UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE REDES Y SERVICIOS TELEMÁTICOS

TRABAJO FIN DE MASTER

Development of a Neologism Detection System based on Epidemic Models

PABLO RUBIO NOGUERA
2023

TRABAJO DE FIN DE MASTER

Título:	Desarrollo de un sistema de detección de neologismos basado en modelos epidémicos
Título (inglés):	Development of a Neologism Detection System based on Epidemic Models
Autor:	PABLO RUBIO NOGUERA
Tutor:	CARLOS ÁNGEL IGLESIAS FERNÁNDEZ
Ponente:	PONENTE
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 MASTER

Development of a Neologism Detection System based on Epidemic Models

JUNIO 2023

Resumen

La capacidad de conocer la forma en la que nos comunicamos e interactuamos con la gente que nos rodea es fundamental en los tiempos en los que vivimos. Gracias a todas las tecnologías que tenemos a nuestra disposición en el día a día, somos capaces de conocer a una gran variedad de gente de cualquier parte del mundo, lo que nos posibilita conocer cómo viven, qué cultura tienen y cuál es su forma de hablar y de relacionarse con el mundo.

Ha medida que vamos cambiando como sociedad y van surgiendo nuevos hábitos, nuestra fora de hablar cambia, dando lugar a la creación de nuevas palabras con significados completamente nuevos, que se introducen en nuestra lengua. Estas nuevas palabras se las llama neologismos.

Como se ha mencionado, conocer estas nuevas palabras es fundamental, por lo que han surgido maneras de detectar neologismos para estar en la vanguardia de este sector. La manera de detectarlas se basa principalmente en usar técnicas de procesamiento de lenguaje natural. Sin emabrgo, ha surgido una nueva vertiente, en cual se detectan los neologismos por su tendencia de uso.

Este proyecto consiste en detectar neologismos utilizando esta última técnica, la cual consiste en ser capaces de ajustar los datos de popularidad de un neologismo según un modelo epidemiológico llamado SIR. Se usa este modelo, ya que se utiliza para expresar de forma matemática la propagación de una infección, la cual comienza con muy poca gente infectada, pero a medida que pasa el tiempo el número de infectados aumentan, hasta que finalmente se acaban recuperando y finaliza la infección. Esto mismo se puede llevar al campo de los neologismos, los cuales muy poca gente utiliza al prinicipio, pero con el tiempo van crecienco en popularidad, hasta que llegado un punto la gente deja de utilizarlos.

Palabras clave: Neologismos, modelo SIR

Abstract

The ability to know how we communicate and interact with the people around us is fundamental in the times we live. Thanks to all the technologies we have at our disposal on a daily basis, we are able to meet a wide variety of people from all over the world, which allows us to know how they live, what culture they have, and how they speak and relate to the world.

As we change as a society and new habits emerge, our way of speaking changes, leading to the creation of new words with completely new meanings which are introduced into our language. These new words are called neologisms.

As mentioned, knowing these new words is essential, so ways of detecting neologisms have emerged to be at the forefront of this sector. The way to detect them is mainly based on the use of natural language processing techniques. However, a new approach has emerged in which neologisms are detected by their usage trend.

This project consists of detecting neologisms using the latter technique, which consists of being able to adjust the popularity data of a neologism according to an epidemiological model called SIR. This model is used because it is used to mathematically express the spread of an infection, which starts with very few infected people, but as time goes by, the number of infected people increases, until finally they end up recovering and the infection ends. The same can be taken to the field of neologisms, which very few people use at first, but over time they grow in popularity, until at a certain point people stop using them.

Keywords: neologisms, SIR model

Contents

List of Figures

CHAPTER 1

Introduction

This chapter will introduce the context of the project, including a brief overview of all the different parts that will be discussed in the project. It will also break down a series of objectives to be achieved during the realization of the project. Moreover, it will introduce the structure of the document with an overview of each chapter.

1.1 Context

Nowadays, people are constantly changing the way they talk and communicate. Thanks to technological advances, we have the ability to interact with many more people than we would have been able to in the past, and as a result, we are able to acquire new knowledge about people from all over the world. All this interaction means that we have a richer vocabulary and are able to express the same thing in different ways. Because of all this, in the Spanish language, new words have emerged that are the result of words from other languages, either they are already existing words but are given a different meaning or, finally, words that have been created by combining others. All these new words are called "neologisms".

As you can see, knowing these neologisms is extremely important, since thanks to them we will be able to reach people easily while interacting with them, due to the great use that is given to these neologisms. Therefore, studies and technological research on the detection of these neologisms have emerged to be at the forefront of communication.

Most of these studies are focused on detecting these neologisms by using natural language processing techniques. However, there exists a Chinese article [13], which tries to analyze and detect these words using another point of view. It is based on detecting neologisms according to a contagion model, in which people begin to transmit these words, acquiring greater popularity as more people use them, until finally, the neologism loses interest and, therefore, decreases in terms of popularity. This contagion model can be also called *epidemic model*, due to this is exactly what happens with an epidemic, only that instead of spreading words, an infection is transmitted.

This kind of neologisms detection is what this project will do, by following certain guidelines when carrying out this detection process, such as the way we select or discard a neologism or the kind of algorithm we implement to get accurate results.

1.2 Project goals

The main objective of this project will be to implement a process capable of fitting the trend of use of a neologism to an epidemic model as best as possible; thus, it will be possible to state that a priori a Spanish neologism can be detected through the growth or decline of its popularity. Other objectives arise from this one, which are mentioned as follows:

- Establish guidelines by which neologisms are selected or discarded for this project and why.
- Reason which sections of the neologism are useful to process to meet the main objective, in terms of popularity or usage of the neologism.
- Implement an algorithm capable of processing and analyzing neologism data to fit it to an epidemic model.
- Set up an automatic process for this algorithm.

Lastly, once we achieved the goals above, there will be a final goal in which we will be able to index the results we got into the elastic stack, in order to have a much better product, which clients would be able to work with.

1.3 Structure of this document

The remaining of this document is structured as follows:

- ***Chapter 2***: We will talk about the different studies or technologies we have used or followed, in order to achieve our project goals.
- ***Chapter 3***: We will explain the methodology and architecture implemented in this project and the different stages we have developed.
- ***Chapter 4***: We will show different examples of neologisms used in the project and its results for each stage.
- ***Chapter 5***: We will comment on whether we have achieved our purposed goals and what future implementations could be made to this project to improve it.

CHAPTER 2

State of Art and Enabling Technologies

Introduction to state of art and enabling technologies

This section describes about all the technologies and studies that have been used in this project or are related to it. In particular, we discuss the following:

- Neologisms detection
- Epidemic model
- Tools and technologies

2.1 Neologism detection

This project consists in detecting Spanish neologisms using an epidemic model. As is said before, a neologism is a word already existing in a language, but which acquires a new meaning, or a word created by combining other words or coming from another language. There exist many ways or techniques for detecting these neologisms, and in this section we are going to talk about some of them.

One of the best ways to detect a neologism is to use natural language processing. Natural language processing (NLP) is an interdisciplinary subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. The goal is a computer capable of “understanding” the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves [28].

Some NLP techniques used in neologisms detection are:

- *Tokenization*: Tokenization is a simple process that takes raw data and converts them into a useful data string. It is used in natural language processing to divide paragraphs and sentences into smaller units that can be assigned a more easily meaning [2].
- *Lemmatization*: Lemmatization is another technique used to reduce inflected words to their root word. It describes the algorithmic process of identifying an inflected word’s “lemma” (dictionary form) based on its intended meaning [19].
- *Stemming*: Stemming is a technique used to reduce an inflected word to its stem. For example, the words ‘programming’, ‘programmer’, and ’programs’ can be reduced to the stem of the common word ’program’. In other words, the word “program” can be used as a synonym for the three previous inflection words [19].

- *Part-Of-Speech tagging*: Part-of-speech (POS) tagging is the process of labeling words in a text with their corresponding parts of speech in NLP. It helps algorithms understand the grammatical structure and meaning of a text [9].

A web-based tool called *NeoTrack* developed by Maarten Janssen [11], uses some of these techniques, such as tokenization, in combination with others for detecting neologisms. In particular, the method by which it is able to detect them is called semiautomatic neologism detection. As shown in Fig. 2.1, NeoTrack uses an *exclusion based neologism candidate extraction*, in which there is a list of known words called *exclusion list* which does not contain any neologism within, and compare whether the words of the tokenized text are in that list or not. If they are not inside the list, then they are categorized as neologisms candidate, and a human will decide if those candidates are really neologisms or not. All candidates classified by the human as not neologisms will be added to the exclusion list.

This method is called computer-aid research and its advantage is that it becomes possible to really establish which words are new by comparing the new text to the collection of all the text in a reference corpus.

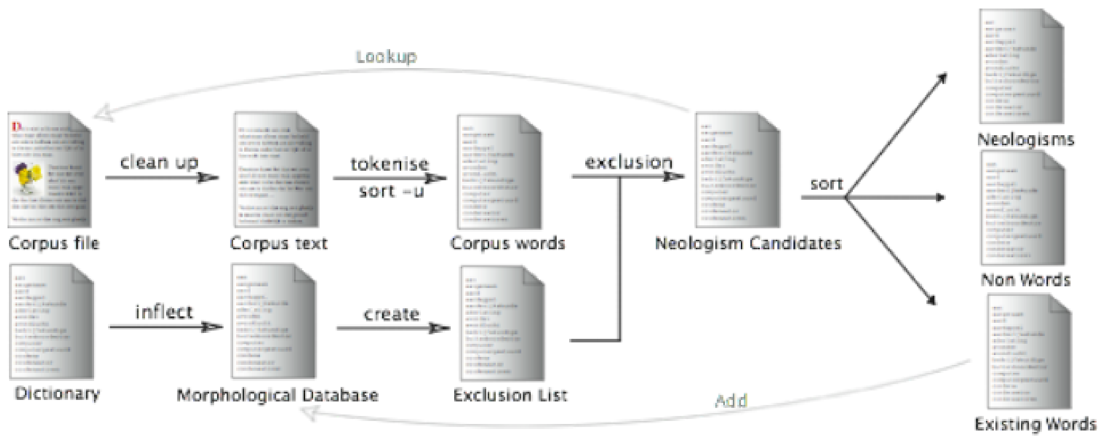


Figure 2.1: NeoTrack flow-chart [11]

Another technique, also developed by Maarten Janssen, is called NeoTag [12], which describes a modified POS tagger that explicitly considers new tags for known words, thus making it a better fit for neologism research. A neologism can be an existing word with a different meaning from the original. For that reason, the grammatical category will also be different. In terms of POS tagging, this means words that are known in the training corpus but are used in a different POS in the text to be tagged.

NeoTag is a n-gram tagger that calculates the most likely POS tag for each word. Also, it is able to lemmatize (providing a lemma or citation form for each word in the corpus) either for and unknown words. This particular feature is very interesting because most lemmatizers are unable to lemmatize unknown words, and this lemmatizer can do it. It simply lemmatizes using the citation form of similar words. In addition, it forces the system to consider POS tags for known words that were not evidenced by the training corpus.

In principle, NeoTag is not limited to grammatical neologism, but it can detect formal neologisms as well: When running NeoTag over a corpus, it not only marks off grammatical neologism candidates, but it also more trivially detects formal neologism candidates (like the method explained previously).

2.2 Epidemic model

Once has been discussed about neologism detection and different algorithms to achieve it, it is time to talk about the model it is used to fit a neologism trend to predict its evolution.

The technologies described previously show how to detect a neologism using especially language processing techniques, such as tokenization, lemmatization, and so on. However, this is a completely different approach. According to [13], neologisms have a distinguished trend curve, which starts with a faster uphill slope, leading to an inflection point where the curve starts descending until it converges to some point. For this reason, that study decided to verify if an epidemic model called the SIR model will fit this curve, that is, the trending neologism curve has similarities with a epidemic model approach.

The following lines will explain this epidemic model in detail.

2.2.1 SIR model

This model mathematically describes how an infection spreads over a population over time [21]. For doing this, it divides the population in three groups; susceptible, infected and recovered.

- *Susceptible (S)*: group of people who are vulnerable to be infected.
- *Infected (I)*: group of people within the infection that can transmit it to others. It can take a few days until they get infected before they can start transmitting.
- *Recovered (R)*: group of people who have already passed the infection and cannot transmit it anymore.

For simplicity, we will consider the total population N as N = I + S + R.

In Fig. 2.2 is shown the flow among these groups, but the infection will be a neologism in terms of popularity.

