, both MBTI personality traits as possible hobbies, country of belonging, political preferences ...

Of course, **Stylometry** techniques can be used for e-mail services, with applications such as knowing what is spam and what is not. Also, it can be used to fight against cybercrime, for example, to know who is committing a crime for their style when writing.

In fact, we can say that among the possible modern uses of the **Stylometry**, we can find the aforementioned email study and cybercrime study, but also it can be used in the analysis of music lyrics, music melody, paintings, literary works, forensic linguistic, plagiarism, social networking and instant messaging, among others [35].

## 1.2 Project goals

The main objective of this project is to design and to develop a **Python Stylometry Library**, based on many metrics that can analyze the style of a text.

One of the particularities of this library will be that it is going to be able to measure the style both in English and Spanish texts. For this reason, when we were developing it, we must be careful with this topic because not all metrics works equally well with English texts as with Spanish texts.

Once the library is ready, the next task is to have the possibility of observing this metrics in a beautiful and comfortable scenario that we will call ***Dashboard***. In this **Dashboard**, we will be able to select a text that we want to analyze, we will see the style metrics of that text and we will be able to compare the style of that text with another text.

In particular, in this final work we are interested in the analysis of radical texts, published by terrorists groups like ISIS, Al Qaeda, IRA, etc.

The results of the comparison between styles could be useful to know a priori if a text is a terrorist text or is a text talking about terrorism (a new, a book, etc.).

## 1.3 Structure of this document

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

***Chapter 1: Introduction*:** This chapter introduces the project and makes a brief introduction about what is the meaning of the word **Stylometry**.

***Chapter 2: Enabling Technologies*:** This chapter presents the most important technologies used in the project.

***Chapter 3: Development of the Library*:** This is the chapter where is told how works the library, how it has been developed and which are the metrics included in the library.

***Chapter 4: Architecture*:** This chapter explains how the visualization system was made and explains the architecture of the project.

***Chapter 5: Case of Use*:** A case of use of the library is presented in this chapter: the comparison between a text that talks about terrorism with a terrorist statement.

***Chapter 6: Conclusions*:** In this chapter the conclusions of the project are presented, the achieved goals, the problems found and the future work over this project.

CHAPTER 2

# Enabling Technologies

## 2.1 Introduction

This chapter introduces all the technologies used on this project. Firstly, it is important to introduce the concept of Natural Language Processing (**NLP**). **NLP** is a field of computer science where the connection between the computer and human language is studied. This science is based on the analysis of the human natural language by computer techniques.

The history of **NLP** begins around 1950, when Alan Turing published his book *Computing machinery and intelligence* [41], where what we know as the Turing Test is described, which is a test where the intelligence of a machine is measured based on its ability to maintain conversations in natural language with a human being. Beginning in the 80s, a revolution in the field of **NLP** began with the arrival of algorithms based on machine learning and Artificial Intelligence.

The processing of written texts by a machine is not always a simple process. First of all, the text that will be analyzed is preprocessed for its correct analysis. This process is called *tokenization*, and it includes the separation of the text in words (tokens) and the processing of them, avoiding possible undesirable characters like punctuation signs, contractions, exclamation and interrogation marks, etc. This is the most important step in **NLP**, since, depending on how those words are obtained, it will be easier or harder the analysis of the texts and the way to work with them. Furthermore, it is important to note

that in many occasions there are errors of transcription in the texts and it is necessary to know how to work with it.

Afterwards, a variety of functions can be applied to obtain information about the text, as Part of Speech tagging, the obtaining of the different words that conform the text, knowing the common and uncommon words within it, etc.

For the processing of texts using **NLP** techniques, the project has been developed mainly in Python, using extensively the **NLTK** library, as well as some others, such as Matplotlib, Scipy and others that will be told next.

## 2.2 Natural Language Processing

Some Python libraries have been used in this project for processing texts and extract a lot of features of the same.

### 2.2.1 NLTK

**NLTK** (Natural Language Toolkit) [21] is an interface and a set of libraries for working in Python with **NLP** techniques. Among others, it has more than 50 corpus to work with them, a lot of lexical resources and it is one of the best tools for working in the **NLP** field.

It provides a big variety of preprocessing and processing functions, of which the most important are classification, tokenization, stemming, tagging, parsing, and semantic reasoning.

Now, we are going to present the use that we have made in this project of **NLTK**:

- **NLTK Corpora**: **NLTK** has a large collection of corpus. They are all very different from each other and can be useful for processing texts, since they serve us as dictionaries of different languages, classifiers by genre, synonyms dictionaries, etc. Some examples of these corpus are the Brown Corpus [20], which contains 1.15M of words classified by genre or the Reuters Corpus [32], with 1.3M of words.

- **Tokenization**: is the process of transform a text into separated words or sentences called tokens. This is one of the most important steps for working with texts. Firstly, the text is filtered for avoiding undesirable characters, expanding contractions, etc. Then, the tokenizer function splits the whole text by marking blank spaces or phrase endings as the reference point for this division. Finally, all the words are processed again for deleting possible mistakes in the transcription of the words.

- **Pos-Tagging**: is the process of assigning different tags related to Part of Speech to each word in the text. Thus, we can know which of the words are adjectives,

verbs, proper nouns, common nouns, conjunctions, adverbs, etc. The process of Pos-Tagging was carried out using the **NLTK** recommended Pos-Tagger and the Stanford Pos-Tagger [23], which returns better results for both English and Spanish texts. The Stanford Pos-Tagger is a Java implementation of a log-linear Part of Speech tagger, available for several languages, among which include Spanish and English languages.

- **Stemming and Lemmatization**: these are the processes that are responsible for reducing the inflectional form of a word into a common base or root. However, these processes work in a different way, as we show in the next pictures [31]:

| Form | Morphological information | Lemma |
|---|---|---|
| studies | Third person, singular number, present tense of the verb study | study |
| studying | Gerund of the verb study | study |
| niñas | Feminine gender, plural number of the noun niño | niño |
| niñez | Singular number of the noun niñez | niñez |

Figure 2.1: Lemmatization Process

| Form | Suffix | Stem |
|---|---|---|
| studies | -es | studi |
| studying | -ing | study |
| niñas | -as | niñ |
| niñez | -ez | niñ |

Figure 2.2: Stemming Process

- **Removing Stop Words**: Stop Words are those words that do not add meaning to the text. It could be necessary to remove them when we are analyzing a text. **NLTK** provides a corpus of Stop Words both in Spanish and English language for knowing which are these words and facilitate its removing.

- **FreqDist**: the component FreqDist of **NLTK** provides us some functions for searching which are the most common words within a text, among others.

### 2.2.2 Gensim

Gensim [30] is an open-source Python library created by Radim Řehůřek for unsupervised topic modelling and Natural Language Processing, using modern statistical machine-learning. It includes some implementations of Word2Vec algorithms, Latent Semantic Analysis (LSA, LSI), Latent Dirichlet Allocation (LDA), TF-IDF and more. The most important point of this library in this project is that it allows us to work with Word Embeddings models and has a lot of functions that makes the work with them very easy.

### 2.2.3 GSITK

GSITK (GSI Toolkit) [4] is a library on top of scikit-learn that eases the development process on **NLP** machine learning driven projects. It uses numpy, pandas and related libraries to easy the development. The main features of the GSITK library are the Word2Vec Features, that implements a generic word vector model, previously loaded a Word Embeddings model

that has to be compatible with Gensim. It allows, among others, to transform a text into a numeric vector to work with it.

### 2.2.4 Word2Vec

Word2Vec (Word to Vector) is a set of models that are used to produce Word Embeddings. The input of Word2Vec models would be a large corpus of text, and the output will be a vector space typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in that vector space.

In this project, they have been used some models of Word Embeddings for measuring some metrics. In particular, for the Spanish language it has been used a pre-trained model [39] with 1,000,653 word embeddings of dimension 300 trained on the Spanish Billion Words Corpus.

For the English language, they have been used two pre-trained Word Embeddings models. The first of them is the "Google News Vectors Negative 300 Word Embeddings" model [38]. It includes word vectors of dimension 300 for a vocabulary of 3 million words and phrases that they trained on roughly 100 billion words from a Google News dataset.

The second model [27] used for the English language is the "Wikipedia News Model" with more than 1 million word vectors of dimension 300 trained on the 2017 Wikipedia entries. It seems that this model is better than the Google News model. Even so, in the development of this project the two models have been used the same.

### 2.2.5 LangDetect

LangDetect [8] is a Python module to detect in which language a text is written. It is based on the Google's Java library "language-detection". It supports 55 different languages, including Spanish and English. Although it allows many languages, it is possible to add others.

## 2.3 Dashboard

The **Dashboard** is the graphical interface in which the results of this project will be shown. Data is captured using **GSI Crawler**, analyzed with **Senpy**, stored in **ElasticSearch** and finally shown in the **Dashboard**. All this process is made with the **Luigi** orchestrator.

### 2.3.1 Senpy

**Senpy** [33] is a framework developed by the **GSI group** for sentiment and emotion analysis services. One of the main reasons why the use of **Senpy** is easy and comfortable is because

the services offered share a common semantic API and vocabularies based on ontologies.

**Senpy** is based on the analysis of a piece of text, analyzing it with a plugin or several plugins and finally returning the results of this analysis. The following figure shows how **Senpy** works:

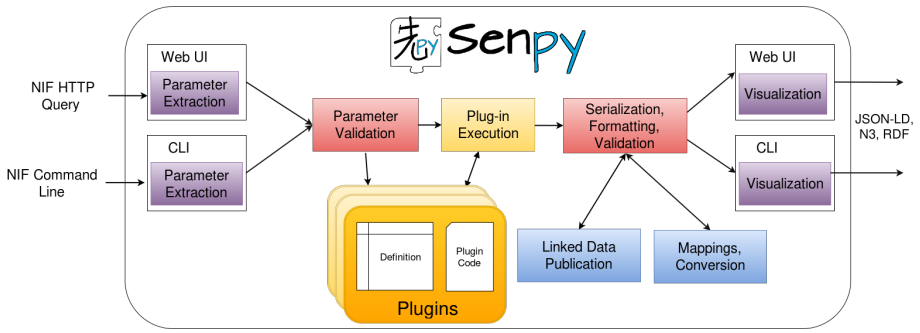



Figure 2.3: Senpy architecture [33]

### 2.3.2 ElasticSearch

**ElasticSearch** [13] is a search engine based on the Lucene library and tries to make all its features available through the JSON and Java API. It is developed in Java and official **ElasticSearch** clients are available in many different languages, such as Java, Ruby or Python. It is published as open source under the conditions of the Apache license.

**ElasticSearch** uses Query DSL for making queries to the indexed documents. This tool makes the queries in **ElasticSearch** can be done in a very flexible way, adding the fact that this is used through a JSON interface, the queries are simple and the process of debugging too.

One of the main characteristic of **ElasticSearch** is that it only supports the JSON format as a response, so formats like CSV or XML are not supported. Even so, the JSON format is widely supported by numerous programming languages, so it makes **Elastic-Search** the perfect tool to use, for example, in Big Data analysis.

**ElasticSearch** JSON data is sorted by *indexes*, where each of these *indexes* will be where the result of a kind of analysis or another is stored. Each analysis will be stored in *documents* inside its corresponding *index*.

### 2.3.3 Luigi

**Luigi** is a Python module created by Spotify engineers that helps to build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization, etc.

It is based on *tasks*, being each task an unit of work. These tasks and the dependence between them is defined in a pipeline, where it is said to the orchestrator the execution

order of these tasks and the input needed in each task, being these inputs the output of another task.

**Luigi** is a very good solution for running a lot of long or heavy processes who depend on each other. Also is a good tool to have a global vision on error control of each of the parts that make up the execution of the pipeline.

### 2.3.4 GSI Crawler

**GSI Crawler** is a framework developed by the **GSI group** which aims to extract information from web pages enriching following semantic approaches.

In this project, **GSI Crawler** tool has been used for scraping the web in search of news related with terrorism. In particular, news have been scraped from the New York Times, the CNN and Al Jazeera.

As **GSI Crawler** is the basis of the visualization of this project, in *Chapter 4: Architecture* will be explained how this tool has been used.

### 2.3.5 Graphic Interface

The graphic interface of the project, where results will be showed, it is based on a web interface using web components made with the Polymer JavaScript library. This library provides functions to make stylish components for showing data.

Data showed in these web components comes from our **ElasticSearch** index after being analyzed. Some examples of these web components are Google Chart or Number Charts, all of them modified to accept entries from **ElasticSearch**. Also the data can travel from component to component through a Polymer functionality called *bending*. This makes the **Dashboard** a versatile, comfortable and fully functional graphic interface.

## 2.4 Another Libraries

Another libraries have been used for developing the Stylometry library. Specifically, they have been used Scipy (a Python open-source library that provides different tools and mathematics algorithms focused on mathematics, science or engineering fields), WordFreq [37] (a library for looking up the frequencies of words in many languages) or Matplotlib (a powerful Python library for plotting graphs).

CHAPTER 3

# Development of the Library

## 3.1 Introduction

The main goal of this project is the design and development of a stylometry library. For measuring the **style**, it is important to determine which measures are going to be useful to reach that goal. The main measures of the library are **Readability Index**, **Vocabulary Richness**, **Formality Measure** and **Coherence Measure**. At the same time, these indexes can be measured in different ways.

The first step for measuring the **style** of a text is to extract features of it. With this, we can extract information from the text which we will then use to measure the **style**, as well as to have a global vision of how the text is designed or what it is composed of.

The library has been designed as a set of modules to carry out these measures and they will be explained one by one.

## 3.2 Text Statistics

The library provides many functions for having a global vision of the text and to know its main features for measuring the **style**. The first step to have these features is to ***tokenize*** the text.

As explained above, ***tokenization*** is the process of transform a text into separated

words (***tokens***). Firstly, it is important to say that all the functions of this library has a header with two arguments: a text and the language of the text. The language of the text is very important because some functions work in different ways depending on the language.

The first step for ***tokenize*** a text is to clean it. To do this, all the special characters that do not add any value to the text (dashes, bars, etc.) are removed. Then, characters like quotes or apostrophes that can be written in different ways, are replaced by a single sign so that there is no mistake when analyzing the text (e.g. signs like ‘ or ´ are replaced by ’).

If the text is written in English, the function uses a library called *contractions* [43], which has a method called *fix* that expands all the contractions in a text (e.g. “*It's John*” becomes “*It is John*”). By the other hand, if the text is written in Spanish, the function removes Spanish punctuation signs like ¿ or ¡.

Once the text is fairly clean, the text is ***tokenized***. This step is carried out with the **NLTK** library function *word_tokenize*, which is a function that splits a text into words, assuming that the separation between words is a space or a punctuation mark. Once we have the words of the text, punctuation marks like exclamations, commas or dots are removed. The last step is a final cleaning of each word in the text.

Thus, we have each of the words that form the text clean and separate to make a study of it and, first, obtain a series of text statistics that allow us to measure the **style**.

**English tokenization** example:

*“He's dead. -said Peter- Where were you when he was murder?” –> ‘he’, ‘is’, ‘dead’, ‘said’, ‘Peter’, ‘Where’, ‘were’, ‘you’, ‘when’, ‘he’, ‘was’, ‘murder’.*

**Spanish tokenization** example:

*“Está muerto. -dijo Peter- ¿Dónde estabas tú cuando fue asesinado?” –> ‘Está’, ‘muerto’, ‘dijo’, ‘Peter’, ‘Dónde’, ‘estabas’, ‘tú’, ‘cuando’, ‘fue’, ‘asesinado’.*

We can also ***tokenize*** the text into sentences and count how many sentences has a text using **NLTK** library and specifically the Punkt tokenizer [45], that “divides a text into a list of sentences by using an unsupervised algorithm to build a model for abbreviation words, collocations, and words that start sentences.”. In this library, a pre-trained Punkt Tokenizer for English language and another for Spanish language are used. The function dedicated to this process, after using the Punkt tokenizer, checks also the existence of paragraphs, which are to be counted as new sentences.

The rest of the functions present in the **Text Statistics** module will be counted below:

- **Characters per Sentence and Characters per Word**: These are two simple functions, that calculates an average of the number of characters per sentence and the number of characters per word. Even though they are the simplest functions in the

library, these are one of the most important functions because they are going to be very useful when we will want to measure the **Readability Index** of a text.

- **Words per Sentence**: This function, similar to the previous two, calculates an average of the words per sentence in the text. It is going to be useful to measure the **Readability Index** too.

- **Different Words**: This function identifies all the different words within the text (when we talk about **Vocabulary Richness** we will call them "***types***"). The algorithm sorts the words alphabetically and then compares a word with the next word in the list. If they are different, it means that they are different words. If not, it keeps checking.

- **Word Classes**: This function makes a Part of Speech (POS) analysis of the text, classifying the words in conjunctions, adjectives, verbs, nouns, adverbs, prepositions, determiners, pronouns and interjections. Depending on the language of the text, it uses a classifier or another. For texts written in English, the function uses the **NLTK** recommended POS-Tagger. By the other hand, for texts written in Spanish it uses the Stanford POS-Tagger [23].