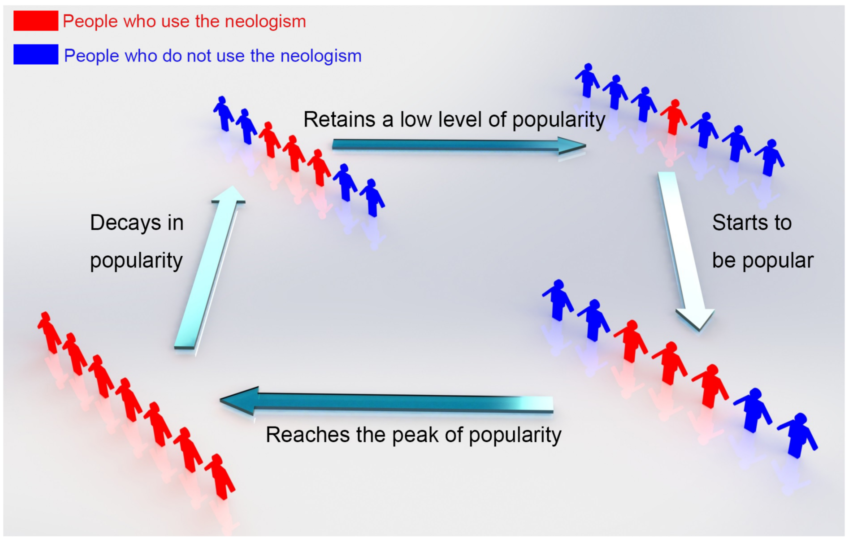

🟥 People who use the neologism
🟦 People who do not use the neologism
Retains a low level of popularity
Starts to be popular
Reaches the peak of popularity
Decays in popularity

Figure 2.2: Epidemic model [13]

2.2.2 SIR model equations

When an infection appears in a population (in terms of epidemic), it usually spreads really quickly, and the number of infected compared to susceptible or recovered increases a lot.

Infected come from susceptible people, who get infected when they come in contact with an infected person. At some point before the epidemic starts, infected people will stop transmitting the infection to the recovered group. Therefore, susceptible people will turn infected and eventually recover.

Now we will explain mathematically how this model is described.

There is a population of N, a probability of being infected of 20% and an average number of 5 people that an infected person will be in contact with per day. Consequently, an infected person will infect 1 person per day (5*20%). This is called as β (“beta”), the expected amount of people an infected person infects per day.

Let us consider now that an infected person can spread and infect people for 7 days (this is called D). This means that in a week, an infected person will be able to infect 7 other people (always referring to the susceptible group). This number will be called R_0 and is the result of multiplying *beta* with D ($R_0 = \beta * D$. Instead of talking about D, we will create another variable called γ (”gamma”), which is the inverse of D (1/D). This variable means the recovery ratio or the proportion of infected recovering per day. For example, if currently 30 people are infected and D = 3 (so they are infected for three days), then per day, 1/3 (so 10) of them will recover, and hence $\gamma = 1/3$. With $\gamma = 1/D$, so $D = 1/\gamma$ and $R_0 = \beta * D$ it follows that $R_0 = \beta/\gamma$.

It is difficult to obtain a formula for each group at a time t (S(t), I(t), and R(t)).

However, it is really easy to calculate the change of these groups per unit of time. The names of each group to describe this change will be S'(t), I'(t), and R'(t).

The change from time t to time t + 1 in the susceptible group will be the number of people who pass from this group to the infected. Using the variables defined above, we obtain Eq. 2.1.

$$S'(t) = -\beta * I(t) * S(t)/N \tag{2.1}$$

Why is this S(t)*N in the equation? Well, let say $\beta = 1$ N=100, I=60 and S=30 at a time t (obviously R=10). At the next time t + 1, the susceptible will have to be 60 less ($I*\beta = 60$, but there are only 30 people, so the susceptible people will be $S(t{+}1) = S(t){-}\beta{*}I{*}S/N = 12$ people.

Having a look now at the infected group, the value of I'(t) will be the number of new infected people (calculated previously) minus the number of new recoveries, leading to the following equation:

$$I'(t) = \beta * I(t) * S(t)/N - \gamma * I(t) \tag{2.2}$$

Finally, as mentioned before, R'(t) will be the number of new recoveries from the infected group:

$$R'(t) = \gamma * I(t) \tag{2.3}$$

These S'(t), I'(t), and R'(t) variables in mathematical terms are differential equations, so the final equations of the SIR model are:

$$\begin{cases} \frac{dS(t)}{t} = -\beta * I(t) * S(t) \\ \frac{dI(t)}{t} = \beta * I(t) * S(t) - \gamma * I(t) \\ \frac{dR(t)}{t} = \gamma * I(t) \end{cases} \tag{2.4}$$

where now the variable β is the original value multiplied by N, and thus N is eliminated in the final equations.

Finally, Fig. 2.3 illustrates an example of how these groups increase or decrease their population during an epidemic. These variations are shown by three curves, one for each group.

As we can see, at the beginning of the epidemic, all people are susceptible to being infected, except one person who is infected. As the days go by, the susceptible people begin to decrease because they become infected or have just been recovered, until some point when all people are recovered, and the epidemic ends. In this case, due to the low value of β, the infected do not increase too much, but there will be cases in which this group can rise exponentially.

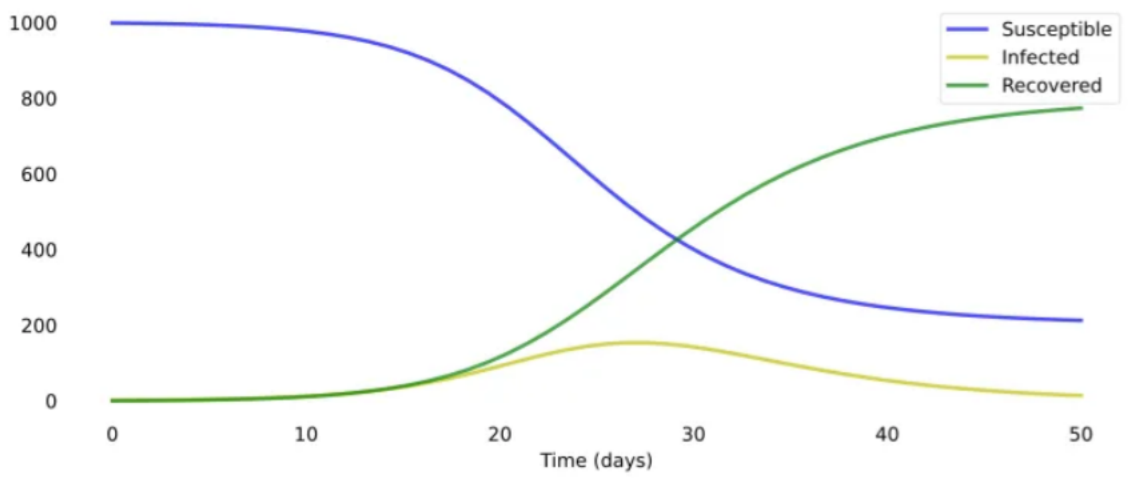


Figure 2.3: SIR curves [7]

2.3 Tools and technologies

In addition to all the theory and research on neologisms and epidemic models that this project requires, it is also necessary to know about different tools, in order to replicate the Chinese model [13], but with Spanish neologisms. Therefore, this section includes a brief explanation of the main tools and technologies used in the project.

2.3.1 Noise reduction

As mentioned above, neologisms describe a trend curve shaped similarly to an epidemic model. The problem with this curve is that it will require smoothing it to implement a model-related fitting algorithm due to the emergence of various peaks that are considered noise. For this reason, some noise-reducing methods will be explained below.

The first method consists of defining an odd-size window whose size refers to the number of discrete points of the curve that can be packaged at once (adjacent points). That window will start at the beginning of the curve, and for each group of discrete points, it will calculate the average for all the values in the window and replace those values with the calculated.

The second method is similar to the previous one. In this case, there is a convolution operation among the window and the adjacent points selected. Let us say that there is a window defined as an array $WinArray = [3, 5]$ and an array of adjacent points $DisArray = [1, 2, 4, 7, 9]$. The convolution operation will be the result of doing the following calculation:

- First element: $5 * 0 + 3 * 1 = 3$
- Second element: $5 * 1 + 3 * 2 = 11$
- Third element: $5 * 2 + 3 * 4$
- Fourth element: $5 * 4 + 3 * 7$
- Fifth element: $5 * 7 + 3 * 9$

The final result is the array $FinArray = [3, 11, 22, 41, 62]$

The third method is to use a Savitzky–Golay filter, a digital filter that has the mission of increasing the precision of the data without distorting the signal trend, as shown in Fig. 2.4. This is achieved, again using the convolution, by fitting successive subsets of adjacent data points with a low-degree polynomial using the method of linear least squares.

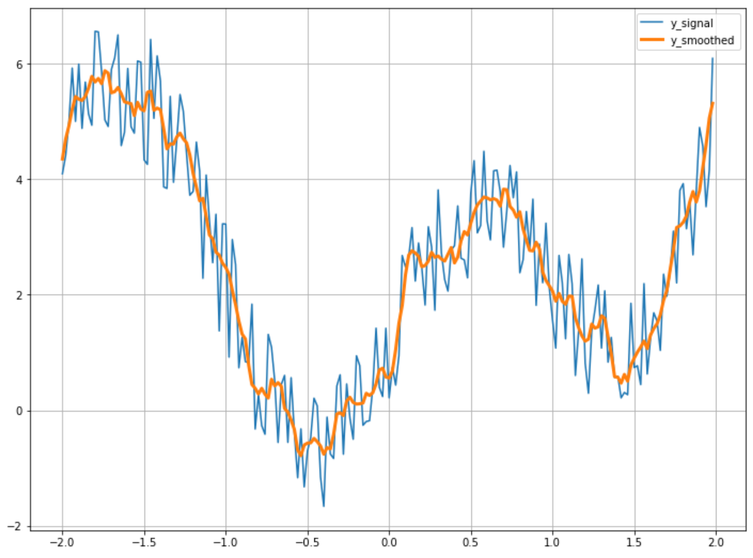


Figure 2.4: Savitzky-Golay filter

2.3.2 Google Trends

The application where the neologisms have been extracted is Google Trends. Google Trends was created by Google and is designed to see the topics people are or are not following, practically in real time. Journalists can use this information to explore potential story ideas and can also feature Trends data within news stories to illustrate a general level of interest in, say, a political candidate, social issue, or event.

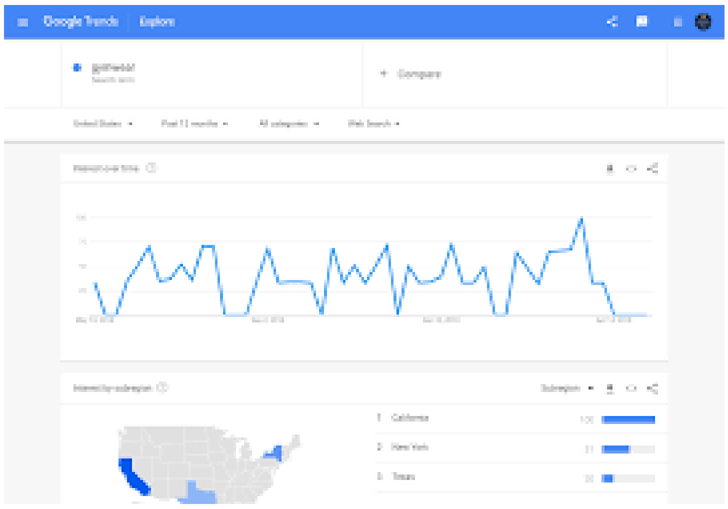


Figure 2.5: Google Trends

The Google Trends homepage (google.com/trends) (Fig. 2.5) features clustered topics that Google detects are related and trending together on either Search, Google News, or YouTube. Trending stories are collected based on Google's Knowledge Graph technology, which collects search information from these three Google platforms to detect when stories are trending based on the relative increase in volume and the absolute volume of searches.

2.3.3 Pyhton Libraries

In this area we will talk about the different Python libraries that have been used during this project.

2.3.3.1 Scikit-learn

Scikit-learn or Sklearn [31] is a free software machine learning library for the Python programming language. It features various classification, regression, and clustering algorithms, including support vector machines, random forests, gradient boosting, k-means, and so on, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

It started by the French David Cournapeau during the project a Google Summer of Code in June 2007. In 2010, contributors Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort and Vincent Michel, from the French Institute for Research in Computer Science and Automation in Saclay, France, took leadership of the project and released the first public version of the library on February 1, 2010.

The purpose of using this library during the project is to preprocess the data, in order to have them on the same scale and with the same structure, and thus all the following stages of the project can be defined generally for all types of data.

2.3.3.2 Scipy

Scipy [32] is a free and open source Python library that is used for scientific and technical computing created by Travis Oliphant, Pearu Peterson, Eric Jones in 2001. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, and other common tasks in science and engineering. The ones that we are using in this project are:

- *Integrate* [29]: Calculating the numerical value of a definite integral and describing the numerical solution of differential equations.
- *Signal*: Signal processing tools. We are using it in order to calculate prominences and peaks inside a curve.

2.3.3.3 Numpy

Numpy [30] is an open source software library for the Python programming language, which adds support for large multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications.

We use this library because other libraries used also in this project, such as pandas, Scipy, or Sklearn, are able to use the Numpy multidimensional arrays, and hence we do not need to make any data transformations while using these libraries. In addition, in this project, we apply different filtering techniques, and some of these are provided by Numpy.

2.3.3.4 Matplotlib

Matplotlib [10] is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Pyplot is a Matplotlib module that provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, and the advantage of being free and open source [27].

All the chart illustrations of this project related to final results, preprocessing or filtering, have been created with this tool. In fact, Matplotlib has the ability to combine different curves in one figure, in order to compare different results at once.

2.3.3.5 Pandas

Pandas [26] is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. The aim of this is to be the fundamental high-level building block for performing practical real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any language. It is already well on its way to this goal.

In 2008, pandas development began at AQR Capital Management. By the end of 2009 it had been open-sourced, and is actively supported today by a community of like-minded individuals around the world who contribute their time and energy to help make open-source pandas possible.

It is used for creating data structures called DataFrames, that allow us to use different tools (provided also by Pandas), in order to analyze and process data in different ways, such as, verifying if there exist null values or transforming data into new data types.

2.3.3.6 Lmfit

Lmfit [16] provides a high-level interface to nonlinear optimization and curve fitting problems for Python. Lmfit builds on the Levenberg-Marquardt algorithm of scipy.optimize.leastsq(), but also supports most of the optimization method from scipy.optimize. Initially inspired by (and named for) extending the Levenberg-Marquardt method from scipy.optimize.leastsq, lmfit now provides a number of useful enhancements to optimization and data fitting problems.

The package is very useful because thanks to it we are able to optimize and fit our neologisms data curves with the SIR model by varying different parameters and finding the solution that better fits to them.

2.4 ElasticSearch and Kibana

Elasticsearch [24] is a free and open distributed search and analytics engine for all data types, including textual, numeric, geospatial, structured and unstructured. Elasticsearch is developed from Apache Lucene and was first introduced in 2010 by Elasticsearch N.V. (now known as Elastic). Known for its simple REST APIs, distributed nature, speed, and scalability, Elasticsearch is the core component of the Elastic Stack, a set of free and open tools for data ingestion, enrichment, storage, analysis, and visualization.

Kibana [25] is a free and open front-end application that sits on top of the Elastic Stack and provides data visualization and search capabilities for data indexed in Elasticsearch.

Kibana also acts as the user interface for monitoring, managing, and securing an Elastic Stack cluster; as well as a centralized hub for the integrated solutions developed on Elastic Stack. Developed in 2013 in the Elasticsearch community, Kibana has become the window to the Elastic Stack itself, providing a portal for users and enterprises.

CHAPTER 3

Architecture and Methodology

This chapter presents the architecture and methodology used in this work. It describes the overall architecture of the project, with the connections between the different components involved in the development of the project.

In this section, we describe the architecture and methodology of this project and its different steps or stages. Fig. 3.1 illustrates what the shape of the architecture of the project, in which we *extract the neologisms from Google Trends*, *insert them manually into our neologism detection process*, and finally *index the results* through an API into our ElasticSearch engine to visualise the results using Kibana. If we focus on the methodology, i.e. the neologism detection process, the Fig. 3.2 shows it in detail, in which each process (rectangular blocks) corresponds to one stage.

First, we have to *select a neologism* with which we want to work. Then, we need to observe in Google Trends if the neologism trend follows the predefined statements (we will detail them below) and if not, discard it and select a new one. Next, we take the neologism data (the Google Trends curve data), *analyze and process* them, in order to have good data for the following stages. Once the data are preprocessed, it is time to smooth the curve by *removing all the noise and disturbances* due to the fact that we only need the curve tendency and not all its information. Now, we *select a curve segment* that we want to fit to the SIR model, we *fit* it, and we obtain the results.

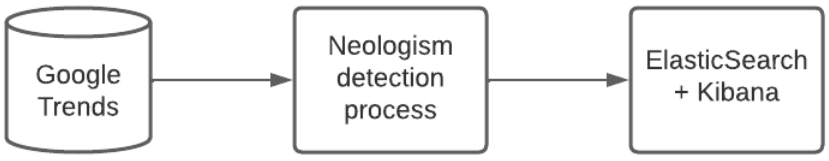


Figure 3.1: Final project architecture

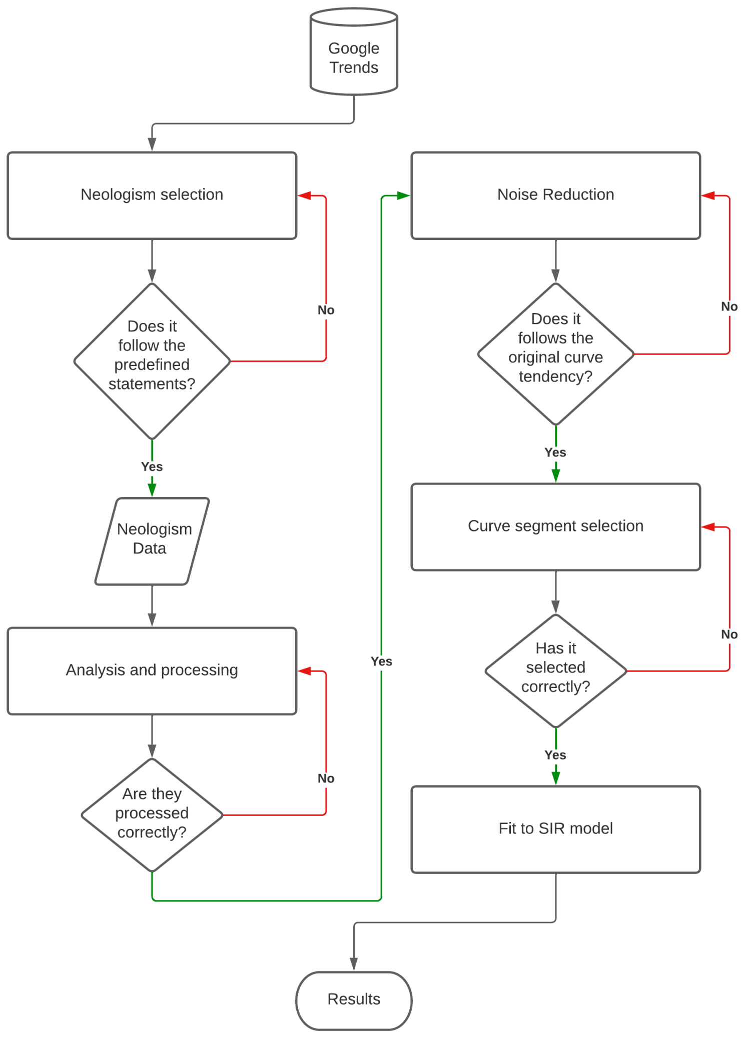


Figure 3.2: Methodology flow diagram

3.1 Neologism selection

The selection of neologisms is the first filter we use for this study. Thanks to this stage, we are able to decide which neologisms have the appropriate features for the development of this project.

The first step will be to select some words that have already been considered neologisms from an article, journal, or website. In this case, we extract these words from a linguistic dissemination article called "Martes Neológico", an article launched in October of 2015 by the "Observatori de Neologia" of the group "IULATERM" of the university "Universitat Pompeu Fabra" and the "Instituto Cervantes". Each Tuesday, they offer grammatical and semantic information about new neologisms. Also, they include information about the origin of the word, frequency of occurrence, and more. For "Martes Neólogico", these neologisms are considered words that are used by the population and documented by the press, but which are not included in the dictionary of the Royal Spanish Academy.

As part of the explanation, we will illustrate all the information that appears for a single neologism. This neologism will be "*Metaverso*", whose description mentioning it is written by Carlos Ruiz Fernández of Pompeu University in Fabra, Spain [6].

This term was coined by science fiction writer Neal Stephenson, who first used it in his 1992 novel Snow Crash, where he described a virtual environment in which people interact through avatars.

As for the formation of this neologism, we see that it is an acronym formed by the union of two elements: the first is the compositional element meta-, which comes from the Greek and means, among other things, 'after' or 'next to', while the second is -verse, which is the ending of the word universe. With this construction, Stephenson wanted to emphasize what would come after the Internet.

Today, most of the neologisms used by the population are generated primarily on the Internet. Due to all the fashions and the great changes that we live each day, our way of life is constantly changing, and for that reason our way of communicating changes too. In fact, there are many neologisms that have a high rise in the first weeks/months since its creation, which it transforms in many researches, news, or posts, by thousands or millions of people. Then, maybe because of the appearance of a new fashion or trend, or just because people get bored of these words, the neologisms which were rising start to decay and converge to some point where maybe people do not use it anymore or still use it but with less enthusiast (and in the future could rise again). This conduct is reflected in Fig. 3.3 extracted from Google Trends with the neologism "*aporofobia*" (is a term used to describe a feeling of rejection of the poor, the helpless, the one who lacks outlets, the one who lacks means or resources [20]).

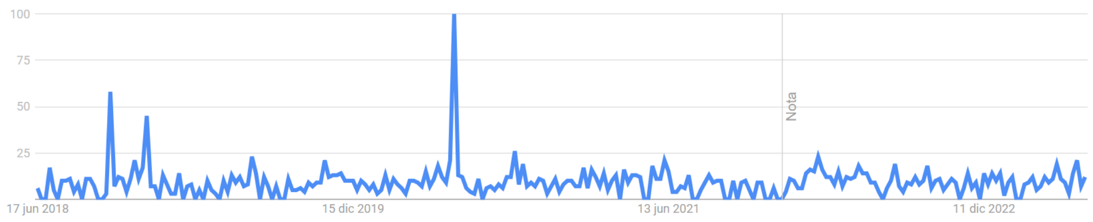


Figure 3.3: "Aporofobia" rise-decay behavior

However, there are many neologisms that appear or disappear according to the stage of the year in which we find ourselves. For example, there are words that are most often used in Christmas or Summer like "Noel" or "beach". In the case of neologisms, an example of this would be "*Semana Blanca*" (The holiday period in compulsory Spanish education is supposed to be devoted to skiing or other sports practiced on snow [18]) whose appearance tendency is illustrated in Fig. 3.4.

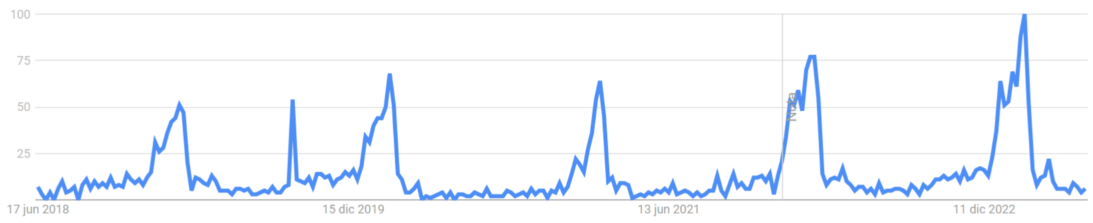


Figure 3.4: Semana Blanca appearance tendency

Finally, there are neologisms that maintain their usage trend, and therefore do not fit any of the patterns we have seen above. Fig. 3.5 shows this behavior with the neologism "*tetris*"(a situation involving several elements that have to be fitted tightly together [17]).

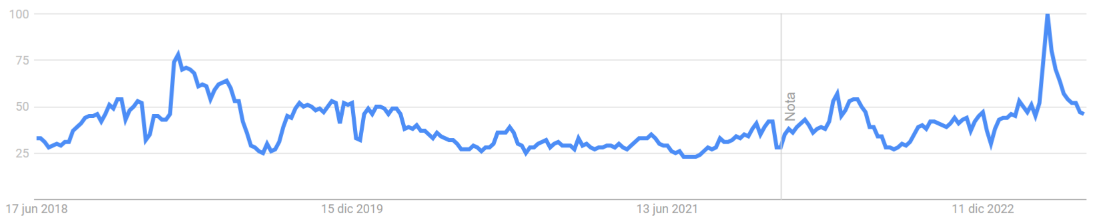


Figure 3.5: Tetris tendency behavior

Once we have seen different behaviors of the neologisms in terms of tendency of use, we will define the statements on which we are going to support in order to select or discard a neologism for this project.

1. We will select those neologisms that follow the behavior in Fig. 3.3 or Fig. 3.4 and

discard those which follow Fig. 3.5

2. If a neologism do not have enough data provided from Google Trends we will discard it.

3. If the patterns of neologisms show only ascending trends, we will discard them.

Fig. 3.6 is an example of Statement 3, in which for neologism "*chatbot*" (a computer program designed to simulate a conversation with a human user, usually over the Internet; especially one used to provide information or assistance to the user as part of an automated service [15]), even though there is no ascending trend, there is no convergence point yet where the neologism stabilizes.