- **Short Words**: This function counts the number of short words within the text. It is a simple function that calculates how many words have less or equal than three characters. In English, with less than three characters we can find words like "*a*", "*the*", "*sad*" or "*put*". In Spanish we can find words like "*él*", "*soy*", "*por*" or "*dos*".

- **Frequency of Words**: This is an auxiliary function that is responsible for calculating how many times each word appears in the text. It is done using the FreqDist module of the **NLTK** library that allows to calculate the most common words in a text and the repetition frequency of each word within it.

- **Removing StopWords**: Another auxiliary function. Using the StopWords Corpus, if a word appears in the corpus, it means that it is a StopWord and we should remove it from the tokens list. If the word does not appear in the corpus, it is not a StopWord and we can suppose that this word adds meaning to the text.

- **Common and Uncommon Words**: This function checks if a word is a common word or an uncommon word in its language. To do this, the implementation of this function has been changing throughout the development of the project.

  The first implementation of this function was based on the comparison of each word with a dictionary with a lot of words of the language of that word. First of all, the

function obtained the different words of the text as explained before. Then, each word goes through a process of lemmatization and, if the lemmatized word appears in the dictionary, it means that this word is a common one. If it does not appear, it means that it is an uncommon word.

The second implementation uses the Gensim library. The first step for using this library is to load a Word Embeddings model, one for English and another for Spanish (the models used in this project are defined and explained in Chapter 2: *Enabling Technologies*).

The vocabulary of each model is in the field *vocab* of the model. The algorithm of this implementation works with the next norm: if a word appears in the vocabulary of the model, the function considers that this is a common word. If not, it will be an uncommon word.

The third implementation is based on an external library called WordFreq [37]. This library has a function called *zipf_frequency*, that returns the frequency of occurrence of a word in its language. It is based on the Zipf Law [28], that is an empiric law that determines the frequency of appearance of a word in its language. This frequency of appearance follows a distribution that can be approximate by:

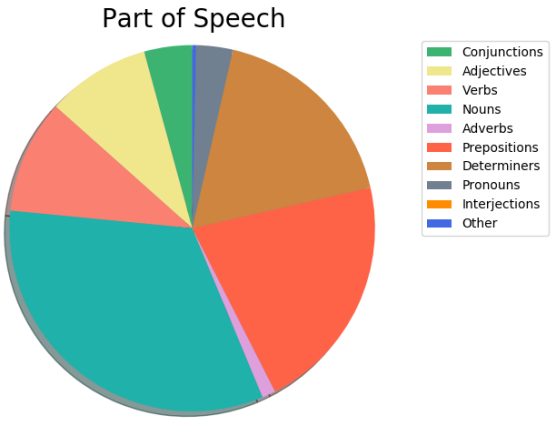
$$P_n \sim \frac{1}{n^a} \tag{3.1}$$

$P_n$ represents the frequency of the umpteenth most frequent word and the exponent $a$ is a positive real number near to 1.

Therefore, the *zipf_frequency* function returns a number between 0 and 10 (it is a logarithm scale). The larger the result, the more common the word is in its language. If the result of the *zipf_frequency* function is bigger than 2, the algorithm considers that the word is a common word. If not, the word will be labeled as an uncommon word.

The third implementation is the better algorithm for implementing this function. For this reason, when common and uncommon words will be needed for something in this text, it will be used the third implementation of this function.

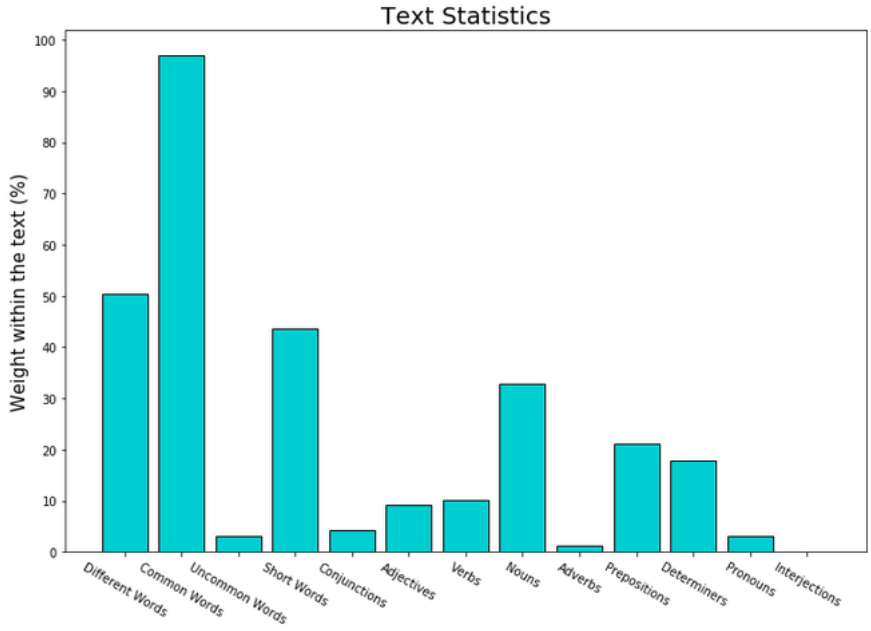
Once explained the operation of the Text Statistics module, we can see an example of how they are exactly the features of a text, applied in this case to a text written in Spanish from the Spanish newspaper El País [40]:




(a) POS Analysis

| Features | Value | Weight within the text (%) |
|---|---|---|
| Total words | 588 | |
| Total sentences | 17 | |
| Words per sentence | 34.59 | |
| Characters per sentence | 177.12 | |
| Characters per word | 5.12 | |
| Different words | 297 | 50.51% |
| Common Words | 288 | 96.96% |
| Uncommon Words | 9 | 3.03% |
| Short Words | 257 | 43.70% |
| Adjectives | 54 | 9.18% |
| Adverbs | 7 | 1.19% |
| Conjunctions | 25 | 4.25% |
| Determiners | 105 | 17.86% |
| Interjections | 0 | 0.00% |
| Nouns | 193 | 32.82% |
| Prepositions | 124 | 21.09% |
| Pronouns | 19 | 3.23% |
| Verbs | 59 | 10.03% |

(b) Text Statistics Table




(c) Text Statistics Graph

Figure 3.1: Text Statistics Analysis

In the first figure (a), it can be seen that in this newspaper article the nouns (32.82%) predominate over the rest of the words, followed by the prepositions (21.09%), the determiners (17.86%) and the verbs (10.03%). It should be noted that the adverbs (1.19%) do not have a great weight within the text and there are no interjections.

On the other hand, the text has 297 different words (50.51%) from the 588 that make up the text. Within those 50.51% of different words, 288 (48.98%) are common words and 9 (1.53%) are uncommon words. This latest information has been carried out using the third implementation of the Common and Uncommon Words algorithm. The uncommon words of

this text would be: "*Brexit*", "*DUP*", "*Féin*", "*Irlandas*", "*Londonderry*", "*norirlandesa*", "*norirlandés*", "*parapetándose*" and "*unionismo*".

It also highlights that 257 (43.71%) words are short words (less or equal than 3 characters). The high number of determiners and prepositions (together are the 38.95% of the words of the text) could explain this result.

By last, in the figure below (c) it can be seen a summary in the form of a graph with the weight of all these measures within the text.

## 3.3 Style Metrics

Once we have the main features of the text that we want to analyze, it is possible to measure its **style**. The **style** will be defined by four metrics, having them at the same time several submetrics. These metrics and their submetrics are going to be defined bellow.

### 3.3.1 Readability Index

**Readability Index** [19] is a **style** metric that measures how easy or difficult is to read a text. "Readability" should not be confused with "Comprehension" or another similar word, because this metric only measures how much difficult is to read a text, but it does not measure how much difficult is to comprehend it.

For texts written in Spanish, there are two very good indexes: **INFLESZ Readability Index** and $\mu$ **Readability Index**.

For texts written in English, in this text we are going to talk about three indexes: **ARI** (Automated Readability Index), **Fog Count** and **Flesch Readability Index**. Regarding these indexes, there is an article [19] published in 1975 talking about a recalculation of these indexes for using them in the Navy army. In this project, these improvements have been implemented in the library and defined and analyzed in this text.

All of these indexes are based on the length of the sentences or the words of a text or even the number of syllables that a text has. The reason that the **Readability Index** depends on the language in which a text is written is because, in the case that we only want to analyze texts written in Spanish or English, Spanish sentences and words are usually largest than English sentences and words. Also, Spanish words usually has more syllables than English words.

For this reason, English indexes are different to Spanish indexes. Now all these indexes are going to be explained one by one.

#### 3.3.1.1 Spanish Readability Index Metrics

The Spanish Readability Index Metrics implemented in this library are the **INFLESZ Readability Index** and the $\mu$ **Readability Index**.

- **INFLESZ Readability Index** [5]: In 1993, an algorithm for measuring the **Readability Index** in texts written in Spanish was proposed. This formula was a modification of the **Flesch Index** (explained later) and another Spanish **Readability Index** made in 1959 by Fernández-Huerta. This algorithm is known as Flesch-Szigriszt Readability Index.

  Some years later, Inés M$^{a}$ Barrio in her doctoral thesis proposed a new scale for the Flesch-Szigriszt Readability Index known as **INFLESZ**. The formula of this algorithm and an explanatory table are shown bellow:

$$INFLESZ = 206.835 - (62.3 * \frac{syllables}{word}) - \frac{words}{sentence} \tag{3.2}$$

| **INFLESZ Reading Difficulty** | |
|---|---|
| **Score** | **Reading Difficulty** |
| >80 | Very Easy |
| 65-80 | Quite Easy |
| 55-65 | Normal |
| 40-55 | A Bit Difficult |
| 0-40 | Very Difficult |

Table 3.1: INFLESZ Reading Difficulty

- $\mu$ **Readability Index** [29]: In 2006, Miguel Muñoz Baquedano and José Muñoz Urra proposed a new index for measuring **Readability Index** for texts written in Spanish. The particularity of this method is that it uses data that are not usually used in the different existing **Readability Index** algorithms, like the variance of the characters per word. The algorithm is as follows:

$$\mu = \frac{n}{(n-1)} * \frac{\bar{x}}{\sigma^2} \tag{3.3}$$

  where $n$ is the number of words within the text, $\bar{x}$ is the average of the number of characters per word and $\sigma^2$ is its variance. The variance is calculated with the next formula:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} \tag{3.4}$$

where $n$ is the number of words within the text, $x_i$ represents the length of each **token** (each word) and $\bar{x}$ is the average of the number of characters per word.

To end, the score of the $\mu$ **Readability Index** represents the difficulty of reading a text, as the next table shows:

| $\mu$ **Reading Difficulty** | |
|---|---|
| **Score** | **Reading Difficulty** |
| >91 | Very Easy |
| 81-91 | Easy |
| 71-81 | A Bit Easy |
| 61-71 | Suitable |
| 51-61 | A Bit Difficult |
| 31-51 | Difficult |
| 0-31 | Very Difficult |

Table 3.2: $\mu$ Reading Difficulty

As an example, it can be seen in the next table the reading difficulty and the values of the previous algorithms applied to three different texts written in Spanish: the newspaper article of the previous section *Text Statistics*, Chapter 1 to 7 of the book "The Little Prince" and a Spanish scientific article about the Global Warming [6]:

| **Spanish Readability Index Table** | | | |
|---|---|---|---|
| **Test Text** | **Readability Index** | **RI Value** | **Reading Difficulty** |
| Newspaper | INFLESZ | 39.382 | Very Difficult |
| Article | $\mu$ | 52.487 | A Bit Difficult |
| The Little | INFLESZ | 71.889 | Quite Easy |
| Prince Tale | $\mu$ | 61.110 | Suitable |
| Scientific | INFLESZ | 40.657 | A Bit Difficult |
| Publication | $\mu$ | 51.239 | A Bit Difficult |

Table 3.3: Spanish Readability Indexes Test

As it can be seen, results of both algorithms are very similar and, in the case of the example of the scientific publication, both algorithms determine that it is a bit difficult to

read it. In the case of the tale, we can say that its reading difficulty is normal or even easy, and the newspaper article is difficult to read.

#### 3.3.1.2 English Readablity Index Metrics

The English Readability Index Metrics implemented in this library are the **ARI** (Automated Readability Index), **Fog Count** and the **Flesch Readability Index**. Also, there were implemented some improvements [19] of these indexes.

- **ARI** [34]: Automated Readability Index (**ARI**) is a **Readability Index** for English texts. It is one of the first readability indexes to be developed. It was developed by the University of Cincinnati in 1967, emerged for the need of the Air Force of the United States to know how much time it took to read manuals and other documents and the difficulty of reading them.

  **ARI** depends on the characters per word and on the words per sentence. Specifically, the mathematical formula of the **ARI** is the following:

  $$ARI = 4.71 * \frac{characters}{word} + 0.5 * \frac{words}{sentence} - 21.43 \tag{3.5}$$

  where $\frac{characters}{word}$ is the average of the characters per word and $\frac{words}{sentence}$ is the average of the words per sentence.

  This formula corresponds to the original **ARI**. On the other hand, this formula was redesigned as follows:

  $$ARI(Redesigned) = 5.84 * \frac{characters}{word} + 0.37 * \frac{words}{sentence} - 26.01 \tag{3.6}$$

  where $\frac{characters}{word}$ is the average of the characters per word and $\frac{words}{sentence}$ is the average of the words per sentence.

  The result of the previous formulas returns the US Grade Level needed to have the capacity to read fluently a text. The correspondent ages to the US Grade Level and the relation with the **Automated Readability Index** are given in the next table:

| **US Grade Level** | |
|---|---|
| **Ages** | **Grade Level** |
| 5-11 | Primary School |
| 11-14 | Middle School |
| 14-18 | High School |
| >18 | College/University |

| **ARI** | |
|---|---|
| **Score** | **Grade Level** |
| 0-5 | Primary School |
| 5-8 | Middle School |
| 8-12 | High School |
| >12 | College/University |

Table 3.4: US Grade Level and ARI

- **Fog Count** [19]: **The Fog Count Readability Index** is another **Readability Index** to consider and also implemented in this **stylometry** library. The peculiarity of this index is that uses the concept of hard words and simple words. For the algorithm, hard words are those words that have more than 2 syllables, and simple words are those that have 2 or less syllables. Defining the Average Fog Count as

$$Average\ Fog\ Count = \frac{(easy\ words) + 3(hard\ words)}{sentences} \tag{3.7}$$

where *sentences* is the number of sentences in the text, the algorithm of this index is as follows:

$$Fog\ Count = \begin{cases} \dfrac{(easy\ words) + 3(hard\ words)}{2(sentences)} & if \quad Average\ Fog\ Count \geq 20 \\ \dfrac{\dfrac{(easy\ words) + 3(hard\ words)}{sentences}}{2} - 2 & if \quad Average\ Fog\ Count < 20 \end{cases} \tag{3.8}$$

Same as the previous two, this index was redesigned as follows:

$$Fog\ Count(Redesigned) = \frac{\dfrac{(easy\ words) + 3(hard\ words)}{sentences}}{2} - 3 \tag{3.9}$$

As can be seen, the redesigned **Fog Count** is very similar to the traditional Fog Count (if Average Fog Count $< 20$). These indexes give us the US Grade Level needed to have a fluent reading of the text. The relation of the **Fog Count** with the US Grade Level is given bellow:

| **Fog Count** | |
|---|---|
| **Score** | **Grade Level** |
| 0-5 | Primary School |
| 5-8 | Middle School |
| 8-12 | High School |
| >12 | College/University |

Table 3.5: Fog Count Result

- **Flesch Readability Index** [19]: This is the last English **Readability Index** implemented in the library. The **Flesch Readability Index** is given by two different formulas. One of them is the **Flesch Reading Ease Index**, that returns a value

that, with a conversion table, gives the Grade Level needed to read fluently the text. The second one is the **Flesch-Kincaid Grade Level Index**, which returns directly the Grade Level needed. It is important to say that the last one is a redesign of the first one.

Both are based on the average of the words per sentence and on the average of the syllables per word. The formulas are described below:

$$Flesch\ Reading\ Ease\ Index = 206.835 - 1.015 * \frac{words}{sentence} - 84.6 * \frac{syllables}{word} \quad (3.10)$$

$$Flesch - Kincaid\ Grade\ Level = 0.39 * \frac{words}{sentence} + 11.8 * \frac{syllables}{word} - 15.59 \quad (3.11)$$

The interpretation of both formulas is given in the tables bellow:

| **Flesch Reading Ease** | |
|---|---|
| **Score** | **Grade Level** |
| >90 | Primary School |
| 65-90 | Middle School |
| 50-65 | High School |
| 0-50 | College/University |

| **Flesch-Kincaid Grade Level** | |
|---|---|
| **Score** | **Grade Level** |
| 0-5 | Primary School |
| 5-8 | Middle School |
| 8-12 | High School |
| >12 | College/University |

Table 3.6: Flesch Reading Ease and Flesch-Kincaid Grade Level

As it can be appreciated, all the metrics defined in this section have the same result table, with the same assignment of Grade Level for each interval: (0 - 5) to Primary School, (5 - 8) to Middle School, (8 - 12) to High School and (> 12) to College/University. Really, these algorithms return a number that can be approximate by the age of the reader. Nevertheless, another approximation is made in this project, assigning the same Grade Level to a specific results interval. In fact, for knowing the needed age it would have to add 5 to the result of each algorithm (e.g. an **ARI** result of 7.23 would mean that that text is prepared to read with at least 12.23 years old).