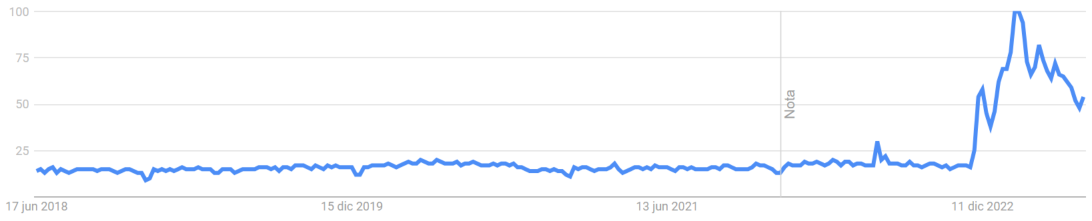


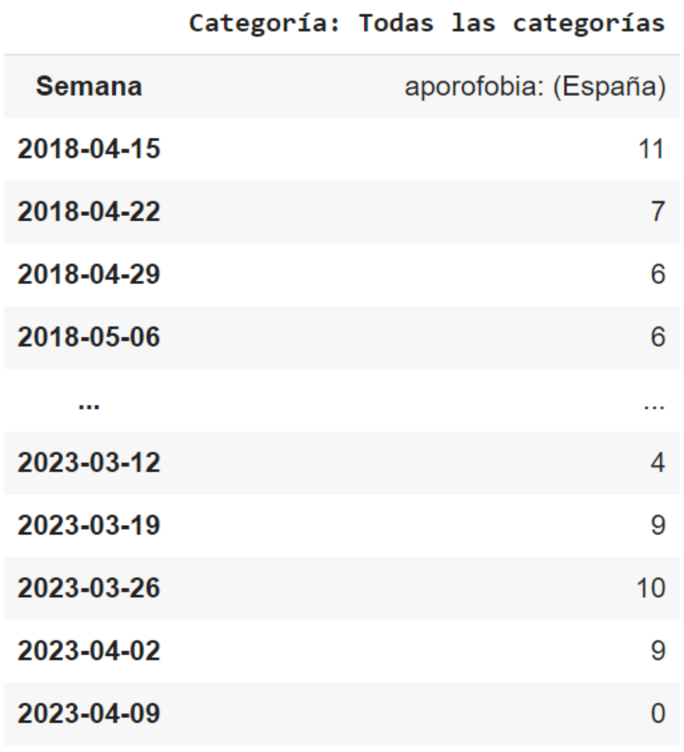
Figure 3.6: Chatbot tendency

3.2 Analysis and Processing

This section will talk about the transformations that we must do to have the neologism data downloaded from Google Trends in the correct format, so that we can operate easily with them in the following stages. Before continuing to explain the procedure, it is important to emphasise that we will only download neologism data that we consider selected or valid by following the rules or statements explained above. In addition, we decided to download data from the last 5 years to have enough information about the neologism. Note that the downloaded data refer to the charts seen above, where the X axis represents the time axis from which the trend of the neologism has started to be measured and a Y axis that represents the popularity it had for a specific date. These values have a range from zero to one hundred, so a value of one hundred indicates maximum popularity and zero indicates minimum popularity.

The first step will be to install the library that will be used to analyse and process the data. As we decide to use Python as the programming language for this project, we select the Pandas library to perform all this analysis and process because it is one of the most famous libraries for doing this kind of work, and therefore there is a lot of documentation that will help us. The type of structure that we manage once we start analysing the neologisms data is called DataFrame, which is a two-dimensional, size-mutable, potentially heterogeneous tabular data.

Once we have Pandas installed, the next step will be to see what the format of these data looks like once we have downloaded it from Google Trends. The format for all the neologisms is illutrated in Fig. 3.7, which is the data for the "*aporofobia*" neologism.

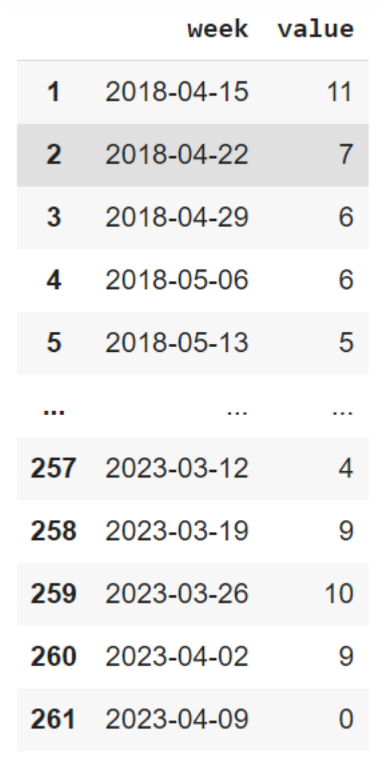


	Categoría: Todas las categorías
Semana	aporofobia: (España)
2018-04-15	11
2018-04-22	7
2018-04-29	6
2018-05-06	6
...	...
2023-03-12	4
2023-03-19	9
2023-03-26	10
2023-04-02	9
2023-04-09	0

Figure 3.7: "Aporofobia" raw data

As we can see, the DataFrame consists of a column called "Category: All categories", which is made up of two other internal columns, one column for the instant times where the popularity values are gathered (they are always separated in weeks), and the other for the value itself. Therefore, the first step will be to transform this DataFrame formed by a column into a new one consisting of the two internal ones.

When we already have the DataFrame we need, we check if there is any value for any time instant in which the popularity value does not exist, along with an inspection of the type of data that each of the two columns has. Although there exists a popularity value for each week for all the neologisms selected, there is a problem with its format. Both columns have as data type the so-called 'object' which is an inappropriate type to operate with. For that reason, we have to transform this 'object' type into 'datetime' in the case of the week column, and into an 'integer' type for the value itself. Finally, in Fig. 3.8 the result of this whole process is represented upon completion.



	week	value
1	2018-04-15	11
2	2018-04-22	7
3	2018-04-29	6
4	2018-05-06	6
5	2018-05-13	5
...	...	...
257	2023-03-12	4
258	2023-03-19	9
259	2023-03-26	10
260	2023-04-02	9
261	2023-04-09	0

Figure 3.8: “Aporofobia” processed data

3.3 Noise Reduction

At this stage we have data about the popularity of a neologism, which have been processed in the previous step and which in turn come from Google Trends after being selected according to whether or not they met the predefined statements. Fig. 3.9 shows a chart of “*aporofobia*” data after the previous processes. Note that this trend chart must be the same as Fig. 3.3, but instead of being from Google Trends, we have created it from the processed data and using the Python library Matplotlib.

As we said previously, this kind of trend has a big rise at some point and then decays to low levels of popularity. However, as we can see in Fig. 3.9, these low levels of popularity can be considered as noise, because there is such a big difference between the highest peak (maximum popularity) and the rest of the curve, and therefore, we just want to fit that great peak in future stages and ignore what is around it.

Another type of noise is illustrated in Fig. 3.10, which represents the trend curve for the neologism “*código QR*” (‘quick response code’, which designates a square two-dimensional dot matrix that can store encoded data [4]).

In this case, visually it is very clear where is the disruption point for this neologism over time, but if we try to extract this point of disruption through programming, we will realize

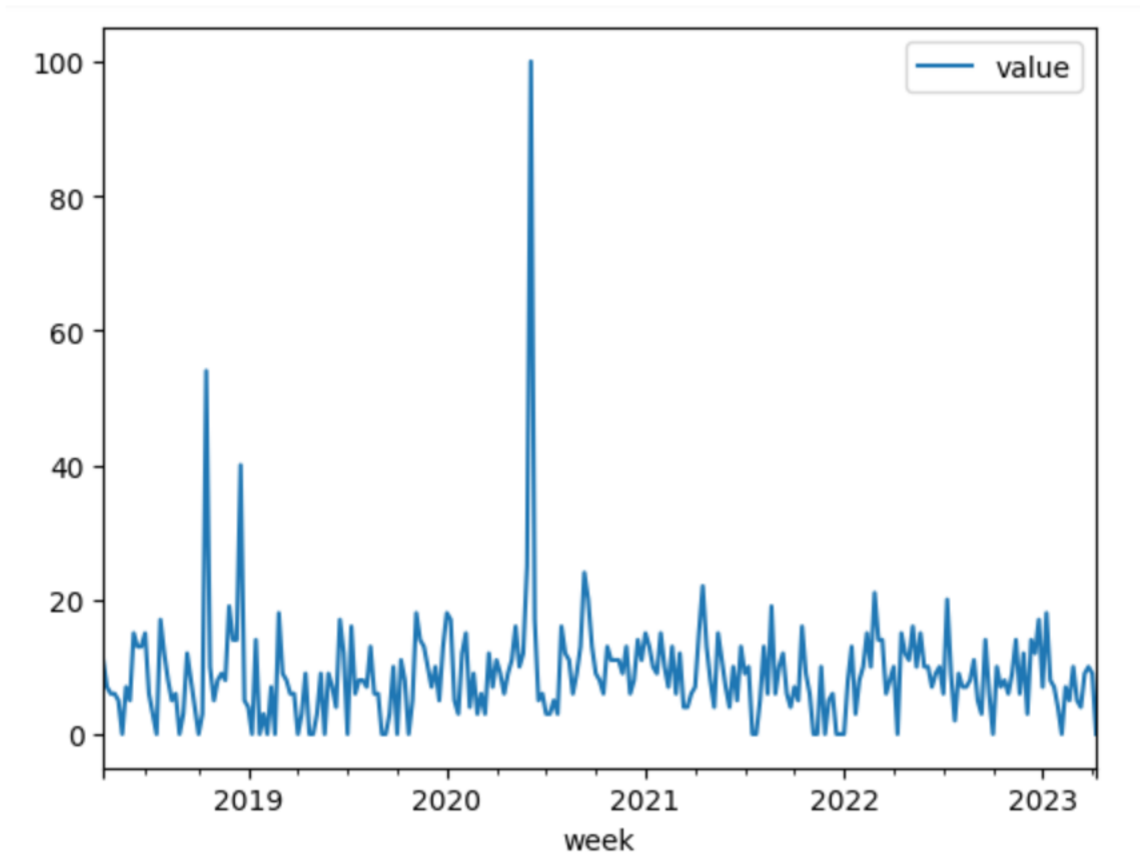


Figure 3.9: “Aporofobia” trend chart created with Matplotlib from processed data

that it is extremely difficult, unless some kind of new processing is done to the data.

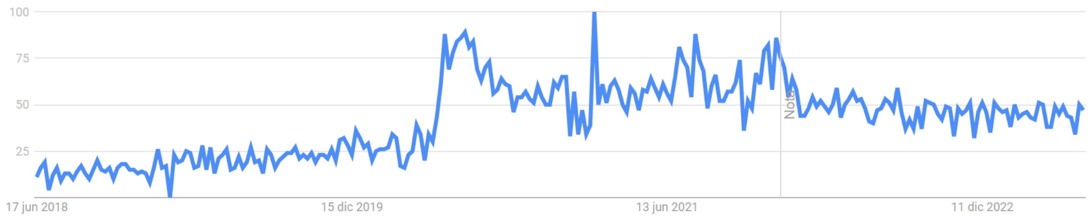


Figure 3.10: "Código" QR trend curve

The type of process that is required to be able to fit these data to the SIR model consists of smoothing the curve in order to keep just the tendency of the curve and not all the data. This process of smoothing, which is necessary to simplify the curve, adjusting the different points so that it is not so abrupt, has been called *noise reduction*.

This noise reduction method consists of applying a digital filter to the input data and getting a curve that keeps the tendency of the data but does not have all those impurities that we have called noise.

The digital filter that we will use in this stage, as was commented on Section 2.3.1, is called the Savitzky-Golay filter, which has the purpose of increasing the precision of the data without distorting the signal trend. For doing it, we have to take note of two parameters; the length of the window and the polynomial order. The length of the window refers to the size of the convolution window. The operation to smooth the curve consists of applying an operation inside a segment of the curve. That curve segment is the window of the digital filter and the operation inside the curve window is the convolution. So, the larger the window length, the more points will be affected by that convolution. The other parameter is the polynomial order, which refers to the order of the polynomial we are going to use to fit successive subsets of adjacent data points by the method of linear least squares (which is the method inside the convolution operation).

Finally, Fig. 3.11 illustrates the trend curve of "*Código QR*" before applying that filter, and Fig. 3.12 shows the result after this filter.

As we can see, there are some prominent peaks that have been removed after the filter, and hence the final figure is not exactly the original. However, this is not important because the tendency of the curve is still there. Those prominences that may appear not only in this neologism, but in many others, are considered as disturbances, because they are really short in terms of time and they appear and disappear really fast, so the filter do not consider them as relevant.

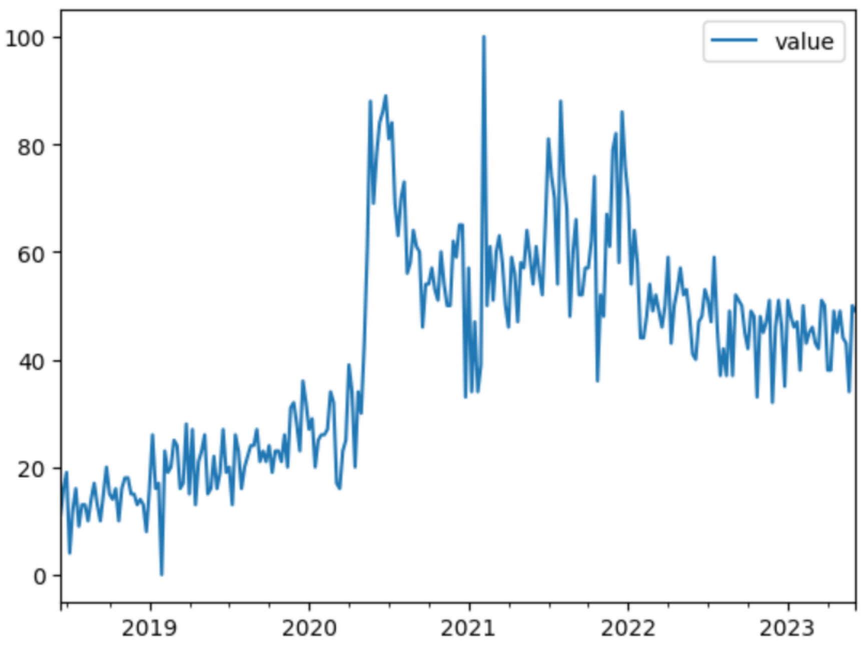


Figure 3.11: “Código QR” trend curve before filter

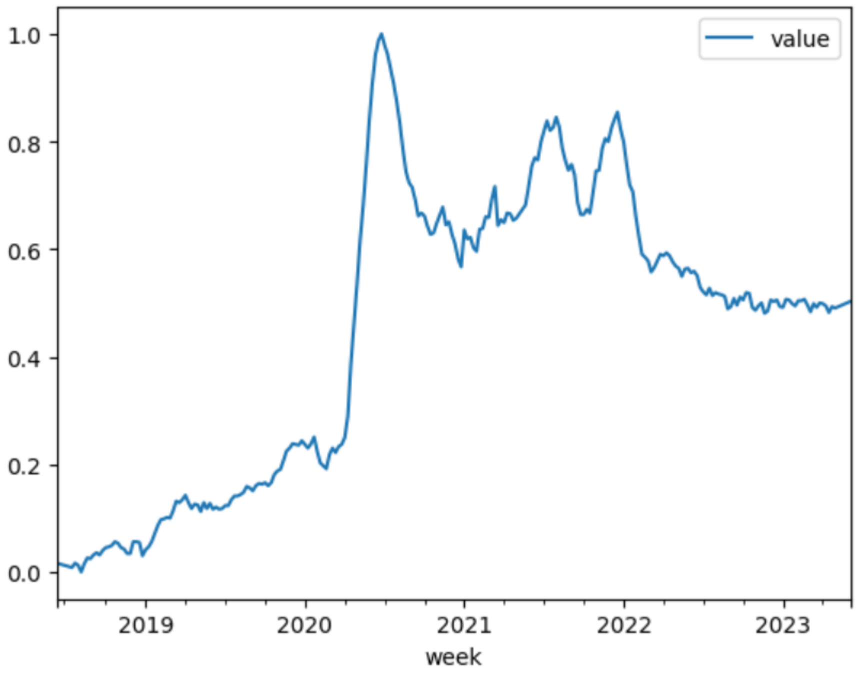


Figure 3.12: “Código QR” trend curve after filter

3.4 Curve Segment Selection

A selected neologism contains at least five years of popularity information, which is represented by a trend curve, as we have seen above. There will be segments of that trend curve where the popularity or usage of that neologism is too low, and others where there exists a disruption point, in which the curve starts to grow. In this section, our goal will be to select a curve segment of this graph which can be similar to the infected curve of the SIR model.

As we commented previously, the SIR model is made up of three curves, one for each group defined in the model. As a reminder, there are three groups; susceptible, infected, and recovered. The susceptible group corresponds to people who have not yet passed the infection and are capable of being infected. Infected people transmit the infection to the susceptible group. Finally, the recovered group refers to people who have already been infected but have passed the infection or are unable to transmit it.

As shown in Fig. 2.3 , each curve of the model has a different behavior. In our case, we are interested in the curve whose behavior resembles the behavior of a neologism, from the point of view of popularity.

As explained above, a neologism tends to have very little popularity at the beginning of its existence, and as time goes by, this popularity increases to a great level until it reaches a point where it begins to decline. Therefore, the curve we are interested in using for the case of a neologism is the infected curve.

Fig. 3.13 illustrates the neologism "*Semana Blanca*" once it has been processed and the filter applied in the previous stages.

Visually, it is very clear which parts of the curve resemble the infection curve of the SIR model. However, obtaining these segments by programming is a laborious task. It should be noted that any of the increments of popularity that this neologism has would be worthwhile for the study.

In order to obtain any of these segments, in which there is an increase in popularity, the following algorithm is established.

First, the values of the prominences of each peak of the curve are calculated, and we select the one with the highest value, since that peak of popularity reflects the highest rise of all the peaks. Fig. 3.14 shows the different prominences for each relative maximum of the "*Semana Blanca*" trend curve. In this case, the maximum prominence is at point 242, which is by chance the absolute maximum of the curve.

The second step consists of calculating the relative minimums at the left and right for that peak, which have been called "base peaks". This is needed because as in the infected curve, we want to know when the neologism starts to grow and when it finishes to descend,

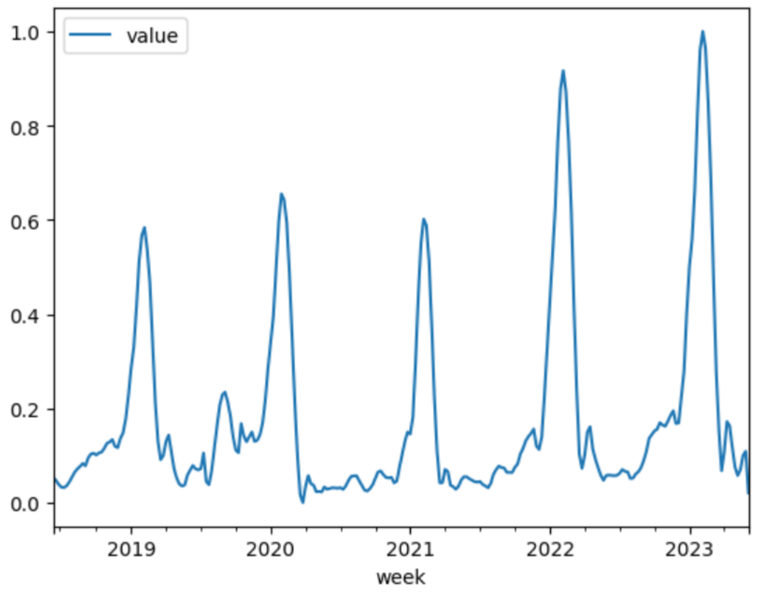


Figure 3.13: “Semana Blanca” trend curve after filter

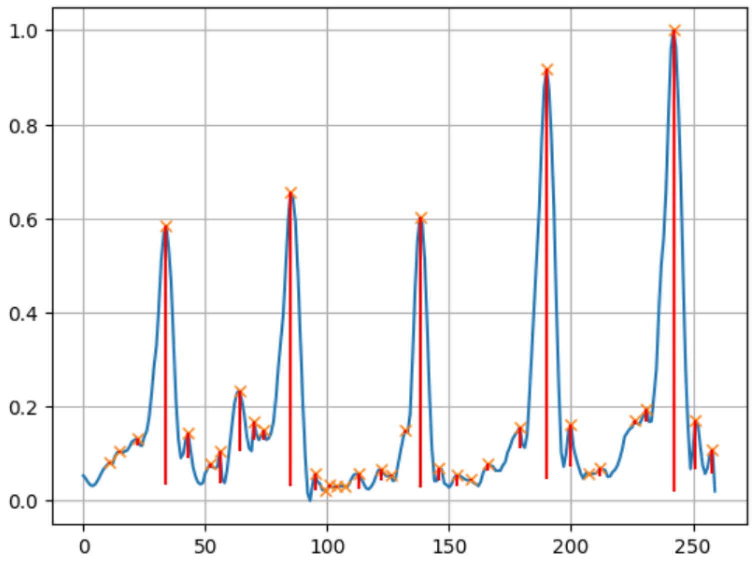


Figure 3.14: “Semana Blanca” prominences

which allows us to see if that segment resembles the infected curve or not. It is possible that there are some relative minimums to both left and right of the peak that do not really mark the beginning and end of that segment; i.e. they are not the “base peaks”. This is because even though the filter has removed most of the impurities from the curve, there are still some minuscule ones. Therefore, when calculating these “base peaks”, these impurities must be taken into account.

Fig. 3.15 illustrates the final curve selected, once the ”base peaks” have been calculated.

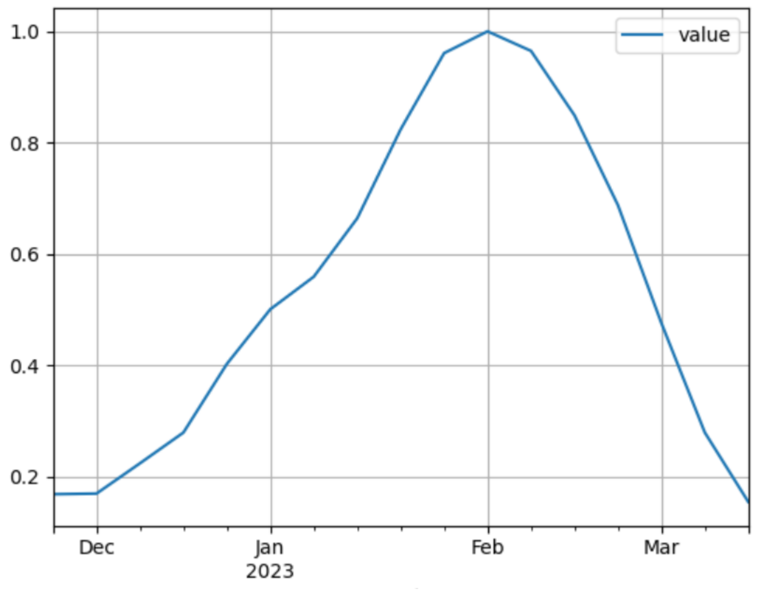


Figure 3.15: “Semana Blanca” selected curved

3.5 Fit to SIR Model

This last section will consist of fitting our neologism, which has been processed, applied a filter, and selected a curve segment from it, with the infected curve of the SIR model, and verifying if it has a similar behavior.

Before starting, it is important to remember the SIR model equations that have been defined above. These equations are as follows:

$$\begin{cases} \frac{dS(t)}{t} = -\beta * I(t) * S(t) \\ \frac{dI(t)}{t} = \beta * I(t) * S(t) - \gamma * I(t) \\ \frac{dR(t)}{t} = \gamma * I(t) \end{cases} \tag{3.1}$$

where β corresponds to the infection ratio, γ to the recovering ratio, and finally I(t), S(t), and R(t) to the infected, susceptible and recovered people at instant t.

For our case, we only want the results of the infected group, since, as we have commented above, the curve generated from these results has a behavior similar to that of the selected neologisms.

To realize this process with Python, we first have to use the method *ode_int* from the subpackage *integration* of the library *Scipy* for integrating a system of ordinary differential equations, as we have in Eq. 4.1. To do this, we have to define the differential equations in Python and set the initial conditions for these differential equations (CI). The predefined CI for all the selected neologisms will be as follows:

- β and γ parameters will have both a value of 0.
- There will not be any recovered person at the beginning, so $R = 0$
- The people susceptible to infection (S group) will be the maximum value of the curve segment selected in 3.4, i.e. the maximum value of popularity for that segment.
- The infected value (I group) will be the first element of the curve segment, i.e. the left "peak base" calculated in the last section.

Once we have defined all initial conditions for our differential equations, we are ready to get the results of the variation of the infected people ($\frac{dI(t)}{t}$). Fig. 3.16 illustrates this result for "Semana Blanca", which is composed of the curve segment of the neologism "Semana Blanca" and its first fit with the infected curve using the predefined initial conditions.

As we can see, due to $\beta = 0$, i.e. there is no infection ratio, the result of the adjustment is a straight line with a value equal to that established in the initial conditions for the infected group.

The next step will consist of carrying out a process that will vary the predefined initial conditions, so that the adjustment achieved is sufficiently similar to the neologism data. Of the five parameters established as initial conditions, only the value of infected people will remain the same as at the beginning, since it is the real value at the beginning of the "infection", i.e. the value at which the neologism begins to increase its popularity. However, the other four parameters are susceptible to change since they are values whose real value is unknown from the beginning.

To achieve this variation, the Python library Lmfit is used, which as mentioned in Sec. 2.3.3 is used to perform curve fitting in Python using the optimization method scipy.optimize.leastsq(). To do this, a model must be created with the difference equations in which the parameters to be varied to perform the fit are set. Once created, it is indicated to which curve it should fit, in our case to the chosen curve segment, and finally the fit is obtained.