As was done before with the indexes in Spanish, the next table is an example table of how these English indexes work. For this example, it is going to be analyzed a kid story ("The Snowman"), and the Chapter 1 of The Theory of the Moral Sentiments by Adam Smith:

| English Readability Index Table | | | |
|---|---|---|---|
| **Test Text** | **Readability Index** | **RI Value** | **Grade Level** |
| Snowman Tale | ARI | 2 | Primary School |
| | ARI (Redesigned) | 1 | Primary School |
| | Flesch Reading Ease | 94.10 | Primary School |
| | Flesch-Kincaid GL | 1.71 | Primary School |
| | Fog Count | 2.33 | Primary School |
| | Fog Count (Redesigned) | 1.83 | Primary School |
| Chapter 1 of The Theory of the Moral Sentiments by Adam Smith | ARI | 15 | College/University |
| | ARI (Redesigned) | 12 | High School |
| | Flesch Reading Ease | 46.65 | College/University |
| | Flesch-Kincaid GL | 13.84 | College/University |
| | Fog Count | 18.66 | College/University |
| | Fog Count (Redesigned) | 17.16 | College/University |

Table 3.7: English Readability Indexes Test

As it can be seen, almost all measures match for each text. In the case of the Snowman tale, all the indexes mark the Grade Level as "Primary School" and all the values are very low. It seems it is a text really easy to read. In the case of the book written by Adam Smith, only the **ARI (Redesigned)** algorithm disagrees with the rest of the algorithms. All of them mark that it is a book appropriate for university students. **ARI (Redesigned)** is the only that marks that it is a book for High School students. Whatever, this index value is 12 and it is on the limit of High School and College/University.

### 3.3.2 Vocabulary Richness

Another interesting metric for measuring **style** is **Vocabulary Richness** [36]. There are several metrics for measuring **Vocabulary Richness**, some of them with interesting characteristics that are going to be shown below.

#### 3.3.2.1 TTR

The **Type Token Ratio (TTR)** [36] is one of the main and easiest metrics for measuring **Vocabulary Richness**. In fact, all the rest metrics defined here depend on this metric. In other words, **TTR** is the basis of all the rest **Vocabulary Richness** metrics.

From now on in this text, the set of words of a text are going to be called ***tokens*** and the set formed by all the different words of that text are going to be called ***types***.

**TTR** is the relation between the **tokens** and the **types** of a text. Mathematically, **TTR** can be represented as:

$$TTR = \frac{types}{tokens} \tag{3.12}$$

As it can be seen, **TTR** is the most intuitive way for measuring the lexical diversity (**Vocabulary Richness**) of a text. There is another easy metric that is strongly related with **TTR** called Mean Word Frequency (MWF) and its value can be interpreted as every how many **tokens** a **type** appears:

$$MWF = \frac{1}{TTR} = \frac{tokens}{types} \tag{3.13}$$

In first view, it could be said that lexical diversity can be easily measure knowing how many different words are used in a text, this is, measuring **Vocabulary Richness** using the **Type Token Ratio**. But it is not always a good idea.

The longer a text is, the more **tokens** make up that text. It means that as a text becomes larger, the **TTR** measure will become smaller (or MWF bigger) because there is not a proportional relationship between the increment of **tokens** and **types**. For this reason, in larger texts we will have a low **TTR** value and probably this value will not correspond with the **Vocabulary Richness** of that texts.

Because of this, they have been emerging arrangements of this measure and another kind of algorithms based on the **TTR** measure like **MSTTR**, **MATTR**, **MTLD** or **HD-D** which are explained below.

#### 3.3.2.2 Improved TTR Metrics

There are some modifications of the **TTR** trying to have a better vision of the **Vocabulary Richness** of a text. The mainly transformations of **TTR** are **Guiraud Index**, **Herdan Index**, **Maas Index** and **Uber Index**.

- **Guiraud Index**

  Pierre Guiraud, in his book Les Caractères Statistiques du Vocabulaire [15], proposed a transformation for the **TTR** based on the square root of the **tokens**. With this, the variation of the index with the text length is the inverse that in the case of the **TTR**. The **Guiraud Index** (called $R$) formula is shown bellow:

$$Guiraud\ Index\ R = \frac{types}{\sqrt{tokens}} \tag{3.14}$$

- **Herdan Index**

**Herdan Index C** [42] is another **Vocabulary Richness** metric. Herdan (1960) proposed a logarithmic formula with the aim that this ratio had a slow variation with the length of the text:

$$Herdan\ Index\ C = \frac{\log(types)}{\log(tokens)} \tag{3.15}$$

- **Maas Index**

  In 1972, Maas [24] proposed another logarithmic measure of the lexical diversity of a text for avoiding the relation of this metric with the text length:

$$Maas\ Index\ a^2 = \frac{\log(tokens) - \log(types)}{\log^2(tokens)} \tag{3.16}$$

- **Uber Index**

  To end, another algorithm was proposed by Dugast in 1980, called **Uber Index U** [42]. It is the inverse measure of the **Maas Index**:

$$Uber\ Index\ U = \frac{1}{Maas\ Index} = \frac{\log^2(tokens)}{\log(tokens) - \log(types)} \tag{3.17}$$

These **TTR** improvements are all based on the relation between the number of all ***tokens*** in the text and the number of all ***types***. Now, we are going to present other metrics for measure **Vocabulary Richness** that use different algorithms but they are also based on **TTR**.

#### 3.3.2.3 MTLD

**MTLD** (Measure of Textual Lexical Diversity) [25] is a sequential algorithm for measuring the **Vocabulary Richness** of a text. The algorithm is as follows:

Firstly, the text (as in almost all algorithms developed in this library) is **tokenized**. Then, an empty segment is created. Words are added in order to appearance in the text to the segment and the **TTR** of that segment is calculated. If the **TTR** of the segment is greater or equal than a limit (between 0 and 1), words are still adding. If not, the words in the segment are deleted and a new one is started. That limit is usually 0.72.

If the text ends and, for this reason, there are no more words to add, a partial segment value is calculated as

$$Partial\ segment = \frac{1 - TTR}{1 - limit} \tag{3.18}$$

because this value is the proportional input that this last segment makes to the calculation of **MTLD**. By last, **MTLD** value will be:

$$MTLD = \frac{tokens}{Segments + Partial\ segment} \tag{3.19}$$

**MTLD** value is not the same if the algorithm makes the analysis of the text read to the right than read to the other way around. For this reason, **MTLD** final value will be the main between **MTLD** to the right and **MTLD** to the left:

$$Final\ MTLD = \frac{MTLD(Right) + MTLD(Left)}{2} \tag{3.20}$$

#### 3.3.2.4 MSTTR

**MSTTR** (Mean Segmental Type Token Ratio) [25] is another sequentially algorithm for measuring **Vocabulary Richness**.

The algorithm is simple: firstly, the text is divided in segments with the same length (usually segments of 100 tokens). Then, the **TTR** is calculated for each segment. The **MSTTR** value is the main between the **TTR** of the different segments:

$$MSTTR = \frac{\sum_{i=1}^{n} TTR(segment_n)}{n} \tag{3.21}$$

where $n$ is the number of segments and $segment_n$ is the segment number n .

It should be noted that the last segment, which usually is not going to have 100 tokens (it usually will have less) will be discarded.

The main problem of this metric is that it does not take into account the whole text, because it works with separated segments.

An example of this metric applied to the same text that in the chapter *Text Statistics* is shown in the next graphic:

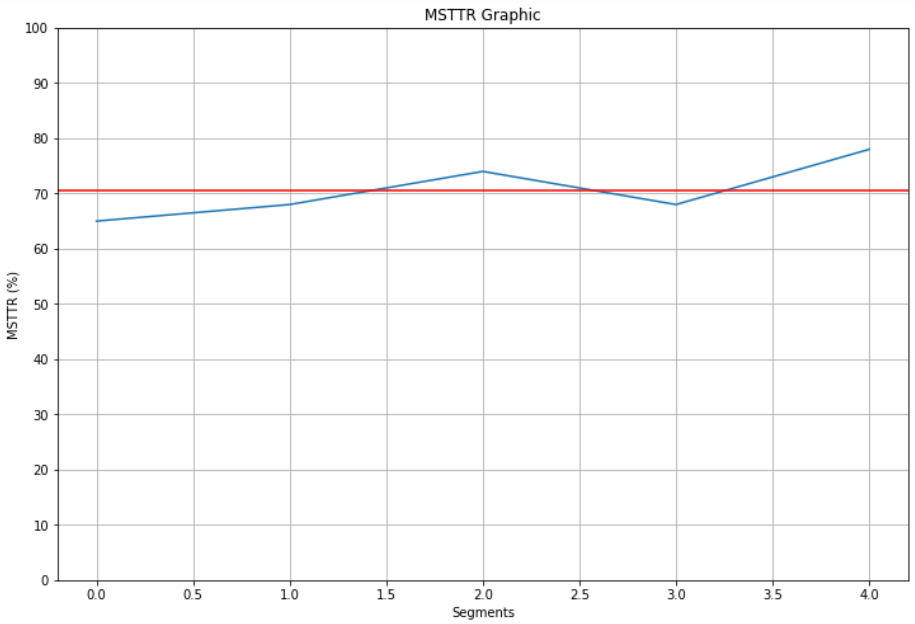



Figure 3.2: MSTTR applied to text [40]

As it can be seen, the text is divided in 5 segments of 100 tokens (the whole text has 588 words) and 1 segment of 88 words. The **TTR** is calculated in all these segments (excepting the last one), with values of 0.65, 0.68, 0.74, 0.68, 0.78 (the blue line). The red line represents the average of these numbers and it is the final **MSTTR** value.

$$MSTTR = \frac{\sum_{i=1}^{n} TTR(segment_n)}{n} = \frac{0.65 + 0.68 + 0.74 + 0.68 + 0.78}{5} = 70.6\%$$

#### 3.3.2.5 MATTR

**MATTR** (Moving Average Type Token Ratio) [7] is a **Vocabulary Richness** metric based on a window moving around the text. Usually, the window size is 500 or 100, and in this text the size window is going to be 100. The algorithm calculates the **TTR** of the first 100 tokens, then calculates the **TTR** of a segment that covers from token 1 to 101, then from token 2 to 102, and so on. Then, all the values are added and the sum is divided by the number of segments. The formula with the algorithm is shown bellow:

$$\frac{\sum_{i=1}^{length(tokens)-99} TTR[token[i], token[i+99]}{length(tokens) - 99} \tag{3.22}$$

where $TTR[token[i], token[i + 99]$ is the TTR applied to the segment composed by tokens[i] to tokens[i+99] and *length(tokens)* is the number of tokens in the text.

As before, we are going to apply the **MATTR** algorithm to the text used in the *Text Statistics* chapter. It can be seen how the **TTR** varies slowly while the window of 100 tokens is moving around the text:

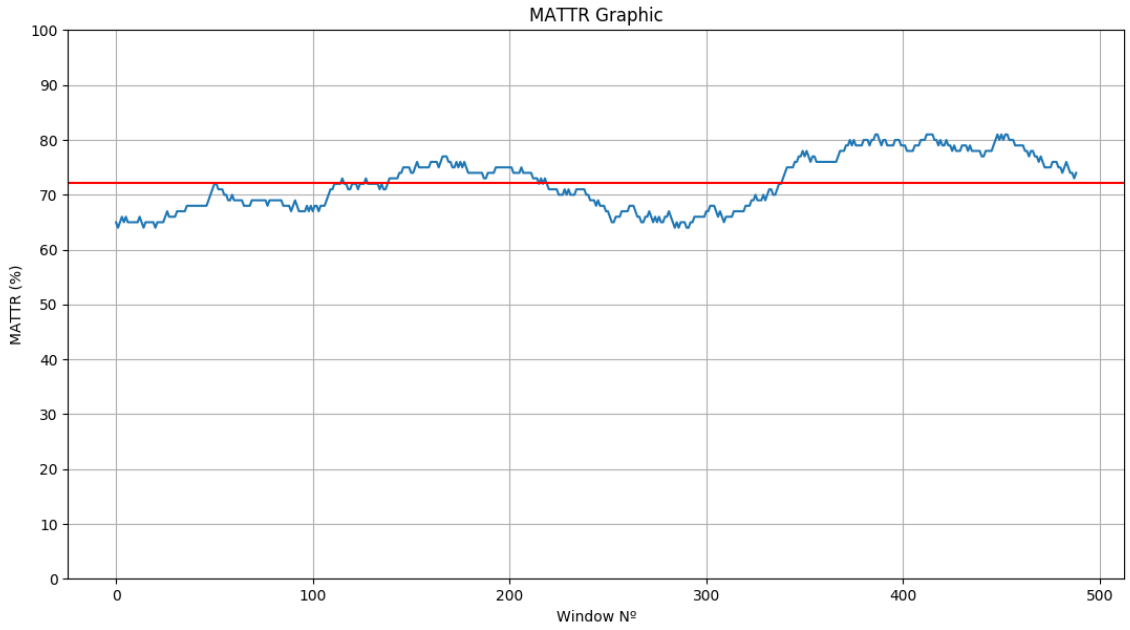



Figure 3.3: MATTR applied to text [40]

As it can be seen, the blue line is the **TTR** calculated over each segment and the red line is an average of all these **TTR** values. The **MATTR** value is a metric that can be used for measuring the variation of **Vocabulary Richness** along the text.

The graphic shows that the window moves 488 times. This is because this text has 588 tokens and the window moves from the segment 0-100 to the segment 488-588. It can be seen that from the token 0 to the 200, lexical diversity grows slowly. Then, from the token 200 to the 300 it decreases and, finally, it stills growing to the end. The average of all the values is the red line (72.19%).

For all this, the **MATTR** metric is a perfect tool for measuring the lexical diversity of a text and the variation of it along the text.

#### 3.3.2.6 HD-D

Before starting with the **HD-D** [25] index, it is necessary to talk about another index that had a big repercussion in the lexical diversity world: the vocd-D index.

For measuring the **Vocabulary Richness** of a text, the vocd-D index calculates first the D coefficient. To calculate D, the algorithm analyzes 100 random samples of the text with a length of 35 tokens and calculates its **TTR**. Then it repeats it with samples from

36 to 50 tokens. Finally, a curve is created with this information and the D coefficient is calculated and used in a formula that measures the **Vocabulary Richness** in that text.

The vocd-D index depends on the D coefficient. Years later it was shown that this D coefficient was a complex approximation to the **hypergeometric curve** and the **HD-D** index arose from that idea.

The **hypergeometric curve** [16] is a probability distribution that measures the probability of, assuming you have a population of $N$ elements of which $d$ elements are of type $A$ and $N$-$d$ are of type $B$. The hypergeometric distribution calculates the probability of obtain x ($0 \leq x \leq d$) elements of the type $A$ in a sample without replacement of $n$ elements of the original population.

The probabilistic formula that define this probability distribution is the next:

$$P(X = x) = \frac{\binom{d}{x}\binom{N-d}{n-x}}{\binom{N}{n}} \tag{3.23}$$

where $N$ is the population size, $n$ is the sample size, $d$ is the number of the elements belonging to the request type and $x$ is the number of elements in the sample that belongs to the request category.

**HD-D** index is based on this hypergeometric distribution. Specifically, the **HD-D** metric analyzes the probability that any type will appear in a sample of 42 random tokens. As for the calculation of coefficient D in the vocd-D index we used samples from 35 to 50 tokens, the **HD-D** index uses samples of 42 tokens because is a number in the middle of 35 and 50.

Once the probabilities of each type are calculated using the hypergeometric distribution and as the weight of each type on the total **TTR** will be 1/42, the sum of the probability of each type multiplied by 1/42 will be the value of the **HD-D** index:

$$HDD = \sum_{i=0}^{n} \frac{1}{42} * P(X = type_i) \tag{3.24}$$

where $P(X = type_i)$ is the probability given by the hypergeometric distribution for any type in the text.

As an example of this probability distribution, if our text length is 674 tokens and a type appears in the text in 63 times, $P(X = type) = \frac{\binom{63}{1}\binom{674-63}{42-1}}{\binom{674}{42}} = 0.065604 = 6.5604\%$, and the contribution to the TTR of this will be $\frac{6.5604\%}{42}$.