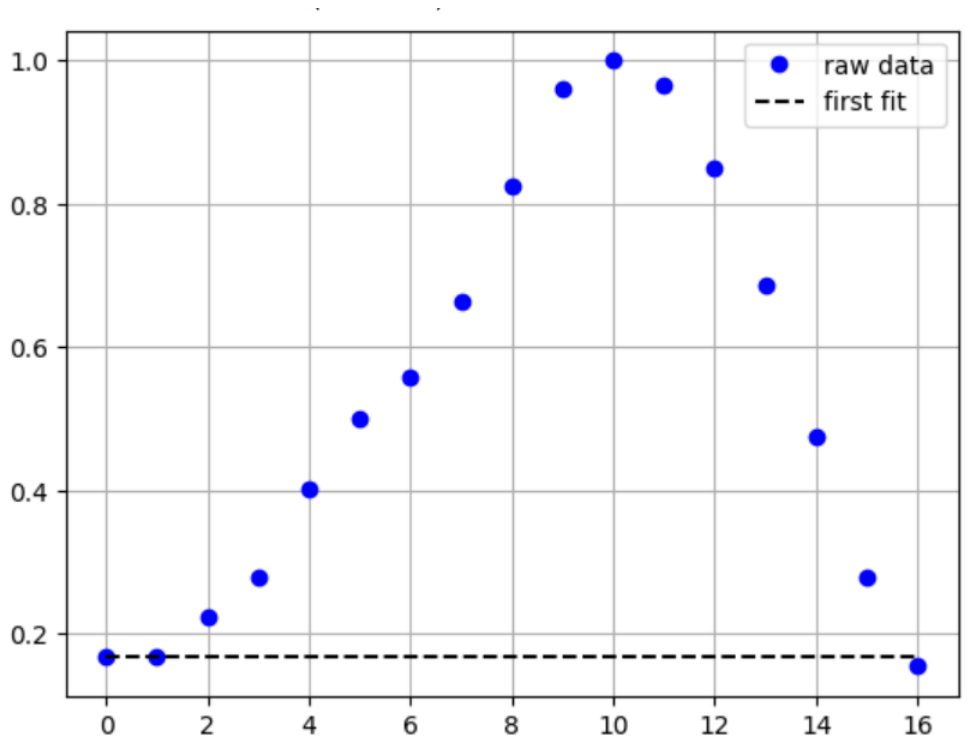
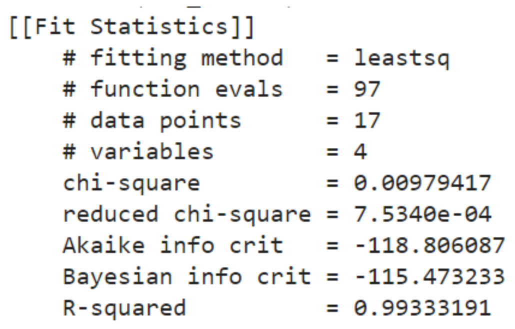


Figure 3.16: “Semana Blanca” first fit

The value that tells us whether we have made a good fit is the coefficient of determination or R^2. R^2 [1] is a measure that provides information on the goodness of fit of a model. In the context of regression, it is a statistical measure of how well the regression line approximates the actual data.

Fig. 3.17 shows different statistics results for “Semana Blanca” once we have done the fitting, in which there is a value of 0.99333191 for R^2 (99.33%). Furthermore, Fig. 3.18 illustrates this best fit with a graph.



```
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 97
    # data points      = 17
    # variables        = 4
    chi-square         = 0.00979417
    reduced chi-square = 7.5340e-04
    Akaike info crit   = -118.806087
    Bayesian info crit = -115.473233
    R-squared          = 0.99333191
```

Figure 3.17: “Semana Blanca” statistics result

As we can see, this neologism has a very similar behavior with the infected curve of

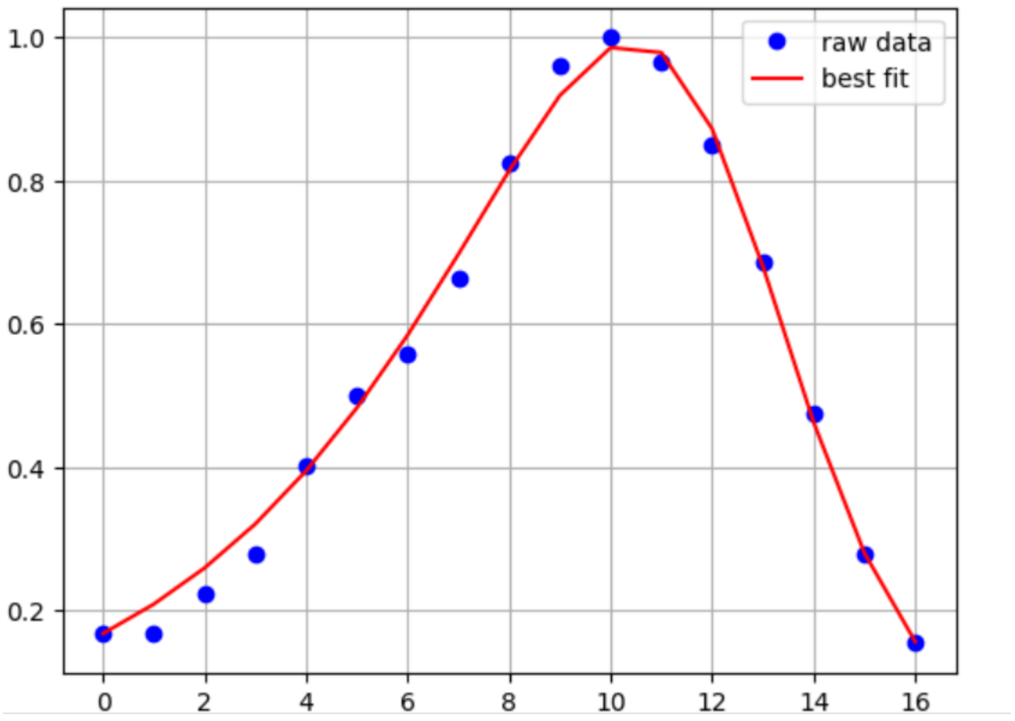


Figure 3.18: “Semana Blanca” best fit

the SIR model, so we can assert that a priori this process of detecting neologism can be replicated for all neologisms that have a trend curve similar to “Semana Blanca”. In fact, if we choose another curve segment of this neologism, we will realize that we will get a similar result.

Despite the good results that we have presented above, it could happen that due to bad filtering the result obtained is a totally different one. If we continue with the example of “Semana Blanca”, and decide to use other parameters for filtering (window_length and polyorder), then proceed to choose a segment of the curve following the algorithm explained in Sec. 3.4 and finally, if we fit the curve using the Lmfit library, we will notice that the result obtained is much worse than the previous one. Fig. 3.19 illustrates the best fit for this neologism having changed the filter parameters. In this case R^2 is only 73.98%, unlike the 99.33% we got above.

If this happens, we will have to go back to Sec. 3.3 and apply the filter again, to get another result, which can fit better with the curve segment selection algorithm.

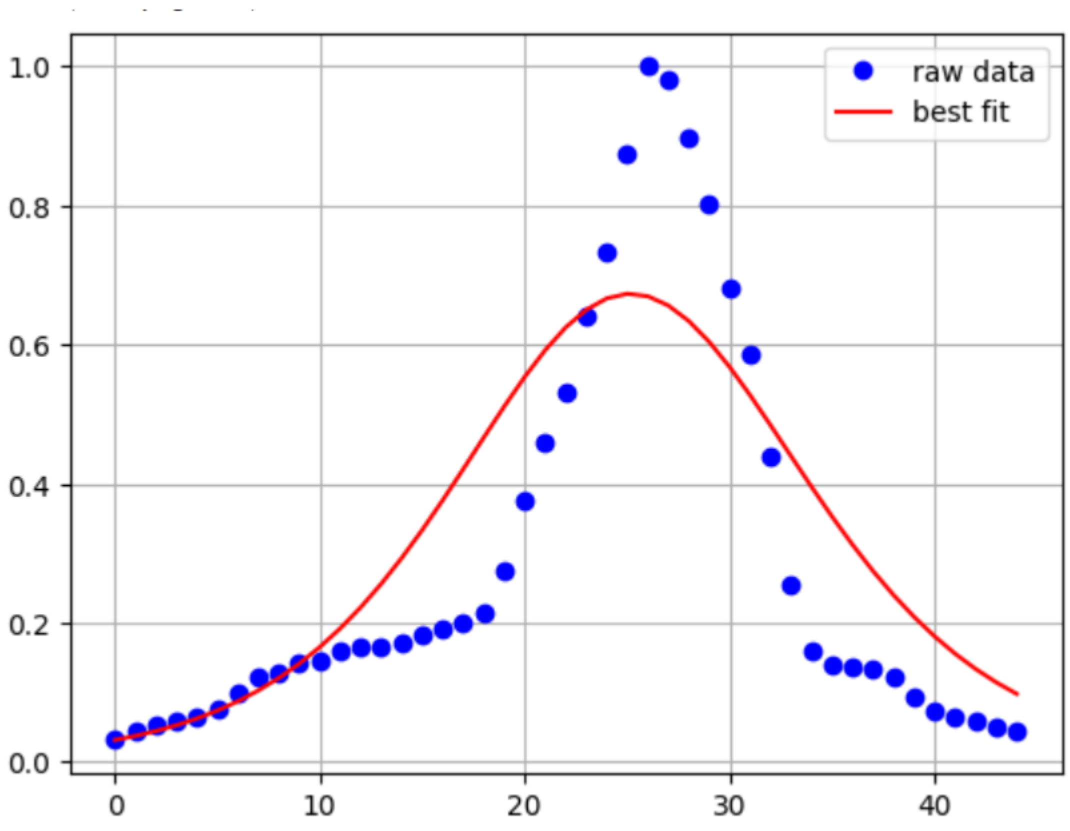


Figure 3.19: “Semana Blanca” bad fit

CHAPTER 4

Case study

In this chapter we will describe selected use cases.

In this chapter, we will illustrate different results of fitting to the SIR model different neologisms using the methodology explained above. We will comment step by step again on all the stages we have done to get the final results, but this time, we will not enter in details about what each stage consists of. We will just show the results we have obtained for each phase, what kind of problems we have and how we solve it, and we will describe some interesting code, in order to see some algorithms we have implemented as explained in Chap. 3. As a reminder, the methodology described previously consists of *selecting from "Martes Neológico" a neologism*, *checking if it meets the predefined statements*, *analyzing and preprocessing the neologism data*, *reducing the noise from the trend curve*, *selecting a curve segment* and *fitting it to the SIR model.*

4.1 Neologisms selection

Before starting, we have to explain the different neologisms we are going to use in the following sections and what its trend curve looks like. These neologisms are; "*Animalista*", "*Poliamor*", "*Estafa Piramidal*", "*Tripanofobia*" and "*Mansplaining*".

4.1.1 "Animalista"

This neologism [14], which is translated as animalist and refers to a "person who advocates for animal rights", was first documented in 1985 in the Oxford English Dictionary (OED). In the Observatori de Neologia (OBNEO) database, numerous occurrences have been recorded. We can highlight, for example, an interview published on April 20, 1989 by La Vanguardia, to Manuel Cases Puig, president of the first Spanish NGO dedicated to animal welfare, the Asociación to the Defense of Animal Rights (ADDA in Spanish). In it, reference is made to "a very intense animal movement".

The meaning of the adjective "Animalista" in the RAE is the following: "that says of art or its manifestations: which has as its main motif the representation of animals". As we can see, its meaning focuses on the field of sculpture and painting, and it is also indicated that it applies to the person "who cultivates animalistic art", which means that it also functions as a noun. In short, the appearance of the words "animalist" and "animalism" only reflects greater consideration and defense of animals.

Fig. 4.1 illustrates the trend curve extracted from Google Trends for this neologism.

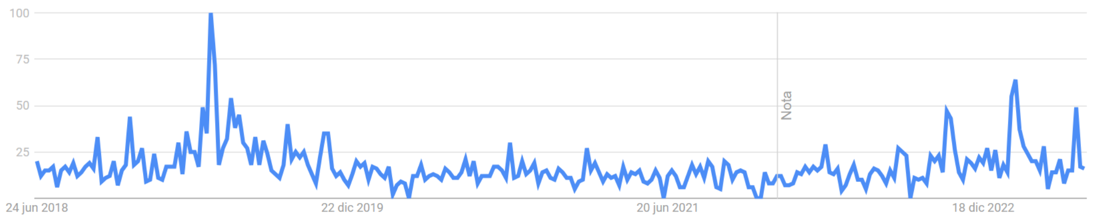


Figure 4.1: "Animalista" trend curve

4.1.2 "Poliamor"

"Poliamor" [22] consists of a non-monogamous social and sentimental system in which sexual or romantic relationships are established with more than one person in a consensual manner (under an ethical conception) between all involved people. In the case of formalizing the links, what are called constellations or pollicles (polyamorous relationships) begin to be established. Although the origin of this neologism is uncertain, one of the first times the use of "Poliamor" was documented was through the adjective polyamorous in 1990 in the article "A bouquet of lovers", by Morning Glory Zell-Ravenhart, published in the American magazine Green Egg.