The next graph shows how the **HD-D** is calculated for the example text used before [40]:

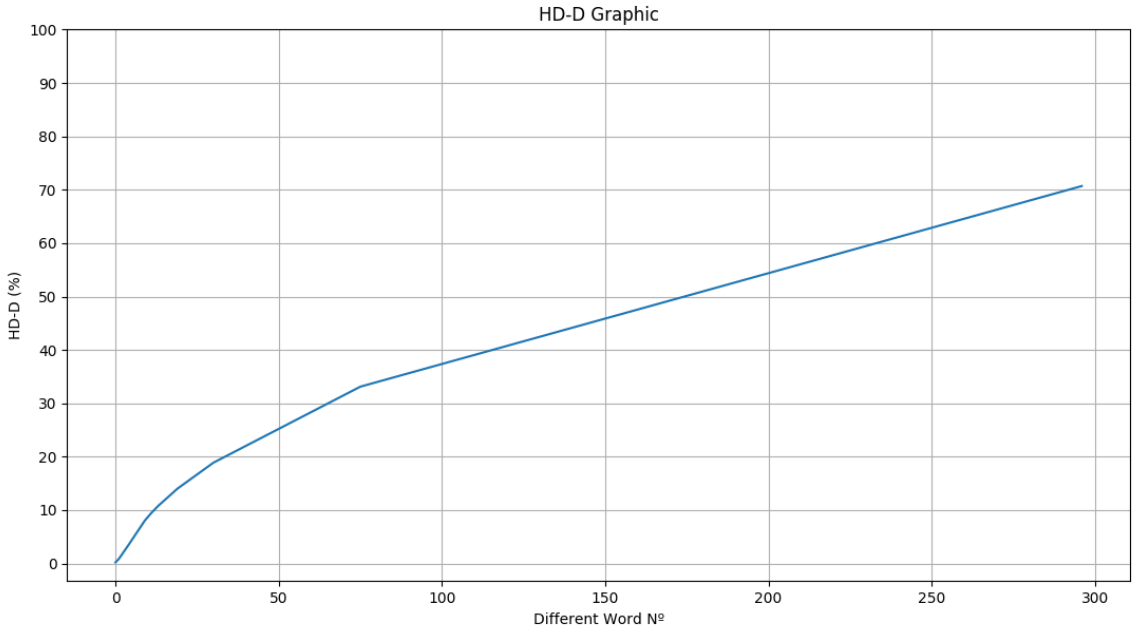



Figure 3.4: HD-D applied to text [40]

This curve represents the adding of the contributions to the **TTR** of every type in the text. First, it is important to say that the algorithm order the types depending on the frequency of that type (the most frequents first). It can be seen that the curve has a bigger slope at the beginning that in the end. This happens because the most frequent types makes a bigger contribution to the total **TTR**. It can be seen too that the text has 297 types and the **HD-D** index, that is the sum of the contributions of all the types after the application of the hypergeometric distribution, values a 70.7%. With all this, it can be said that this text has a good lexical diversity.

### 3.3.3 Formality Measure

The **Formality Measure** [17] is a **stylometry** metric for measuring the degree of formality of a text. There are two formality metrics implemented in the library : **Adjective Score** and **F Score**. Both are based on the POS tagging of the text because depending on the kind of words used in the text and the frequency of them we can say how much is a text formal or informal. Another application of this metric is to classify a text in function of the formality grade as it can be seen in the next explanation. It should be noted that these metrics are valid for both Spanish and English languages.

#### 3.3.3.1 Adjective Score

The **Adjective Score** [10] is a measure that analyzes the adjective density of a text. This score is given by the following formula:

$$Adj\ Score = \frac{N^o\ Adjectives}{tokens} * 100 \tag{3.25}$$

The **Adjective Score** value is usually under the 10%. Lower values mean that the text is informal (a conversation between two persons) and higher values will mean that the text is formal (newspaper articles, scientific publications,...). The table bellow shows the expected **Adjective Score** that every kind of text should have:

| Text Category | Total Tokens | ADJ Tokens | ADJ Density |
|---|---|---|---|
| Conversation | 649,613 | 25,506 | 3.93 |
| Other spoken | 636,982 | 29,327 | 4.60 |
| Fiction | 643,410 | 37,508 | 5.83 |
| Newspapers | 606,149 | 41,441 | 6.84 |
| Unpublished writing | 657,458 | 46,807 | 7.12 |
| Other published writing | 658,762 | 52,466 | 7.96 |
| Non-academic prose | 632,003 | 53,816 | 8.52 |
| Academic prose | 655,748 | 60,468 | 9.22 |

Figure 3.5: Adjective Score expected [10]

In the text used in the *Text Statistics* Chapter, it can be seen that it has 588 tokens, and 54 of them are adjectives. Thus, $Adjective\ Score = \frac{N^o Adjectives}{tokens} * 100 = \frac{54}{588} * 100 = 9.18\%$. In this case, the **Adjective Score** classify the text as a Non-academic prose text or an Academic prose text. But nevertheless, this text is a newspaper article. This happens because, if we see the **Readability Index** of this text, it can be seen that read it could be difficult. Therefore, it can be said that the text, even though is a newspaper article, is very formal and read it can be difficult.

#### 3.3.3.2 F Score

The **F Score** [17] is another **Formality Metric**. It is most important than the **Adjective Score** because, among other things, it takes into account all the POS tagging process and not only the adjectives tokens. The proposed algorithm for this measure is as follows:

$$F\ Score = \frac{\frac{Nouns + Adjectives + Prepositions + Determiners - Pronouns - Verbs - Adverbs - Interjections}{Tokens} * 100 + 100}{2} \tag{3.26}$$

The higher the **F Score**, the higher the degree of formality. With this formula, we can see that texts with a higher number of nouns but, for example, a lower number of verbs or pronouns, will be more formal that other texts.

In the example of the text before, the **F Score** will be:

$$F\ Score = \frac{\frac{193+54+124+105-19-59-7-0}{588} * 100 + 100}{2} = 83.248\%$$

As it can be seen, the text is a very formal text (newspapers article **F Score** usually are between the 60% and the 70%).

How it can be seen in the article published by Francis Heylighen and Jean-Marc Dewaele [17], where they study the **F Score** for texts written in Dutch, English, Italian and French, high results of the **F Score** mean very formal texts like newspapers or scientific magazines. Even so, the text before is extraordinarily formal even if it is a newspaper article. This variation can be produced by the topic that the author is talking about.

In summary, the both metrics explained here are a good manner to see how much a text is formal or informal and can get to classify the texts according to themes or type of text.

### 3.3.4 Coherence Measure

The **Coherence Measure** [11] is the last metric implemented in the library. This index returns how much coherent is a text and therefore how much cohesive is that text.

The index is based on the comparison of each sentence with the next sentence of the text. The sum of all the comparisons of a sentence with the following is the **Coherence Measure Index**:

$$Coherence\ Measure = \frac{\sum_{i=0}^{n-1} Coherence(sentence_i, sentence_{i+1})}{sentences} * 100 \tag{3.27}$$

where *sentences* is the number of sentences within the text, $sentence_i$ is the sentence number $i$ within the text and $Coherence(sentence_i, sentence_{i+1})$ is the comparison of a sentence with the following in the text.

To know how much coherent is a sentence with the following, it is necessary to analyze the semantic relation between them. This can be done working with **WordEmbeddings** models. To do this analysis, it has been used the *gsitk* library, that provides functions for working with **Word2Vec** features.

First of all, for extracting **Word2Vec** features a **WordEmbeddings** model needs to be loaded. The model used will depend on the language of the text. The available models depending on the language of the text are described in the Chapter *Enabling Technologies*.

Once it is loaded, the text is **tokenized**. But this process of **tokenization** is different from the others **tokenization** processes in the rest of the functions of the library: first, the text is **tokenized** in sentences and finally each sentence is **tokenized** in words. It is going to be shown by an example where the test text is going to be:

"*It was nearly Christmas. Katie woke up and found that the world was white and magical. - Snow,she shouted, snow for Christmas.*".

The **tokenization** for analyze the coherence in the text will be:

*[['It', 'was', 'nearly', 'Christmas'], ['Katie', 'woke', 'up', 'and', 'found', 'that', 'the', 'world', 'was', 'white', 'and', 'magical'], ['Snow', 'she', 'shouted', 'snow', 'for', 'Christmas']].*

As it can be seen, the function **tokenizes** the text in sentences and then each sentence is **tokenized** in words.

Then, with the text **tokenized** and an extractor created with a **WordEmbeddings** model loaded, using the function *transform* the text is transformed and stored as an array containing the extracted features. The transformation of the above test text can be seen here:

Finally, each vectorized sentence is compared with its following using the cosine similarity comparison:

$$similarity = cos(\theta) = \frac{A * B}{||A|| * ||B||} = \frac{\sum_{i=1}^{n} Ai * Bi}{\sqrt{\sum_{i=1}^{n} Ai^2} * \sqrt{\sum_{i=1}^{n} Bi^2}} \tag{3.28}$$

where $A$ is a vectorized sentence and $B$ is its following vectorized sentence. The cosine similarity returns a number between 0 and 1 and its number (multiplied by 100) will be the semantic resemblance between the two sentences.

To finish, all the similarities between the sentences of the text are added, divided by the number of sentences and this sum is multiplied by 100 to give a percentage that will be the **Coherence Measure** of the text.

As an example, the next graphic shows how the coherence is measured sentence by sentence:

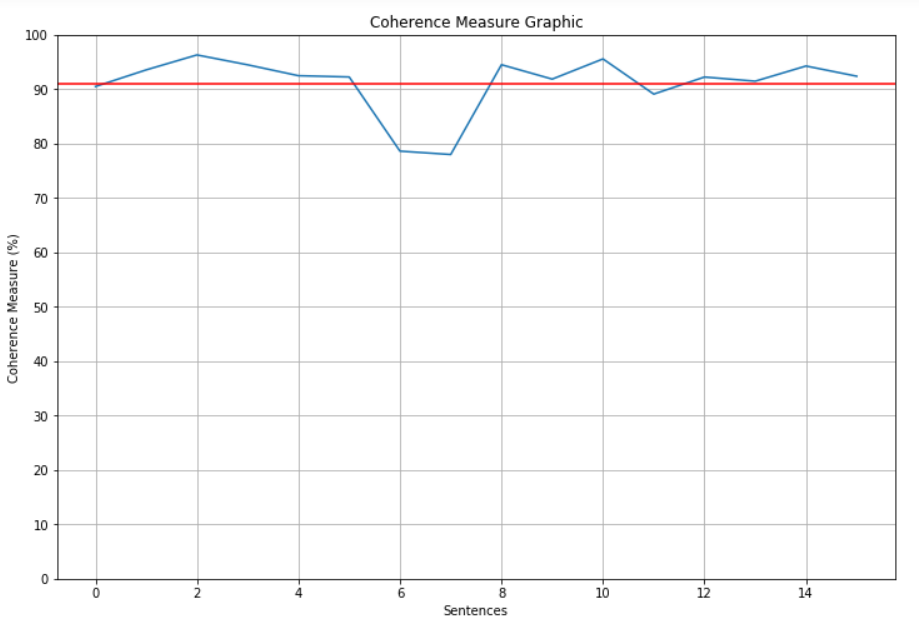



Figure 3.6: Coherence Measure applied to text [40]

As it can be seen, the coherence in the text stays almost constant over the **Coherence Measure Index** that is 91.114%. The only values that vary with this average are the comparison of the $sentence_7$ with the $sentence_8$ and the comparison of the $sentence_8$ with the $sentence_9$, that are around the 78%.

With all this, a **Coherence Measure** of 91.114% is a high value and it can be said that this text is very coherent and cohesive.

A complete analysis of the text [40] is made in the appendix.

CHAPTER 4

# Architecture

## 4.1 Introduction

In this chapter will be shown and explained all the process of compilation of data,treatment of this data and visualization. As explained at *Chapter 2: Enabling Technologies*, different tools have been used for the display of the data, like **GSI Crawler**, **ElasticSearch**, **Luigi**, **Senpy** and some web tools and web languages (HTML, CSS, JavaScript...).

## 4.2 Architecture

This section will show all the components used in the development of the **Dashboard** in which is going to be shown the results of the use of the stylometry library.

As it can be seen in the following figure, this part of the project has been developed using a pipeline (monitored by **Luigi**) composed of a part of compilation of the data, analysis of the data, storage of the data and visualization.

It is important to say that the Dashboard is based on the Trivalent [14] project (Terrorism pReventIon Via rAdicaLisation countEr-NarraTive), of which the use of the **GSI Crawler** tool has been maintained and the visualization part has been extensively modified.

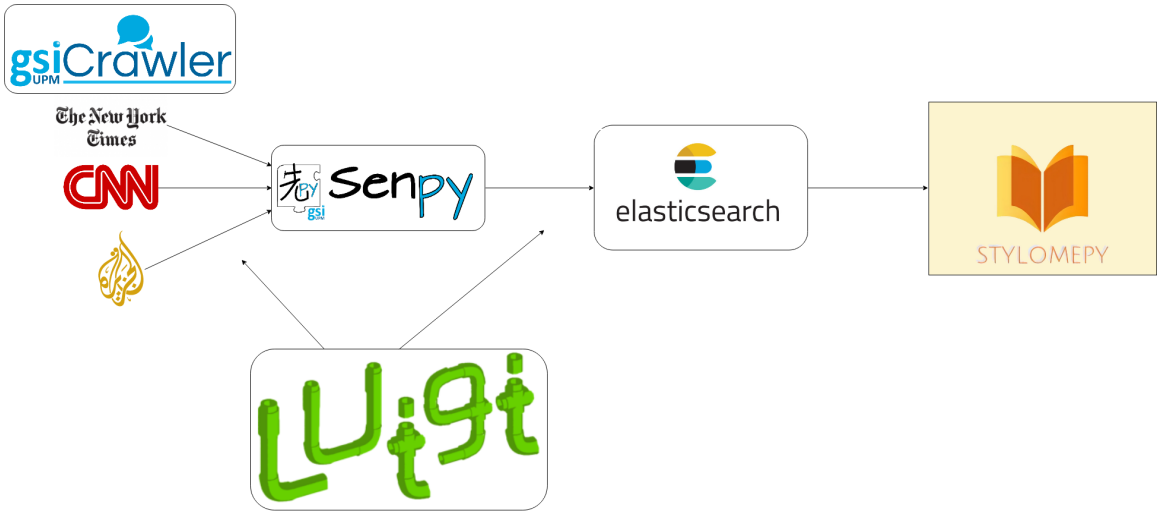



Figure 4.1: Architecture of the visualization part

### 4.2.1 Docker and Docker-Compose

For the deployment of the environment, has been widely used **docker** and **docker-compose**. Firstly, a **Senpy** image with the developed style plugin is mounted using **docker**:

$$sudo\ docker\ build\ .\ --tag\ senpyimage$$

Once the **Senpy** image is mounted, we can build the whole system using **docker-compose**. In particular, **GSI Crawler**, **ElasticSearch**, **Senpy** and **Luigi** are mounted in containers and then, with **docker-compose**, this multi-container application is runned.

$$sudo\ docker-compose\ up\ --build$$

Then, the **Stylomepy Dashboard** is executed with another **docker-compose**. When the whole system is running, **ElasticSearch** will be available at *localhost:19200*, **Senpy** at *localhost:5000* and the **Stylomepy Dashboard** at *localhost:8080*.

### 4.2.2 Data Compilation

For this task, the **GSI Crawler** tool has been used for scrapping some online newspapers for finding news related to terrorism: *The New York Times*, *CNN* and *Al Jazeera*.

The scrapper works finding in the news the word *terrorism*. At the time this text is being written, 115 news items have been scraped from the different sources, as shown in the following figures:

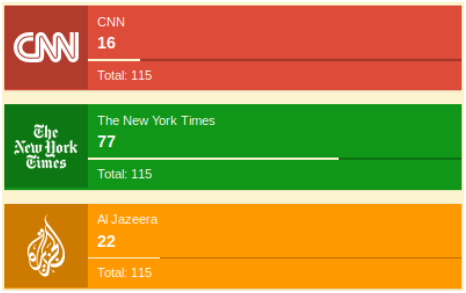



Figure 4.2: News Number Chart

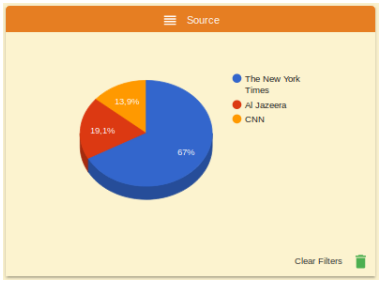



Figure 4.3: News Google Pie Chart

### 4.2.3 Data Analysis

The task dedicated to the analysis of the data compiled by **GSI Crawler** is **Senpy**. As stated above, **Senpy** is based on the use of plugins to make the analysis of a piece of text. In this project, a new plugin has been developed for make the analysis of the style of the texts. The input of the plugin will be a text to analyze and the output will be a JSON document with the analysis of the style among other things.