"Poliamor" is a neologism with the Greek prefix poly-, which designates plurality or abundance, and the noun "amor". Its use is increasingly widespread, and it is not surprising that we can find the word "Poliamor" in corpora, with presence in Mexico, Chile, Argentina, Spain, Costa Rica (CORPES XXI). In the case of English, we find the word polyamory and polyamorous as the 'practice (or the person who practices) of having romantic (and usually sexual) relationships with the consent of the people involved.' In the OBNEO database, very recent examples are also found from different Spanish-speaking countries that indicate that it is a current neologism.

Fig. 4.2 shows the trend curve for this neologism.

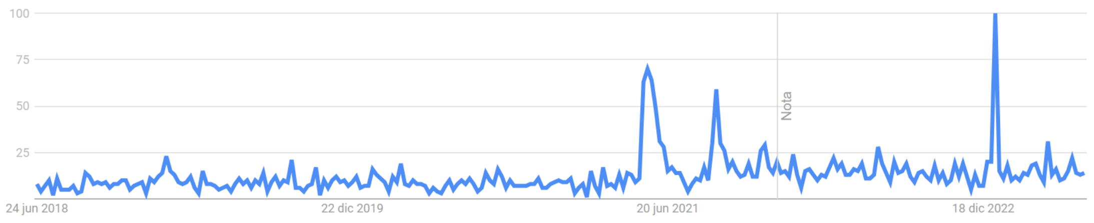


Figure 4.2: "Poliamor" trend curve

4.1.3 "Estafa Piramidal" [8]

Its alleged creator, Carlo Ponzi, an Italian immigrant who settled in the United States at the beginning of the twentieth century, devised a complex network based on the purchase of coupons in his country of origin, at that time with a devalued currency, to exchange them in the United States for postage stamps whose value in dollars was much higher.

Despite the fact that this method of cheating has been around for more than a century, the term "Estafa Piramidal" seems to be brand new. It is not registered in the Dictionary of the Spanish Language of the Royal Academy, and yet it is frequently reported in the Spanish and Latin American press.

The "Estafa Piramidal" is born with a promise of great benefits that come from a non-existent business, and are nourished by the constant incorporation of investors who believe the deception. The participants never get the profits they expected, because they always end up at the top of the pyramid, i.e., in the pockets of the creators of the fraud.

Fig. 4.3 illustrates the trend curve for "Estafa Piramidal".

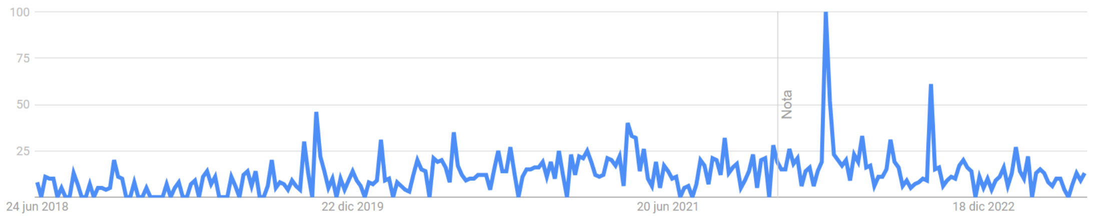


Figure 4.3: "Estafa Piramidal" trend curve

4.1.4 "Tripanofobia"

"Tripanofobia" [5] is related to fear of injections. Although this condition or phobia is not new, the truth is that "Tripanofobia" is a neologic word that has increased its use since the advent of the pandemic. This word has an interesting etymology, formed by two Greco-Latin elements. The first element, trepanum, comes from the Latin trepanum, which in English means 'drill'. The second element, phobia, comes from the Latin phobia, which is defined by the dictionary of the language as 'aversion' or 'rejection.' Textually, this neologism means 'aversion to the drill' and can be paraphrased as 'fear of being drilled', an action that effectively happens when we are given an injection.

This word does not appear in the RAE dictionary nor in any other dictionary of the Spanish language. It does not appear in the Oxford English Dictionary for English, nor in LeRobert for French. From a lexicographic point of view, it could be considered a neologism; however, we must consider that it is actually a word that has slipped from a specialized register into the general language.

Fig. 4.4 illustrates its trend curve extracted from Google Trends.

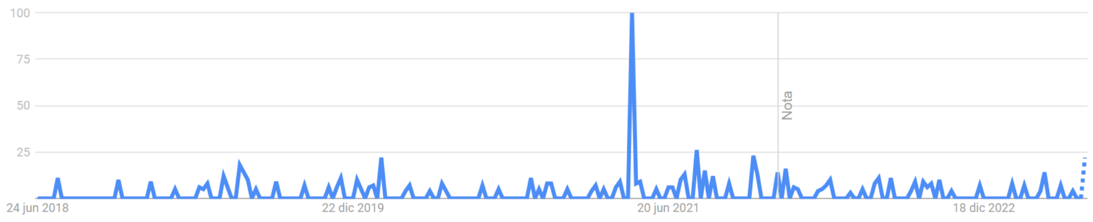


Figure 4.4: "Tripanofobia" trend curve

4.1.5 "Mansplaining"

The term mansplaining [23] designates the male practice of explaining something to a woman in a condescending manner, assuming that he knows much more about the subject (or that she knows nothing) and can correct her. It is a noun that comes from the gerund form of the English verb mansplain. It appears in reference works such as the Oxford English Dictionary, the Cambridge Dictionary, or the Merriam-Webster Dictionary and is formed by composing the words 'man' and 'explain'. Mansplaining is one of the many terms that have emerged to denounce the unfair realities that women face on a daily basis.

Finally, Fig. 4.5 shows the trend curve for this neologism.

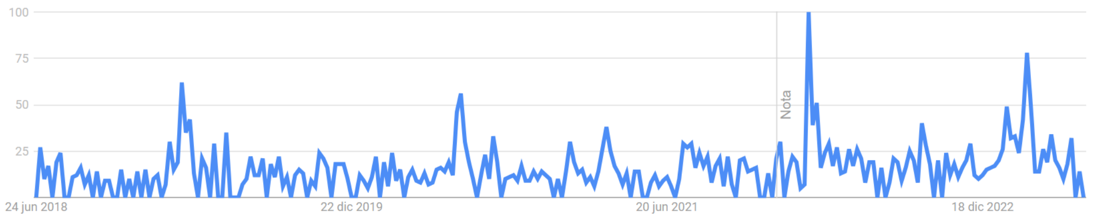
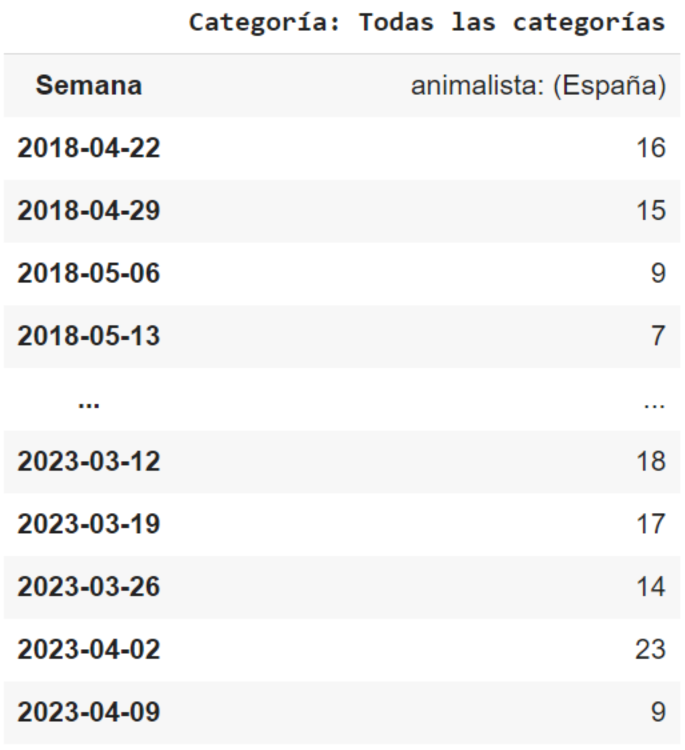


Figure 4.5: "Mansplaining" trend curve

4.2 Analysis and processing

Once the different neologisms have been discussed and the trend graphs for each one have been shown, we proceed to extract the data from these curves in order to process them and adapt them to a format with which we can work in the following stages. It is important to mention that unlike the previous section, where there was an explanatory subsection for each neologism, in this case the processing of the neologisms will be identical for each one of them, and therefore, we will comment on the solutions that have been given in the form of code to the problems that have been appearing.

Once we have got the neologisms data, we have to transform it to adequate results for the following stages. As a reminder, the kind of data extracted from Google Trends is illustrated in Fig. 4.6 for the neologism “Animalista”. The type of structure that makes up the data is called DataFrame and is created from the Python library Pandas.



Categoría: Todas las categorías

Semana	animalista: (España)
2018-04-22	16
2018-04-29	15
2018-05-06	9
2018-05-13	7
...	...
2023-03-12	18
2023-03-19	17
2023-03-26	14
2023-04-02	23
2023-04-09	9

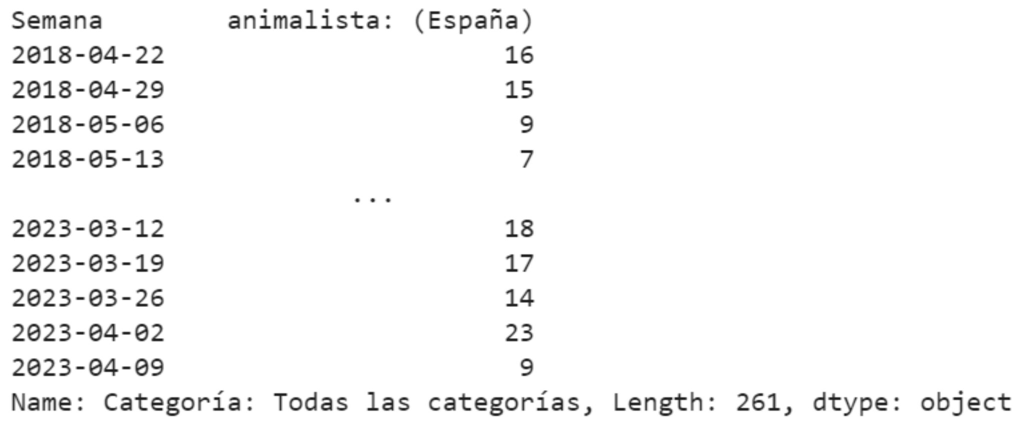
Figure 4.6: “Animalista” data extracted from Google Trends

As was commented on Sec. 3.2, the first step is to keep only the different dates of the column “Semana” and the popularity value for that date. At the beginning of the project, this was a problem because these two columns were inside the generic column “Categoría: Todas las categorías”. The solution to this first problem was to convert this *DataFrame* into a *Series* (another structure type of the Pandas library), and then convert it again to a *Dataframe*, but this time only with the required columns. The following code shows how this process has been done, in which we take the name of the file where the neologism data is stored. Then we open it in ‘csv’ format, and transform it into a *Pandas Series*. Finally, we transform it again to a DataFrame composed of two columns, ‘week’ for dates and ‘value’ for the popularity, and then remove the first row because it does not have relevant data. The code 4.1 is as follows:

Listing 4.1: Transform raw data to a better DataFrame

```
name = 'animalista'
csv_file = name + '.csv'
raw_data = pd.read_csv(csv_file)
sf = pd.Series(raw_data["Categoria: Todas las categorias"])
data = pd.DataFrame({'week':sf.index, 'value':sf.values})
df = data.drop([0])
```

Fig. 4.7 illustrates the 'Pandas Series' for the neologism "Animalista", in order to see that it is really easy to transform this structure, which is composed of the two inner columns into a new DataFrame, where we just need to remove the first value of each column, correspond to 'Semana' in the case of 'week' column and 'animalista: (España)' in the case of 'value' column.



```
Semana        animalista: (España)
2018-04-22                      16
2018-04-29                      15
2018-05-06                       9
2018-05-13                       7
                       ...
2023-03-12                      18
2023-03-19                      17
2023-03-26                      14
2023-04-02                      23
2023-04-09                       9
Name: Categoría: Todas las categorías, Length: 261, dtype: object
```

Figure 4.7: "Animalista" Pandas Series

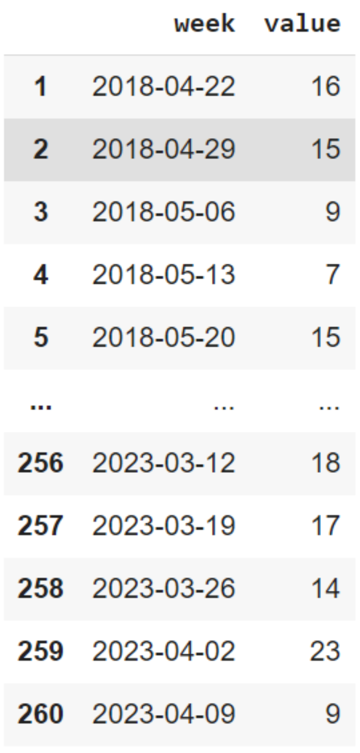
Once we have solved this problem, another one arises. We have to check what is the data type of each column, and if they do not have the correct one, we have to transform into it. In case of 'week' column, we just use the method *to_datetime* from Pandas library to transform *object* into *date*. However, in the case of the 'value' column, although most of the values in this column are numbers, but with the 'object' data type, which means that the transformation to 'integer' format is trivial, on some dates the value '< 1' appears to indicate that the popularity of the neologism on those dates is very low. The solution to this problem will be to transform these values to zero but in the 'object' format, and then transform the whole column to the 'integer' format. The code 4.2 that performs this process is the following:

Listing 4.2: Transform data types

```
df.loc[df.value == "<1", 'value'] = '0'
df.week = pd.to_datetime(df.week)
df['value'] = df['value'].astype('int')
```

In this case, we use the method 'loc' to find the columns that verify that the column 'value' has as value '< 1', and once found, replace them with the value "0". Finally, we can easily convert both columns into the required data types.