This plugin admits three extra-parameters: a parameter to tell **Senpy** if coherence has to be measured, a parameter to determine how much the window has to measure in the calculation of the MATTR (usually is 100) and another parameter to determine how much the MTLD value has to measure (usually is 0.72). **Senpy** provides a Playground environment for testing. The **Senpy** Playground with the Style Plugin loaded is shown in the next figure:

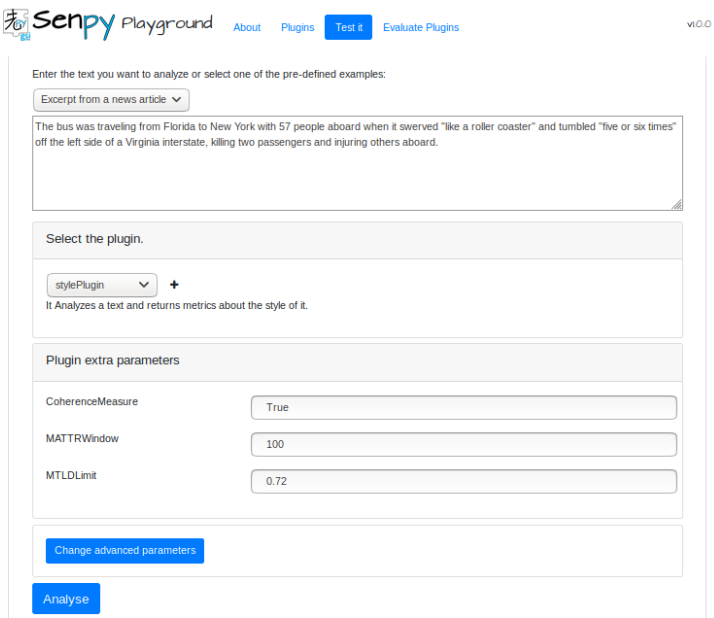



Figure 4.4: Senpy Playground

The plugin analyzes the style of a text measuring it with all the metrics explained in the previous chapter and returning a JSON with all these metrics. To facilitate the use of the tool and the readability of the response, the data is given sorted and represented according to the following vocabulary designed to show the data in the simplest way possible:

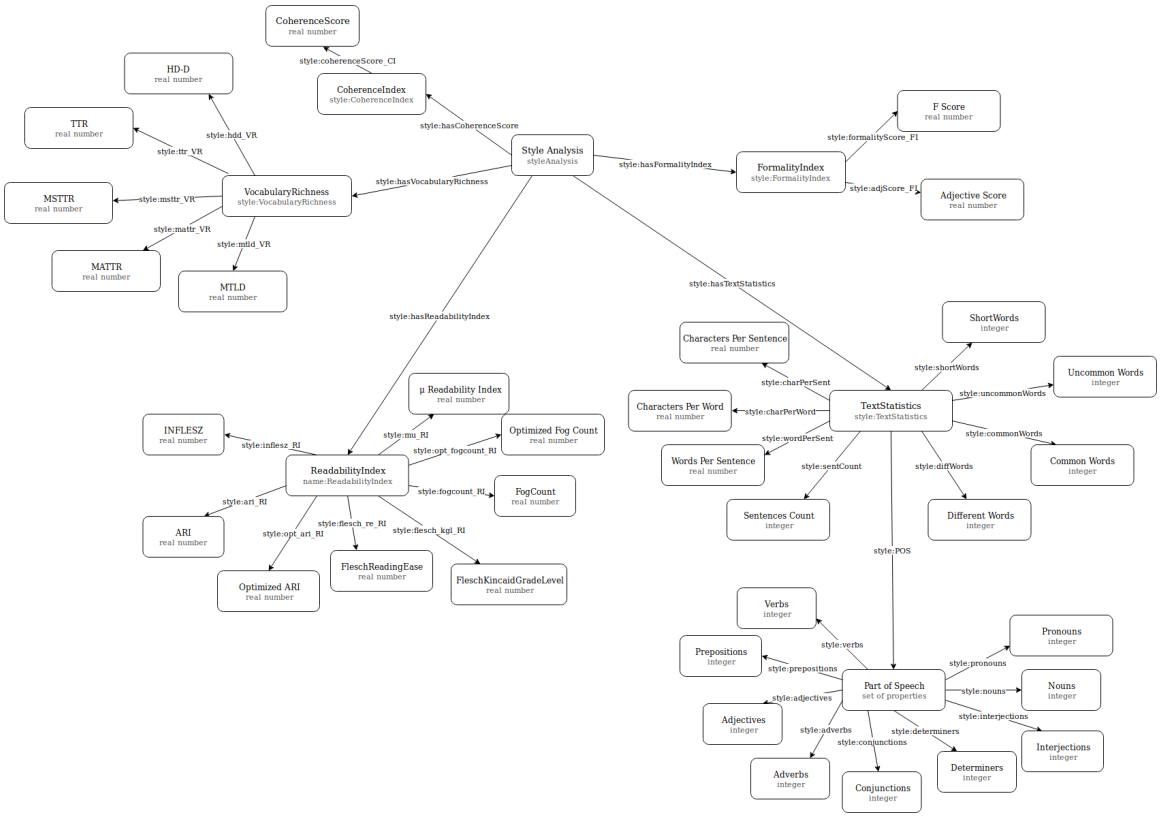



Figure 4.5: Senpy style plugin JSON vocabulary

In the pipeline, the **Senpy** process takes in its input the output of the **GSI Crawler** analysis (a new of a newspaper) and its function is to analyze the style of these news. Finally, it returns a JSON file for each new based on the previous vocabulary.

### 4.2.4 Data Storage

After the **Senpy** analysis, the different JSON documents including the style analysis should be stored. For this task, in this project it has been used **ElasticSearch**.

For each news analyzed, the JSON file with the style analysis of this new is appended to the appropriate index in **ElasticSearch**. For this reason, each entry in the **ElasticSearch** style analysis index will be a different news. In addition, apart from storing the style, it is also stored the news title, the news text and the news published date.

With all this data, it will be simple to extract data from **ElasticSearch** to show it in

the **Dashboard**.

### 4.2.5 Data Visualization

Data visualization is the main task of this process. In the developed **Dashboard** it can be selected a piece of news to see the and to analyze the style of it. The name of the **Dashboard** will be ***Stylomepy***.

The **Dashboard** has been developed using typical web technologies, like HTML, CSS or JavaScript. Specifically, it has been used Polymer, a JavaScript library that allows us to create web components or to use predefined widgets.

The **Dashboard** is widely based on the Trivalent Dashboard developed by the **GSI group**. In particular, the top of the ***Stylomepy*** **Dashboard** inherits three components from the Trivalent Dashboard, as it can be seen in the next figure:

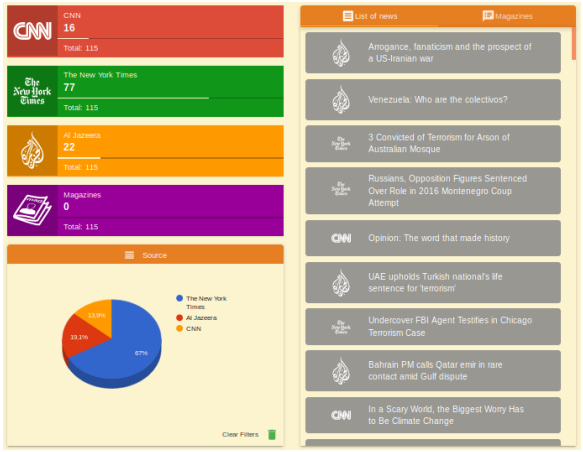



Figure 4.6: Dashboard news source and select menu

As it can be seen, the source of the compiled news is showed in the **Dashboard**. On the right, a select menu, where we can choose the news we want, is displayed. If a news is selected, a modal window like the next one is shown:

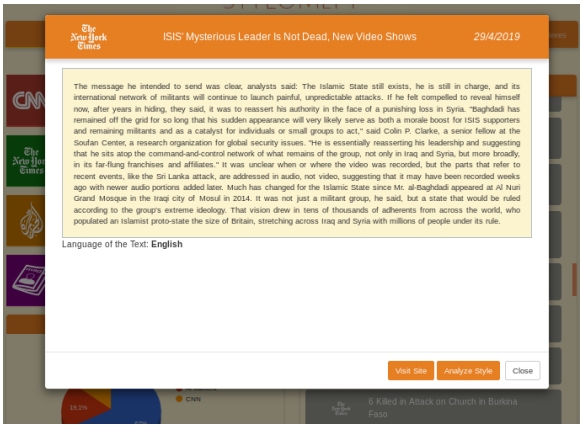



Figure 4.7: Dashboard news modal window

In the previous figure it can be seen the modal window displayed when a news is selected. In the middle of the modal window we can see the text of the news and, at the top, it can be seen the title of the news, the newspaper source and the publication date. At the bottom there are three buttons. The first one is for visiting the web in which the news was published. The second one is for analyzing the style of the news. The last one is for closing the menu. Both this menu and the previous charts have been made using Polymer components, in particular News Charts (for the select menu and the modal window), Number Charts and a Google Chart (the pie graphic).

The **Stylomepy Dashboard** has been made to analyze the style of texts. In particular, the uploaded texts in the **ElasticSearch** index are news related with terrorism or terrorist texts. With the *Analyze Style* button we could select which news we will want to analyze. Thus, when this button is selected, the page will show the style of the selected news in different web components that are going to be explained next.

- **Number Chart**: As was explained before, *Number Charts* are used in the **Dashboard** to show how many news from a certain source there are in our **ElasticSearch** index. In the *Figure 4.6* it can be seen an example.

- **News Chart**: As was explained before too, the news chart shows the news and it gives us the possibility to see them. It consists of two parts:

  - The first one is shown in the *Figure 4.6*. News are ordered in a list and we can see the title of them. We can select them too.

  - The second one is the modal window of the *Figure 4.7*. In this window, we can see the title of the news, the publication date, the source, the language of the text and three buttons. One of them is the *Analyze Style* button.

- **Google Chart**: The Google Chart is the most important component in the **Stylomepy Dashboard**. The most important properties of this component will be reported next:

  - **Field**: The field to which the data to be displayed belongs to.
  - **Data**: The JSON document with all the data stored in the **ElasticSearch** index.
  - **Type**: Is the type of the chart. This property can be *column*, *pie*, *gauge*, etc. In the **Stylomepy Dashboard** the have been used the type *column* and the type *pie*.
  - **Options and Optionsbi**: These properties allows us to personalize the chart. It can be set the height, the width, the name of the axes,the color, if a chart is going to be shown in 3D...
  - **Cols**: It is an array with the labels of the axes.

  The next figure shows how a Pie Google Chart and a Column Google Chart are displayed in the **Stylomepy Dashboard**:

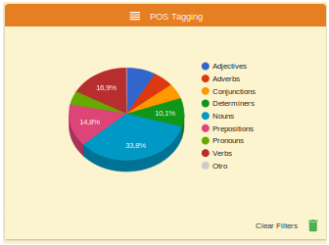



Figure 4.8: POS Pie Google Chart

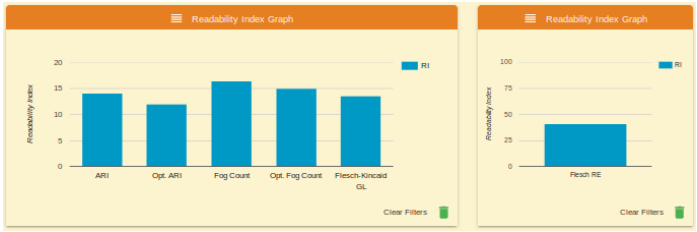



Figure 4.9: Readability Index Column Google Chart

- **d3-progress-meter**: This Polymer web component shows a progress circle in function of a percentage. The main properties needed by the component are the *title*, the *radius* of the circle and the *percentage*. The next figure is an example of this component:

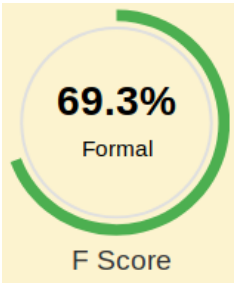



Figure 4.10: F Score Web Component

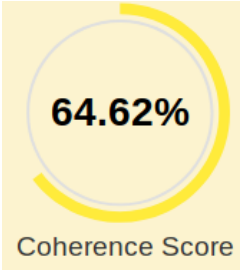



Figure 4.11: Coherence Web Component

- **Gauge with needle**: this a modified JavaScript component based on the Plotly library. It marks with the needle a value based on the data. An example of it is in the image below:

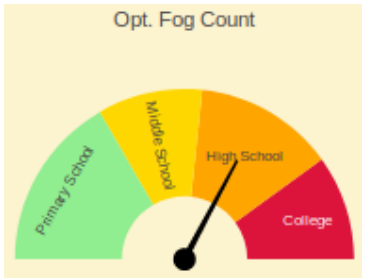



Figure 4.12: Optimized Fog Count Needle Gauge Chart

The complete **Dashboard** can be seen in the annexes.

CHAPTER 5

# Case of Use

## 5.1 Introduction

There is a lot of interesting applications where using this stylometry library can be useful. The study of the style of a text and compare it with another texts can be useful to know the differences between some writers, to classify texts based on their style, prevent plagiarism, etc.

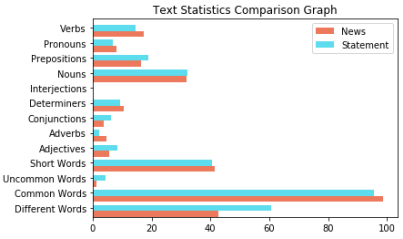
This comparison could be useful, to give concrete examples, to compare the style between two politicians and have a general idea about how each one convince people in one way or another or to know in advance if a text focused on a specific audience is effectively focused on that audience.

This chapter will focus on another possible use case: the analysis of the style of terrorist texts and another texts that talk about terrorism and the comparison between them. To analyze it, a terrorist text and a text that talk about terrorism are going to be compared (both in English and Spanish languages).

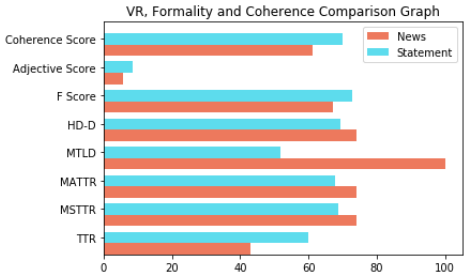
## 5.2 English Comparison

In this section, it is going to be compared a text (a new) talking about ISIS terrorists [18] and an ISIS statement [26], specifically, the Brussels attack ISIS statement. The next figures

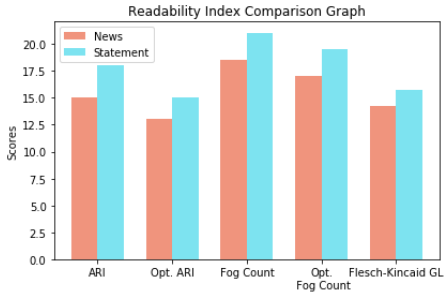
show the style of these texts:



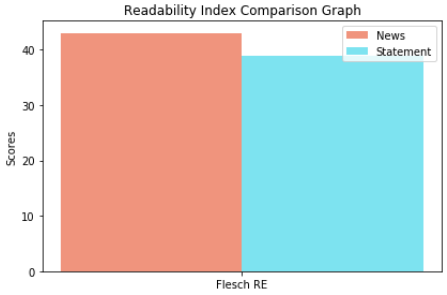

(a) Text Statistics Comparison




(b) VR, Formality and Coherence Comparison




(c) Readability Index Comparison




(d) Readability Index (Flesch RE) Comparison

Figure 5.1: Style Metrics Comparison between a news talking about ISIS terrorism [18] and an ISIS statement [26]

Firstly, we can see (Figure (b)) that the news text is less coherent than the ISIS statement. In fact, the statement has a **Coherence Score** of 70.01 % and the news has a 61.15 %. For this reason, it can be said that the statement is a very coherent text and the newspaper article is coherent too but not as much as the other.

Then, both Formality Indexes (**Adjective Score** and **F Score**) are higher in the statement text that in the newspaper article. This indexes can be interpreted as follows: although the two texts are formal texts, the statement is more formal than the article.

With regard to **Vocabulary Richness** metrics, all of the implemented metrics (except the TTR, but it is not important for measuring the style) have lower values in the statement than in the news text. Even so, the two texts have a high **Vocabulary Richness**. As it can be seen, the most important difference between the two texts in the **Vocabulary Richness** metric is that the MTLD metric is around 100 % (very high) in the article text and in the statement texts is around 50 % (normal).

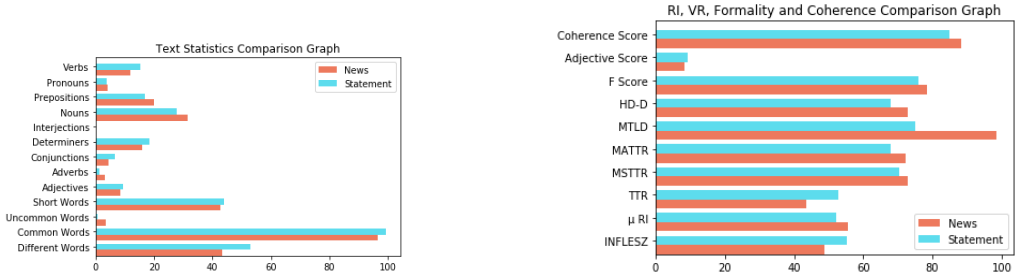
To end, if we look at the third graph, we can see that all the **Readability Indexes** show that the statement text is more difficult to read than the article text. Also in the last figure, the **Flesch RE** graph shows that the statement is more difficult to read, because

this metric, the smaller it is, the more difficult it will be to read.

As a summary, we can say that the statement is more coherent, more formal, more difficult to read and with less lexical diversity. Also we can say that there are metrics that differ, as the **MTLD**.

## 5.3 Spanish Comparison

Now, we are going to see an example like the previous one but with texts written in Spanish. Specifically, it is going to be compared a newspaper article [9] talking about ETA and a statement written by this Basque terrorist band published on 10/20/2011 [3].




(a) Text Statistics Comparison

(b) Readability Index, VR, Formality and Coherence Comparison

Figure 5.2: Style Metrics Comparison between a news talking about ETA terrorism [9] and an ETA statement [3]

In this case, it can be seen that the news text is more coherent than the statement. Even so, both texts are very coherent, having each one a **Coherence Score** higher than 80 %.

Both **Formality Index** are higher values. It would point out that the **Adjective Score** of the statement is a little higher than the **Adjective Score** of the newspaper article. On the other hand, the **F Score** of the statement is a little lower than the **F Score** of the news article, being both very formal texts.

The **Vocabulary Richness** metrics all follow the same pattern: all this metrics, except the TTR (it is not important), are higher in the news text than in the statement. Furthermore, the **MTLD** value in the newspaper article is almost 100 %, when in the statement is around the 75 %. Thus, we can say that both texts have a good **Vocabulary Richness** values.

To end, the Spanish **Readability Index** metrics have medium values, establishing that these two texts are difficult or a bit difficult to read. Anyway, according to the $\mu$ **Readability Index**, the news text is a little more difficult to read. On the other hand,

the **INFLESZ** values establish that the statement is a little more difficult to read.

## 5.4 Evaluating the Results

In the previous sections, it has been studied a case of use of the library based on the comparison between a news text talking about a terrorist group and a statement written by this terrorist group. In the next schema is described metric by metric the results of this analysis:

- **Coherence Measure**: The results depends on the texts. In the English case, the statement was more coherent than the article, but in the Spanish case is the opposite. Anyway, both the articles and the statements are very coherent texts.

- **Formality Measure**: In both languages, the **Adjective Score** of the statement was higher than the **Adjective Score** of the news text. On the other hand, the **F Score** of the Spanish case was higher in the newspaper article than in the statement and in the English case was the opposite. Anyway, both **Formality Indexes** metrics are very high, so it can be said that all texts analyzed were very formal texts.

- **Vocabulary Richness**: In this case, both in the texts written in Spanish and in English all the **Vocabulary Richness** metrics are lower in the statements than in the news texts (excepting the TTR metric, but it is not an important metric to measure the style of a text).

  Furthermore, there is a very interesting value. The **MTLD** metric in the news text is always around the 100 %, but in the statements is much lower. For this reason, the **MTLD** value is one of the most relevant metrics to differentiate a terrorist texts than others like news.

  Finally, it can be said that all texts have a good **Vocabulary Richness**, with metrics with values around the 70 %.

- **Readability Index**: All of the **Readability Index** values show that these texts are difficult or a bit difficult to read. Also, is true that all the indexes of the statement texts, excepting the $\mu$ **Readability Index** (but for very little), are lower than the news texts.

Finally, it can be seen that, analyzing the style of texts, although they probably talk about the same, they can be differentiated one from another using these metrics, making a good comparison and with a good analysis of the data.

CHAPTER 6

# Conclusions and future work

In this chapter we will describe the conclusions extracted from this project, and the thoughts about future work.

## 6.1 Conclusions

In this project, a stylometry library has been developed. This library allows the user to measure the style of a text and to see the most characteristic features of that text.

The first conclusion is that each person writes in a different way. For this reason, each text is totally different from other. Nevertheless, it is possible to classify the texts by the difficult to read them, the target audience, etc.

Each text published in a different field (a newspaper, a novel, a scientific publication, etc.) needs to be suitable to this field. For example, if someone is writing a story for children, that text must be easy to read, not very formal and with few different words. If another person wants to publish a news in a newspaper, the text must be prepared for the readers of that newspaper.

Definitely, it is possible to use the developed stylometry library for measuring the style of a text and to guarantee that the text meets specific characteristics.

As it has been seen, measuring the style of texts can be useful for comparing texts. As the previous examples, it has been seen that some metrics exist that allows us to classify texts. In this project we saw the differences (that it seems that they tend to meet) between

a terrorist text and a text talking about terrorism, like the **MTLD** metric. Using the style to prevent or to analyze what texts are on the network could be interesting.

Furthermore, a visualization module can allow the user to have an easy vision and an easy interpretation about the style of a text (in our case news related with terrorism). For this, the user is provided with a **Dashboard** to analyze the style of the texts, to see how is the style of this text in an easy way and to interpret the results.

On the other hand, the library is developed in Python, and it is accessible for everyone in an easy way. Simply importing the library and applying its functions to a text we can see the metrics that measure the style of that text.

## 6.2 Achieved goals

The achieved goals for the project are described in this section.

- **Develop of a Python stylometry library:** A Python library for measuring the style of texts has been developed. It has a lot of functions that calculate different metrics for measuring the style.

- **Implement the library in Spanish and English:** There are some metrics that do not work well in both languages. For this reason, each function in the library checks the language first and then continues.

- **Developing a Senpy plugin: Senpy** is necessary for make a correct analysis of any text and then be able to use it in another environment.

- **Deployment of a Visualization System:** Based on some tools like **GSI Crawler**, **Luigi** or **Senpy**, the text is collected from the web, analyzed and uploaded to an **ElasticSearch** index to be able to visualize it in the **Dashboard**. The **Dashboard** is made with the Polymer JavaScript library and is a very good way for seeing the results of the analysis of a text.

- **Checks and comparisons between the style of two texts:** It has been made the comparison of the style of two different texts (in this work, in the *Chapter 5: Use Case*, was only made the comparison between a terrorist text and a news talking about terrorism.

## 6.3 Problems found

The biggest problems founded during the development of this project are explained here:

- **Learning new technologies**: Before the starting with a new technology learning it is needed. Some technologies used (like *Gensim*) were difficult to use.
- **Good tokenization of a text**: It was difficult at the beginning to tokenize a text. Problems like undesirables signs in the words was common in the first months but, finally, it was perfected.
- **Deployment of the visualization system**: Due to deployment failures the visualization system could not be implemented. One off The problem was that the outputs of the modules that form the pipeline were not good inputs for the next modules in this pipeline. Other problems were related with the environment system, like problems with Docker dependencies, npm dependencies or failures in the configurations of the different docker-compose.yml files.

## 6.4 Future work

In this section are going to be presented possible future implementations and improvements for the library and the visualization system:

- **Add more metrics**: To improve the library and to make it bigger, others metrics could be implement to add viewpoints on style analysis.
- **Enhance metrics already implemented**: Metrics implemented in the library can always be enhanced. For example, the **Formality Measure** can be implement another function based on the kind of words used, jargon, swearwords and more.
- **Add functionalities to the Dashboard**: The **Dashboard** must be in a constant change. It can be added a tab where the user can compare two texts.
- **To train a model for the classification of the texts**: A machine learning model can be trained to classify texts depending on the style of them. Thus, texts can be classified by age, audience, difficulty, topic (terrorist texts, text about terrorism, etc.).

APPENDIX A

# Impact of this project

## A.1 Introduction

The comparison of the style of the texts is a different form of distinction between texts. It is a good way for analyzing texts and authors, mainly because each person has an intrinsic style and changing it is a very difficult task.

Just as each person speaks and expresses himself in a different way to any other person, also each person writes in a different way. For this reason, analyze the style could give us a lot of information about the writer, as well as his environment.

In this appendix, we are going to analyze the impact of this style analysis.

## A.2 Social Impact

Currently, most of the information we receive comes from the Internet from sites like social networks, online newspapers, instant messaging applications, etc. Regarding the latter, terrorist groups like ISIS thoroughly uses *Telegram* for sending radical messages [46].

Internet access is easy for everyone and each of the people who access the Internet can read some radical text and this can be dangerous. For this reason, detect in time a radical text published on the Internet or sent through an application could be crucial to stop terrorists and to protect to the population.

To control this, analyze the style of the texts published could be a good solution to remove messages with a radical style and thus protect people.

## A.3 Economic Impact

Analyzing the style of the texts can be a very good action to prevent radicalization. As it can be seen in this project, the style analysis of a text is a very simple and efficient task.

For this reason, countries could save money that they spend now on other prevention measures using the style analysis of the texts and thus prevent or reduce the probability of radicalization of people.

## A.4 Environmental Impact

To make possible the analysis of the texts published in the network, it is necessary to have the necessary equipment. This equipment includes computers, servers, and other computer material that needs a lot of energy to run.

Nowadays, this energy is mostly obtained from non-renewable energy sources. It is necessary (to take advantage of technologies such as Big Data or machine learning) to make an energy transition to encourage the use of renewable energies and use green computing models to simplify the environmental impact of this project.

## A.5 Ethical Implications

The most important ethical impact of this project is related with the privacy. The fact of reading and analyzing the style of many of existing texts in the network can be against people privacy rights. To solve this problem, it would be necessary to make laws that prohibit this type of practices or that guarantee that they are carried out in a controlled manner.

Another ethical implication that this project could have would be a bad use of style analysis. For example, instead of using it to discover radical messages it can be used for making actions related with cybercrime (knowing the style of writing of a person could give ideas to the criminal about some personality traits of the writer).

As a final conclusion, it will be necessary to pass laws for trying to prevent of bad uses of this project and others related to it.

APPENDIX B

# Cost of the System

## B.1 Introduction

In this appendix, we will give a budget including the cost The cost of the start-up of the project.

## B.2 Physical Resources

The project has been developed in a personal computer with the following characteristics:

- **CPU**: Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz, 64 bits
- **RAM**: 4 GB
- **Disk**:500 GB

Even so, it would be good to have a more powerful equipment, since at the time of developing the project with the equipment described, more capacity has been missed.

## B.3 Human Resources

:

For the development and the maintenance of this project will be necessary at least two workers: the developer and the maintenance worker.

To develop a project like this, at least they have been necessary 360 hours of development (each ECTS is 3 hours and this project has a value of 12 ECTS). Considering that the work has been carried out by a student and considering that he/she works at the UPM as a scholar, the cost of his/her salary will be around the 2,000 €.

On the other hand, the maintenance worker will be paid with a salary that can be around the 24,000 € and the 64,000 €.

## B.4 Licenses

: As all the programs and tools used in the development of the project are open-source, it will bee not necessary to pay to buy licenses of any kind.

## B.5 Taxes

:

This section will be interesting only if the project is going to be sold to another company. If this happens, the Statute 4/2008 of Spanish law establish that this sale is subject to a tax of the 15 % of the final price of the product.

APPENDIX C

# Analysis of a text

As a final example of **style** metrics and the **stylometry** library, we can see the next table where all the metrics are applied to the text [40]:

| **Style Metric** | **Index** | **Value (%)** |
|---|---|---|
| **Readability Index** | INFLESZ | 39.382 |
| | mu | 52.487 |
| **Vocabulary Richness** | TTR | 50.510 |
| | Guiraud R | 12.248 |
| | Herdan C | 89.289 |
| | Mass a2 | 3.867 |
| | Uber U | 25.856 |
| | MTLD | 79.661 |
| | MSTTR | 71.333 |
| | MATTR | 72.192 |
| | HD-D | 70.700 |
| **Formality Measure** | Adjective Score | 9.184 |
| | F Score | 83.248 |
| **Coherence Measure** | Coherence Measure | 91.114 |

Table C.1: Style Metrics of the text [40]

According with the information given in the table, the **Readability Index** shows that the text is difficult or very difficult to read.

The lexical diversity of the text (**Vocabulary Richness**) has a lot of functions. The most important are **MTLD**, **MSTTR**, **MATTR** and **HD-D** (the rest are the **TTR** and variations of it that are not very reliable), and these metrics returns high values (between the 71% and the 80%), so it can be said that the text has a very good **lexical diversity**.

Regarding the **Formality Measure**, both the **Adjective Score** and the **F Score** returns very high values, so it can be said too that it is a very formal text.

Finally, the **Coherence Measure** shows that the text is very coherent because this index returns a value around 91%.

As a resume, it can be said that this text is **difficult to read**, has **a lot of lexical diversity**, it is **a formal text** and it is a **very coherent text**.

The next graphics show the **MSTTR**, **MATTR**, **HD-D** and **Coherence Measure** graphs of the texts [40]:

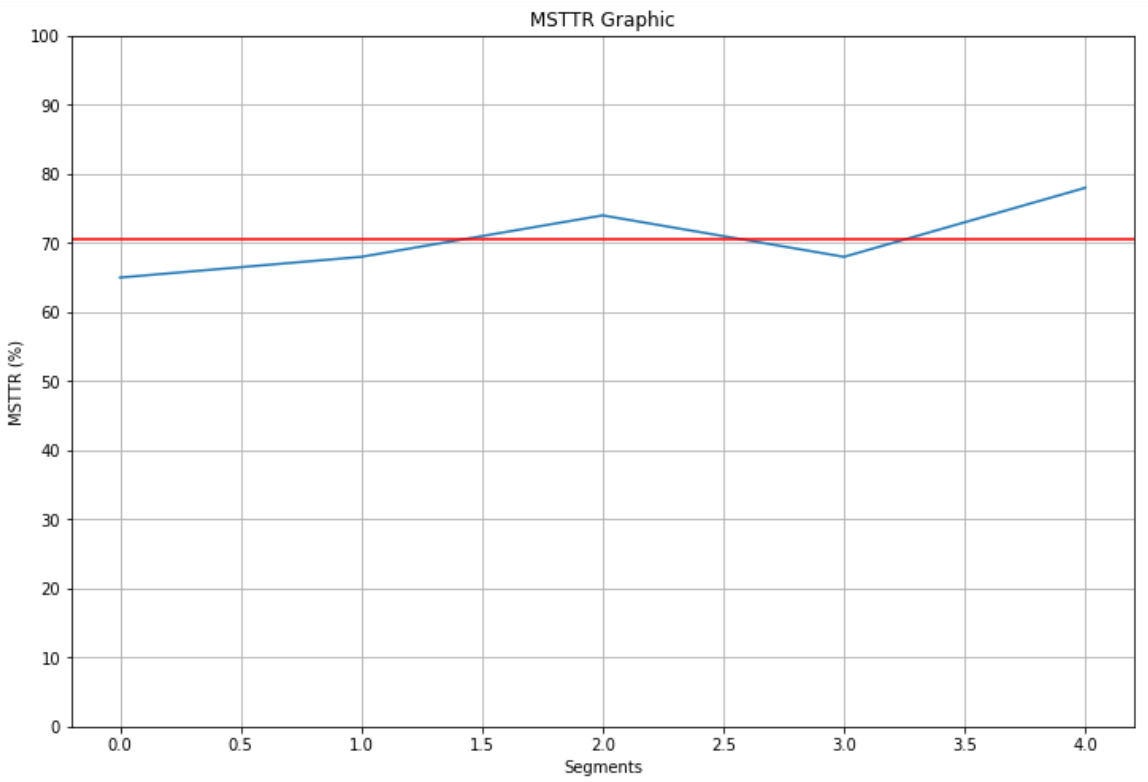



Figure C.1: MSTTR applied to text [40]

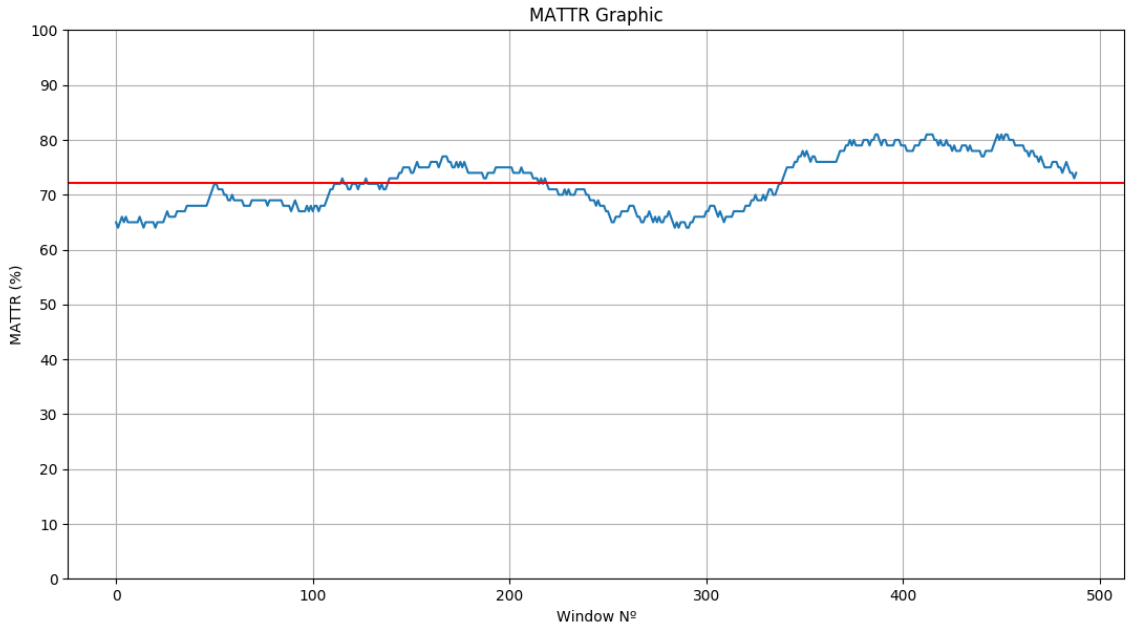



Figure C.2: MATTR applied to text [40]

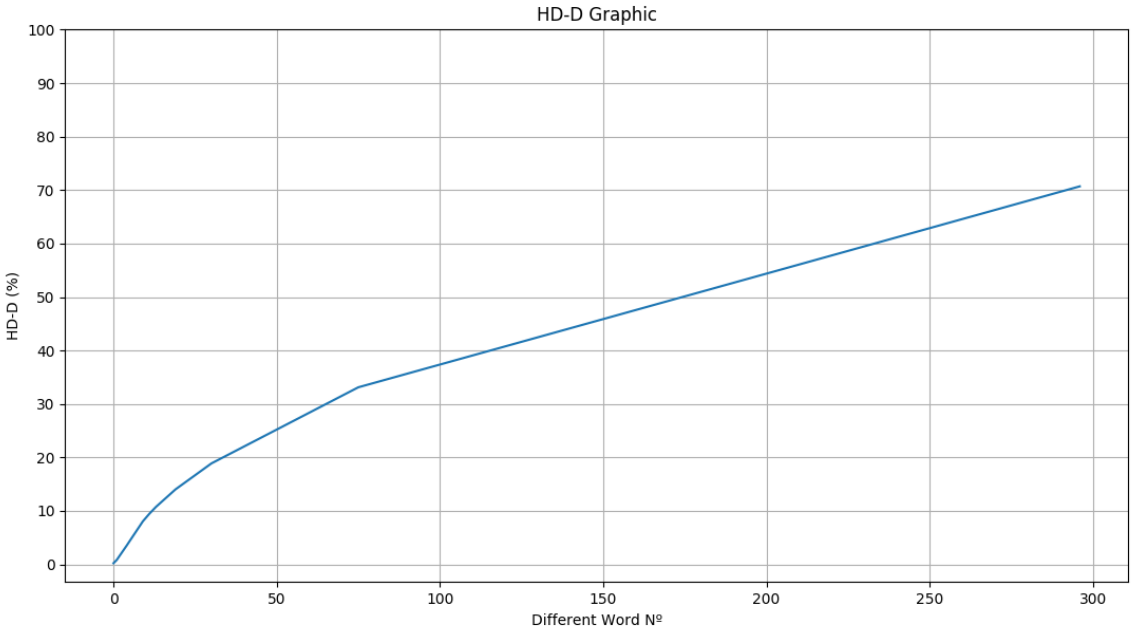



Figure C.3: HD-D applied to text [40]

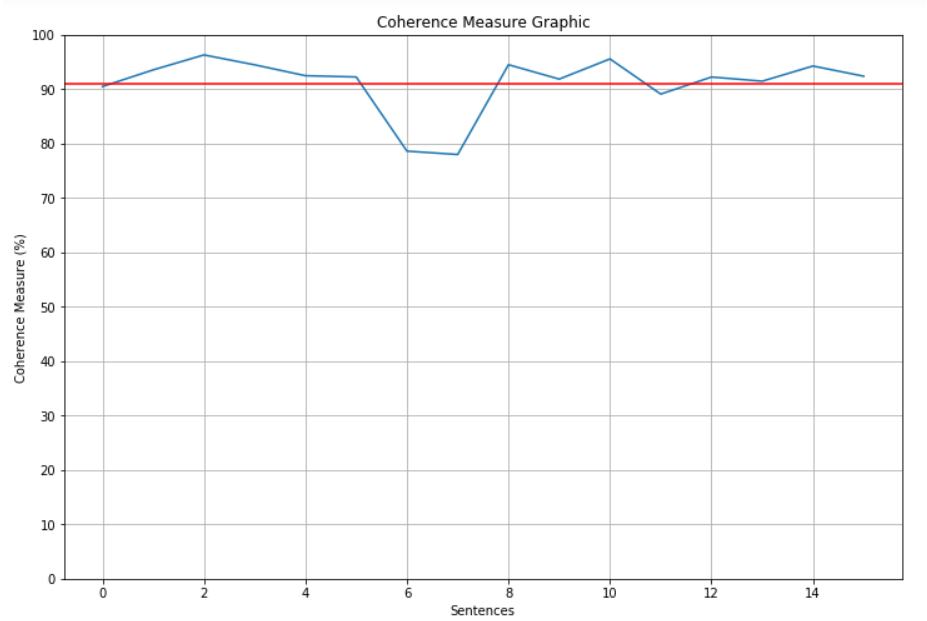



Figure C.4: Coherence Measure applied to text [40]

APPENDIX D

# Dashboard

The next pictures show how is the **Stylomepy Dashboard** and its different tabs. It can be seen each widget that composes the Dashboard, and obtain an exhaustive analysis of the style of a certain text in a glance.

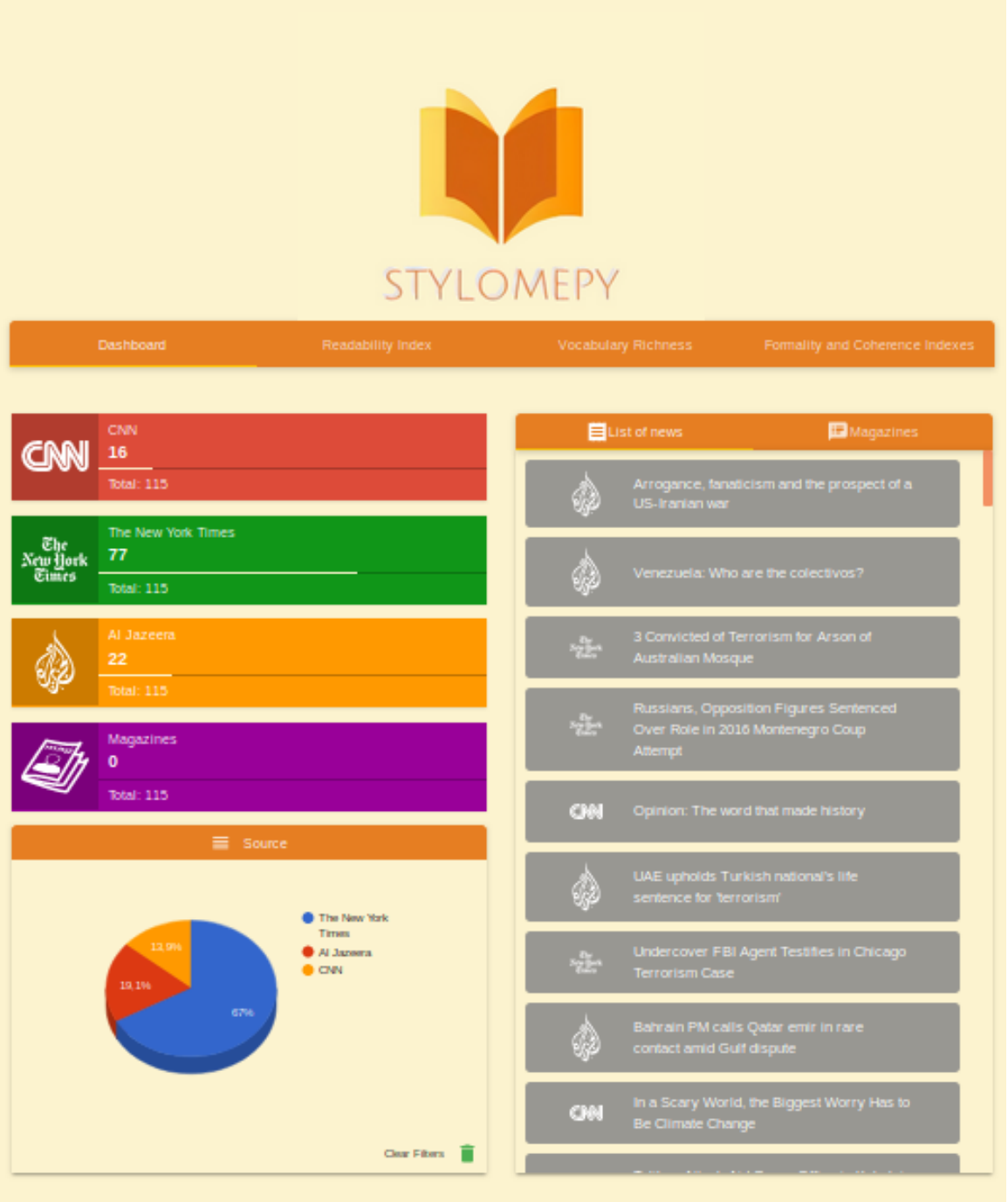



Figure D.1: News Section Stylomepy Dashboard

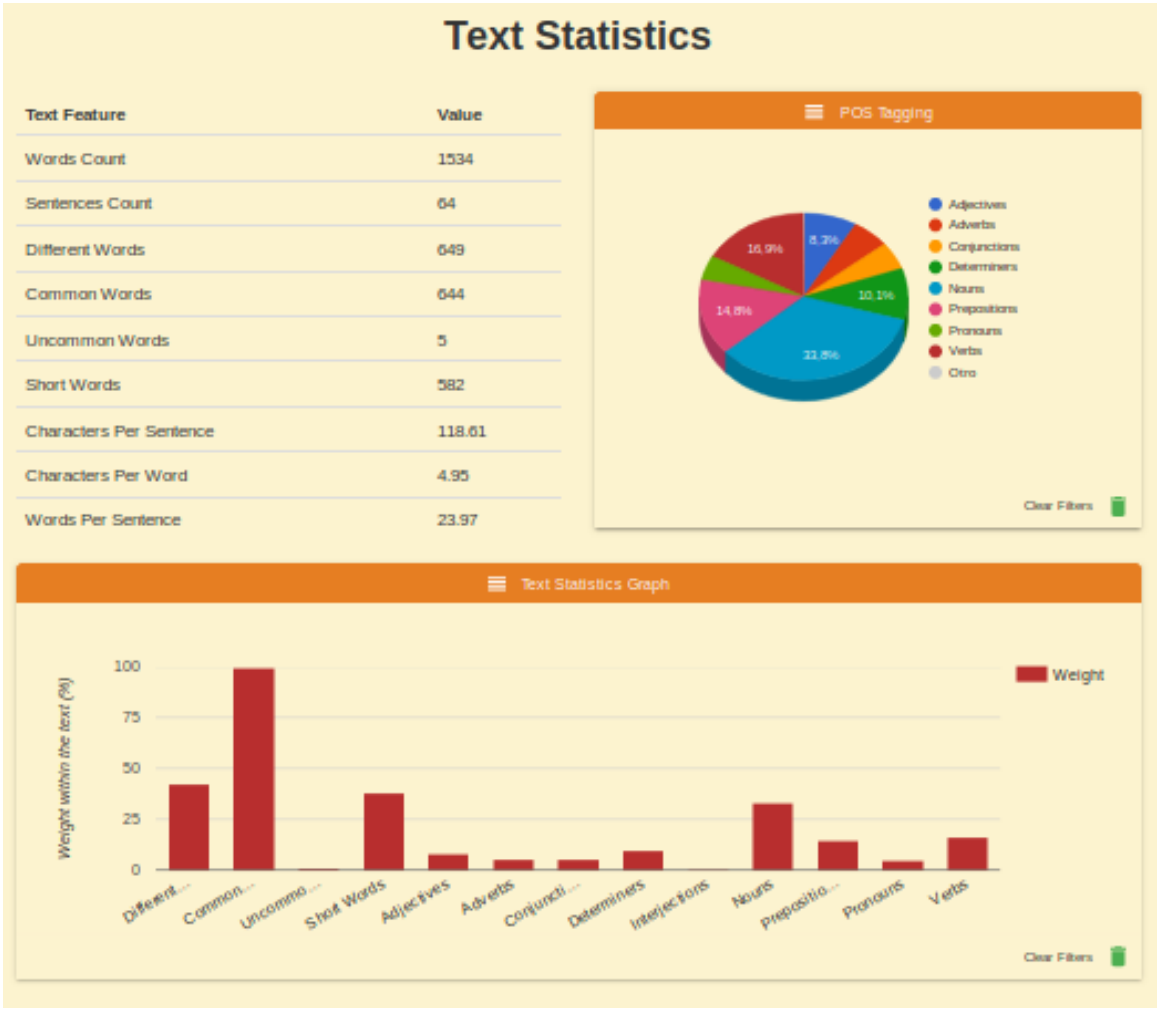



Figure D.2: Text Statistics Section Stylomepy Dashboard

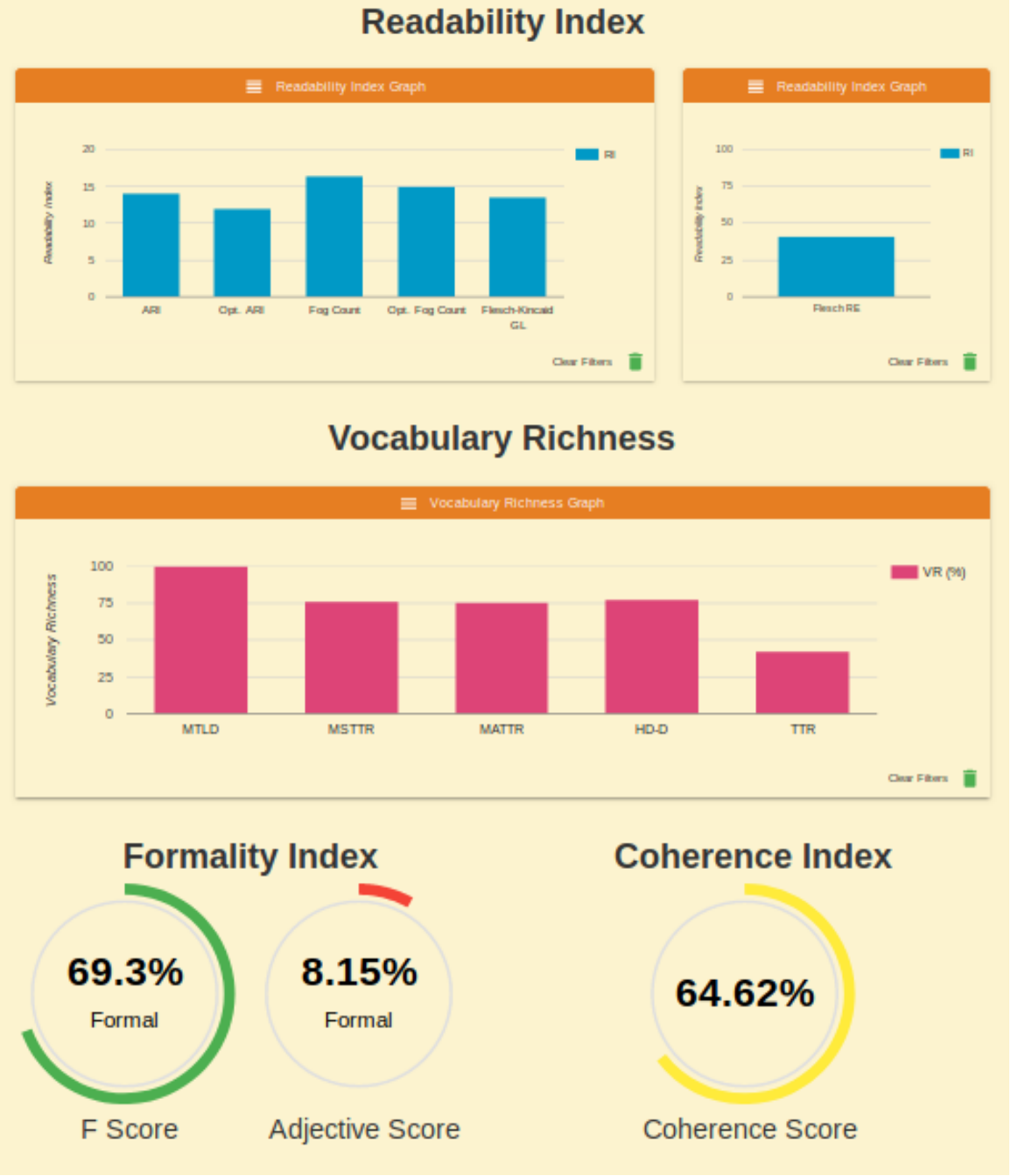



Figure D.3: Readability Index, Vocabulary Richness, Coherence and Formality Section Stylomepy Dashboard

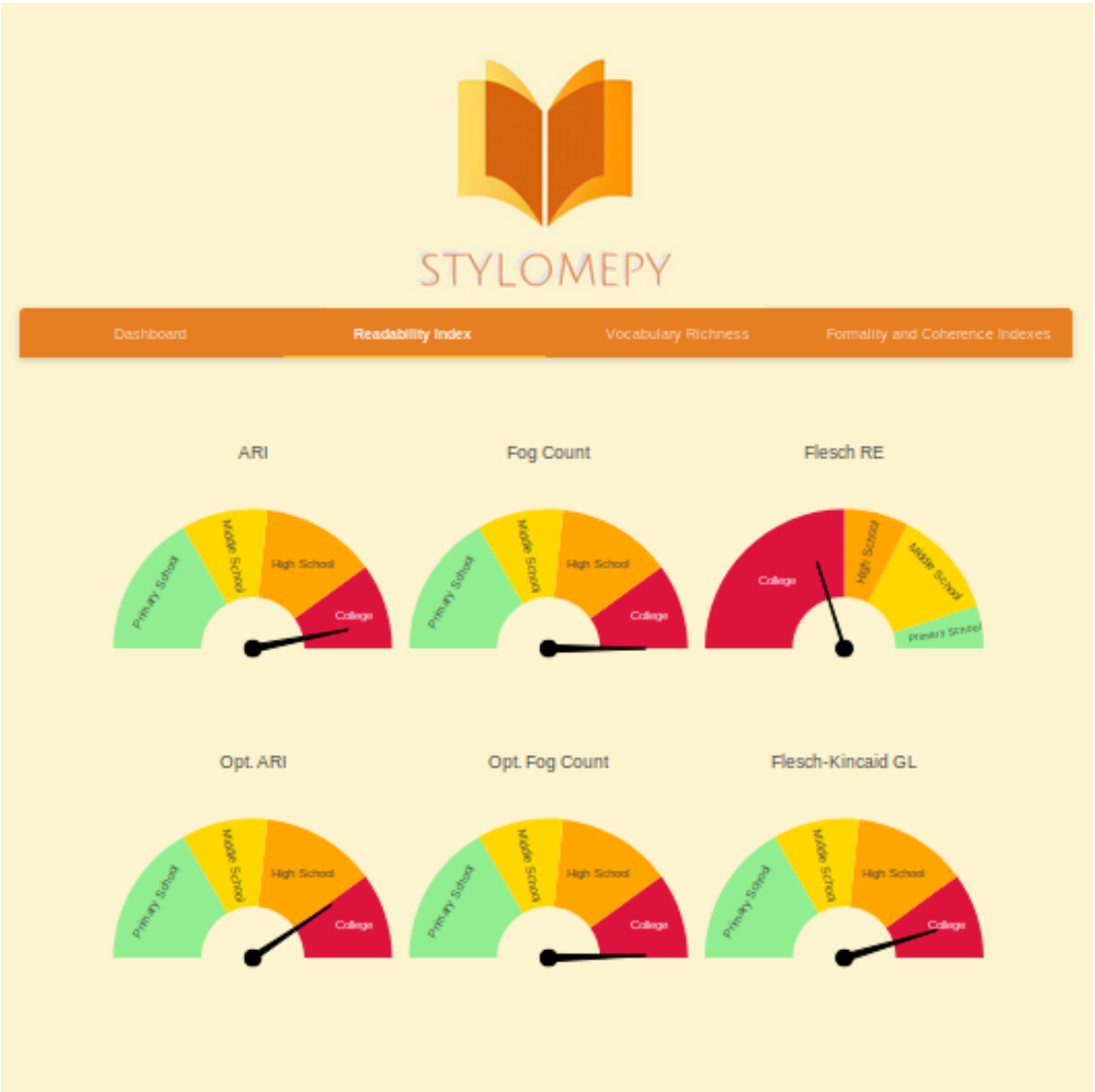



Figure D.4: Readability Index Section Stylomepy Dashboard

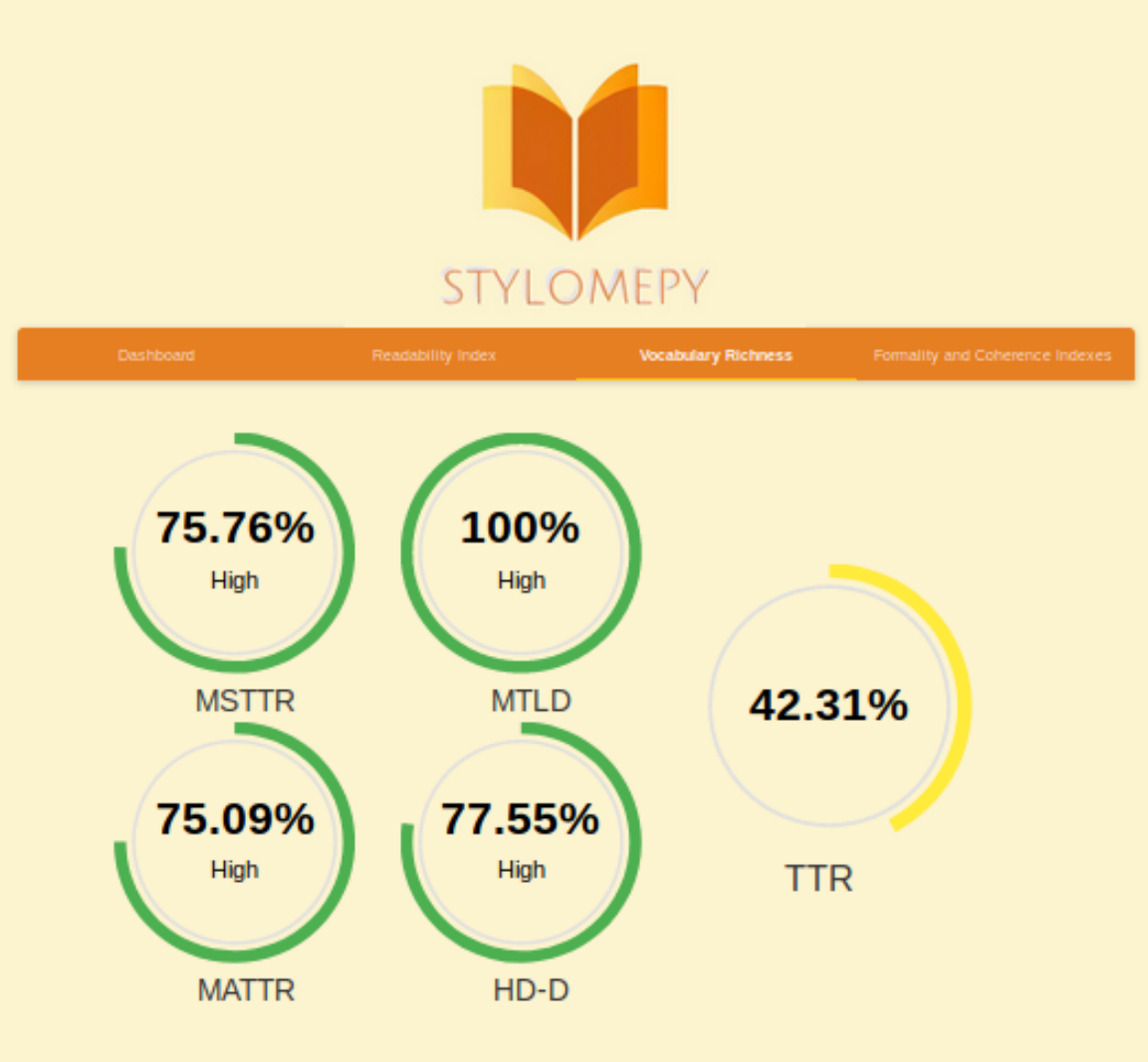



Figure D.5: Vocabulary Richness Section Stylomepy Dashboard

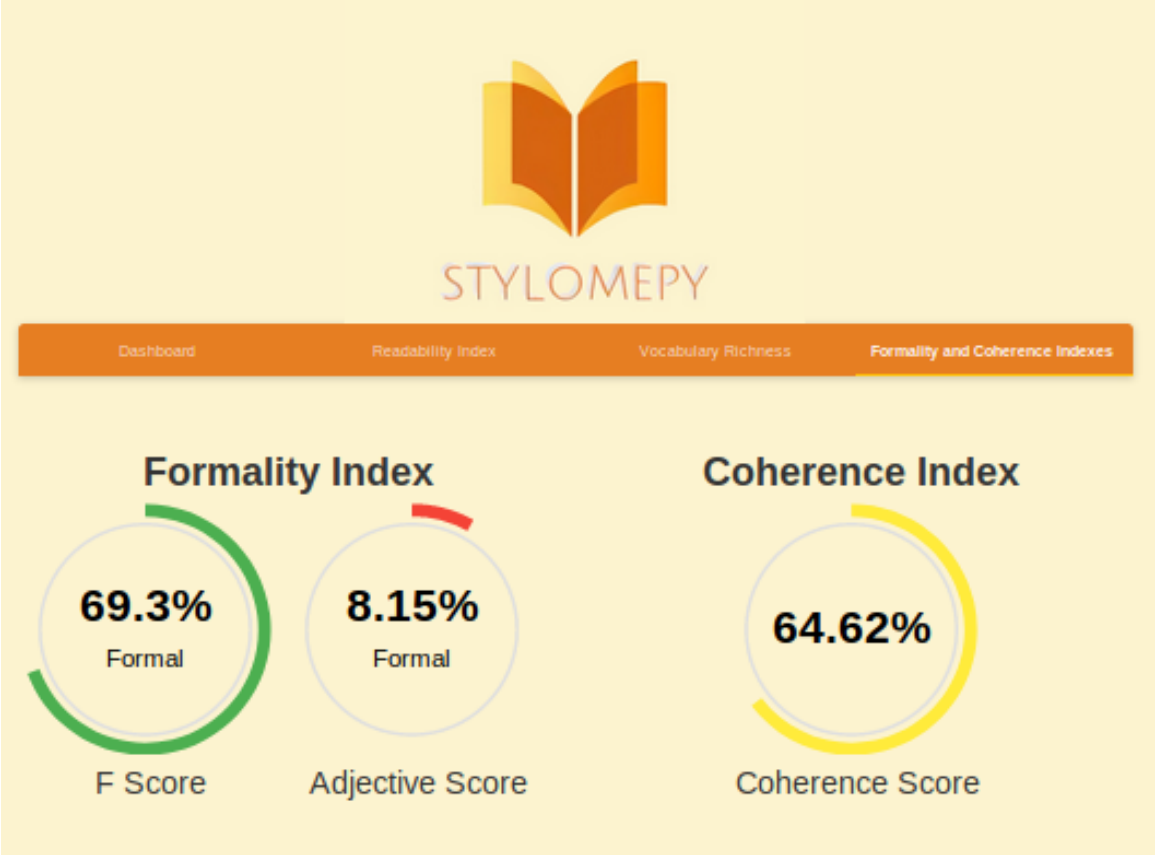



Figure D.6: Formality and Coherence Measures Section Stylomepy Dashboard

# Bibliography


[1] *The Bible: Judges 12:5–6*.

[2] *Oxford Dictionary Of English (Third Edition)*. 2010.

[3] Luis R. Aizpeolea. Eta pone fin a 43 años de terror. *El País*, 2011.

[4] Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Sánchez-Rada, and Carlos A. Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236 – 246, 2017.

[5] Inés Mª Barrio Cantalejo. Legibilidad y salud: los métodos de medición de la legibilidad y su aplicación al diseño de folletos educativos sobre salud. 2007.

[6] Margarita Caballero, Socorro Lozano, and Beatriz Ortega. Efecto invernadero, calentamiento global y cambio climático: una perspectiva desde las ciencias de la tierra. *Revista digital universitaria*, 8(10):2–12, 2007.

[7] Michael A Covington and Joe D McFall. Cutting the gordian knot: The moving-average type–token ratio (mattr). *Journal of quantitative linguistics*, 17(2):94–100, 2010.

[8] Michal Mimino Danilak. langdetect 1.0.7. `https://github.com/Mimino666/langdetect`.

[9] Jesús Duva. Eta intenta asesinar a aznar con un coche bomba. *El País*, 1995.

[10] Alex Chengyu Fang and Jing Cao. Adjective density as a text formality characteristic for automatic text classification: A study based on the british national corpus. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*, volume 1, 2009.

[11] Peter W Foltz, Walter Kintsch, and Thomas K Landauer. The measurement of textual coherence with latent semantic analysis. *Discourse processes*, 25(2-3):285–307, 1998.

[12] Robert Goodman, Matthew Hahn, Madhuri Marella, Christina Ojar, and Sandy Westcott. The use of stylometry for email author identification: a feasibility study. *Proc. Student/Faculty Research Day, CSIS, Pace University, White Plains, NY*, pages 1–7, 2007.

[13] Clinton Gormley and Zachary Tong. *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine.* " O'Reilly Media, Inc.", 2015.

[14] GSI Group. Trivalent: Terrorism pReventIon Via rAdicaLisation countEr-NarraTive. `http://www.gsi.dit.upm.es/es/trivalent`.

[15] Pierre Guiraud. *Les caractères statistiques du vocabulaire*. Presses universitaires de France, 1954.

[16] William L Harkness. Properties of the extended hypergeometric distribution. *The Annals of Mathematical Statistics*, 36(3):938–945, 1965.

[17] Francis Heylighen and Jean-Marc Dewaele. Formality of language: definition, measurement and behavioral determinants. *Interner Bericht, Center "Leo Apostel", Vrije Universiteit Brüssel*, 1999.

[18] Ben Hubbard. Isis' mysterious leader is not dead, new video shows. *The New York Times*, 2019.

[19] J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. 1975.

[20] Henry Kučera and Winthrop Nelson Francis. *Computational analysis of present-day American English*. Dartmouth Publishing Group, 1967.

[21] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[22] Wicenty Lutowslaki. *Principes de stylométrie*. 1890.

[23] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[24] Heinz-Dieter Mass. Über den zusammenhang zwischen wortschatzumfang und länge eines textes. *Zeitschrift für Literaturwissenschaft und Linguistik*, 2(8):73, 1972.

[25] Philip M McCarthy and Scott Jarvis. Mtld, vocd-d, and hd-d: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior research methods*, 42(2):381–392, 2010.

[26] Kelly McCleary, Lindsay Isaac, and Ralph Ellis. Isis claims attack on soldiers in brussels. *CNN*, 2017.

[27] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[28] Marcelo A Montemurro. Beyond the zipf–mandelbrot law in quantitative linguistics. *Physica A: Statistical Mechanics and its Applications*, 300(3-4):567–578, 2001.

[29] M Muñoz and J Muñoz. Legibilidad m$\mu$, 2006.

[30] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[31] Tita Risueño. What is the difference between stemming and lemmatization? https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/.

[32] Tony Rose, Mark Stevenson, and Miles Whitehead. The reuters corpus volume 1-from yesterday's news to tomorrow's language resources. In *LREC*, volume 2, pages 827–832. Las Palmas, 2002.

[33] J Fernando Sánchez-Rada, Carlos A Iglesias, Ignacio Corcuera, and Oscar Araque. Senpy: A pragmatic linked sentiment analysis framework. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 735–742. IEEE, 2016.

[34] RJ Senter and Edgar A Smith. Automated readability index. Technical report, CINCINNATI UNIV OH, 1967.

[35] Gregory Shalhoub, Robin Simon, Ramesh Iyer, Jayendra Tailor, and Sandra Westcott. Stylometry system–use cases and feasibility study. *Forensic Linguistics*, 1(8), 2010.

[36] Zdislava Šišková. Lexical richness in efl students' narratives. *Language Studies Working Papers*, 4:26–36, 2012.

[37] Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. Luminosoinsight/wordfreq: v2.2, October 2018.

[38] Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. Sparse word embeddings using l1 regularized online learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2915–2921. AAAI Press, 2016.

[39] Rachael Tatman. Pre-trained Word Vectors for Spanish. https://www.kaggle.com/rtatman/pretrained-word-vectors-for-spanish.

[40] Patricia Tubella. La policía investiga como "incidente terrorista" la muerte de una periodista en unos disturbios en irlanda del norte. *El País*, 2019.

[41] Alan M Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.

[42] Roeland Van Hout and Anne Vermeer. Comparing measures of lexical richness. *Modelling and assessing vocabulary knowledge*, pages 93–115, 2007.

[43] Pascal van Kooten. contractions 0.0.18. https://github.com/kootenpv/contractions.

[44] Ben Verhoeven, Walter Daelemans, and Barbara Plank. Twisty: a multilingual twitter stylometry corpus for gender and personality profiling. In *Proceedings of the 10th Annual Conference on Language Resources and Evaluation (LREC 2016)/Calzolari, Nicoletta [edit.]; et al.*, pages 1–6, 2016.

[45] Willy, Steven Bird, Edward Loper, Joel Nothman, and Arthur Darcet. Punkt Tokenizer. https://www.nltk.org/_modules/nltk/tokenize/punkt.html.

[46] Ahmet S Yayla and Anne Speckhard. Telegram: The mighty application that isis loves. *International Center for the Study of Violent Extremism*, 2017.