As a result, Fig. 4.8 shows the final DataFrame for the neologism "Animalista".



	week	value
1	2018-04-22	16
2	2018-04-29	15
3	2018-05-06	9
4	2018-05-13	7
5	2018-05-20	15
...	...	...
256	2023-03-12	18
257	2023-03-19	17
258	2023-03-26	14
259	2023-04-02	23
260	2023-04-09	9

Figure 4.8: "Animalista" final DataFrame

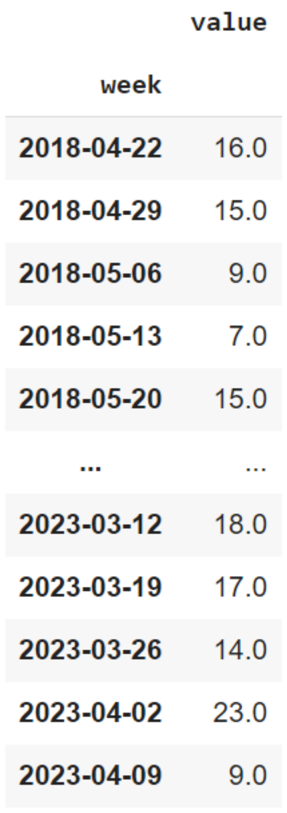
4.3 Noise reduction

This section shows the different results obtained for each of the chosen neologisms, after the application of the filtering process, together with a description of the code that made it possible.

The first step in this stage is to transform the final DataFrame obtained in the last section into a new DataFrame with one column corresponding to the popularity values and whose index is made up of the different dates of those values. The following code 4.3 shows how this transformation is done (by grouping in weeks the different dates and performing the average operation for each of the groups) and Fig. 4.9 the result of it for the neologism “Animalista”.

Listing 4.3: Group data by weeks

```
model = df.groupby(pd.Grouper(key='week', freq='W')).mean()
```



	value
week	
2018-04-22	16.0
2018-04-29	15.0
2018-05-06	9.0
2018-05-13	7.0
2018-05-20	15.0
...	...
2023-03-12	18.0
2023-03-19	17.0
2023-03-26	14.0
2023-04-02	23.0
2023-04-09	9.0

Figure 4.9: “Animalista” new DataFrame for filtering process

As we commented in Sec. 3.3, the digital filter that we use in this stage is the Savitzky-Golay filter. However, before using this method, two different ones were tested, in order to see if they get better results and to understand how the filtering process works for these kinds of curves, but finally they were discarded. These two methods, which are explained below and why they were discarded are called *Convolution* and *Rolling*.

4.3.1 Convolution

As was explained in Sec. 2.3.1, this method performs the convolution operation between an operating window and the data of the different neologisms. The following code 4.4 illustrates how this operation is performed.

Listing 4.4: Convolution operation

```
kernel_size = 10
kernel = np.ones(kernel_size) / kernel_size
convolve_model = np.convolve(model.values.flatten(), kernel, mode='same')
```

The operation consists of establishing a window size (kernel_size variable) and dividing each element of an array of the window size formed by all ones by the window size itself, to obtain the final window to which the convolution will be applied. Finally, using the method 'convolve' from the library 'Numpy', the convolution operation is done among the neologism data (the new DataFrame created above) and the final window. Fig. 4.10 shows the result of this convolution operation in the neologism "Animalista".

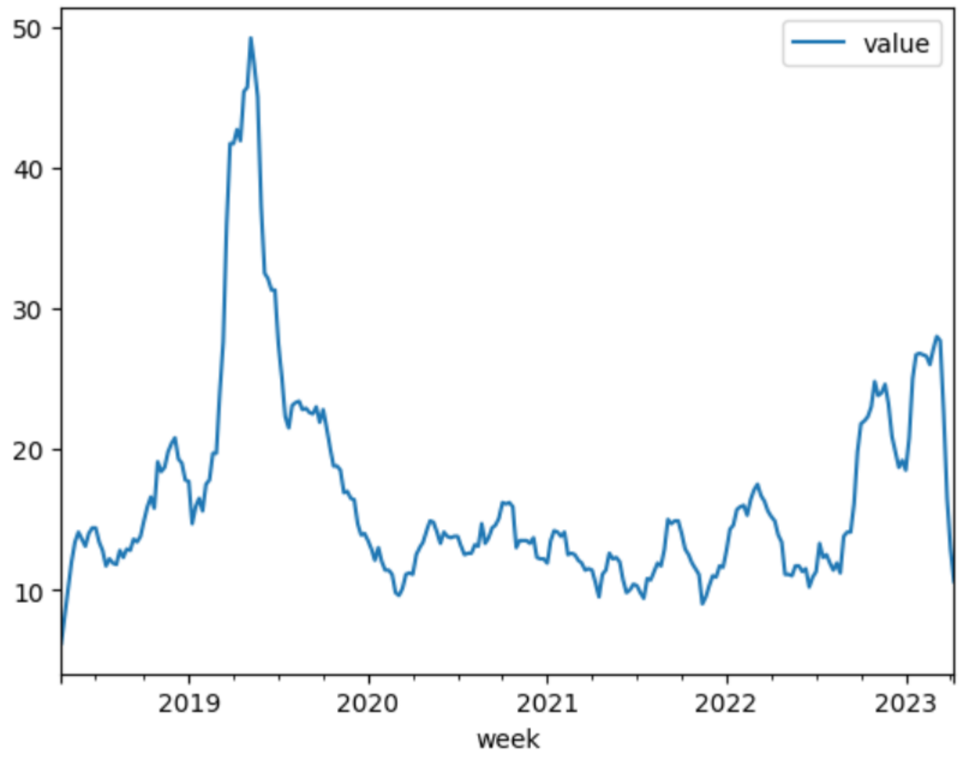


Figure 4.10: "Animalista" convolution operation result

Although this operation maintains the trend of the curve, which is the rule followed to select which type of filter to use, we decided to discard it, because the Savitzky-Golay filter is a more sophisticated tool or method that can be manipulated more easily than this one.

4.3.2 Rolling

This process consists of selecting a window range in which an operation will be performed as in the convolution process. This time, the window is formed by a set of weeks in a row, and the internal operation will be the average of popularity values grouped for that set of weeks. The code 4.5 is as follows:

Listing 4.5: Rolling operation

```
rolling_model = model.rolling('35D', center=True, closed='both').mean()
```

In this case, the set of weeks will be 5, or, in other words, 35 days.

Fig. 4.11 shows the result of this method for the neologism “Animalista”.

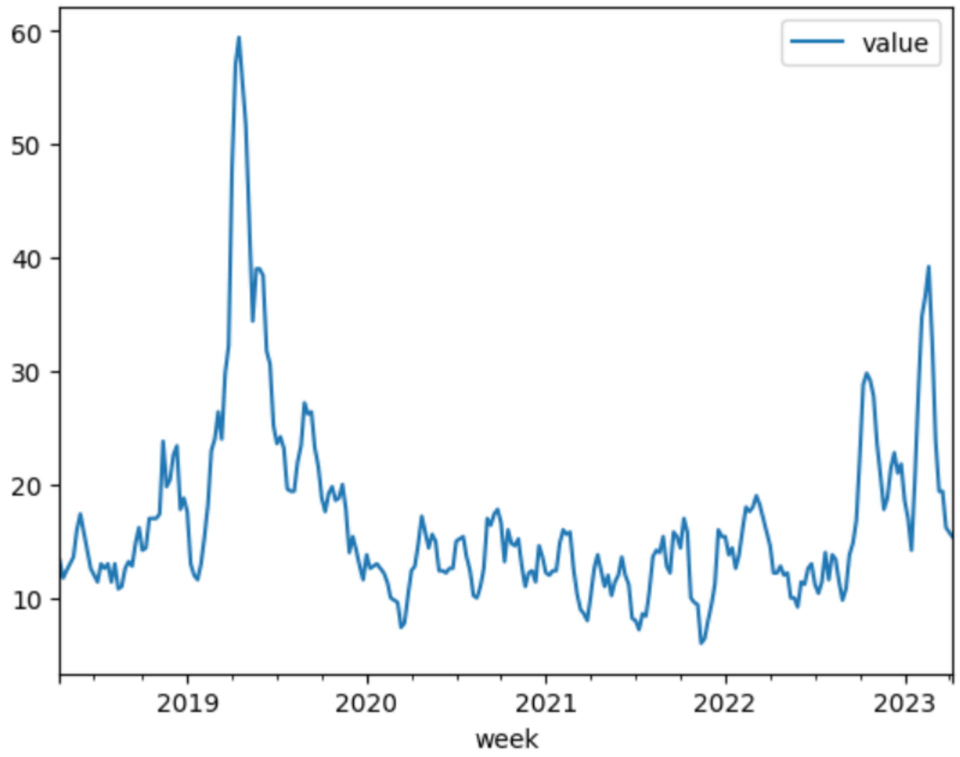
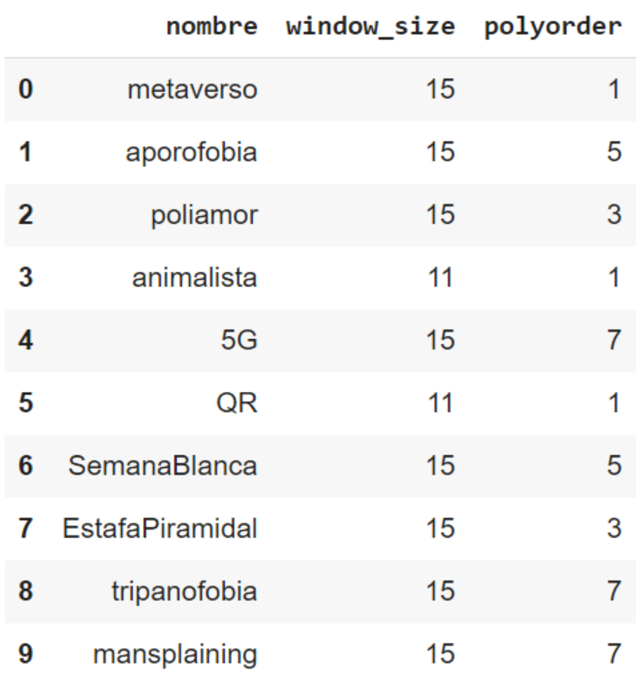


Figure 4.11: “Animalista” rolling operation result

Finally, this method is discarded because it is an operation that does not remove all the noise and also because it is a very simple operation that does not provide a great deal of flexibility when working.

4.3.3 Savitzky-Golay

In order to apply this filter, as is commented in Sec. 3.3, we have to take note of two parameters: *window_length* and *polyorder*. Because we intend to implement an automated process to filter neologisms, a new DataFrame has been created called *filter variables* with the values of these two parameters for each neologism. In this way, when filtering, the parameters will be chosen according to the neologism being treated. Fig. 4.12 illustrates this DataFrame with the neologisms used and its filter parameters.



	nombre	window_size	polyorder
0	metaverso	15	1
1	aporofobia	15	5
2	poliamor	15	3
3	animalista	11	1
4	5G	15	7
5	QR	11	1
6	SemanaBlanca	15	5
7	EstafaPiramidal	15	3
8	tripanofobia	15	7
9	mansplaining	15	7

Figure 4.12: Filter variables DataFrame

Once we have this DataFrame, we use its variables to apply the filter. The code 4.6 is as follows:

Listing 4.6: Savitzky-Golay operation

```
model_filtered = signal.savgol_filter(model.values.flatten(), window_length
    =window_size, polyorder=polyorder)
svg_model = pd.DataFrame(model_filtered, index=model.index, columns=model.
    columns)
```

In order to have the same values in the following stages, we scale all the resulting curves in a range between zero and one using the method *MinMaxScaler* from the library *Sklearn*, as shown in the code 4.7 below:

Listing 4.7: Scale results

```
scaler = preprocessing.MinMaxScaler()
d = scaler.fit_transform(svg_model)
scaled_model = pd.DataFrame(d, columns=model.columns, index=model.index)
```

Finally, the following figures illustrate the filtering results for each neologism compared with the data extracted from Google Trends.

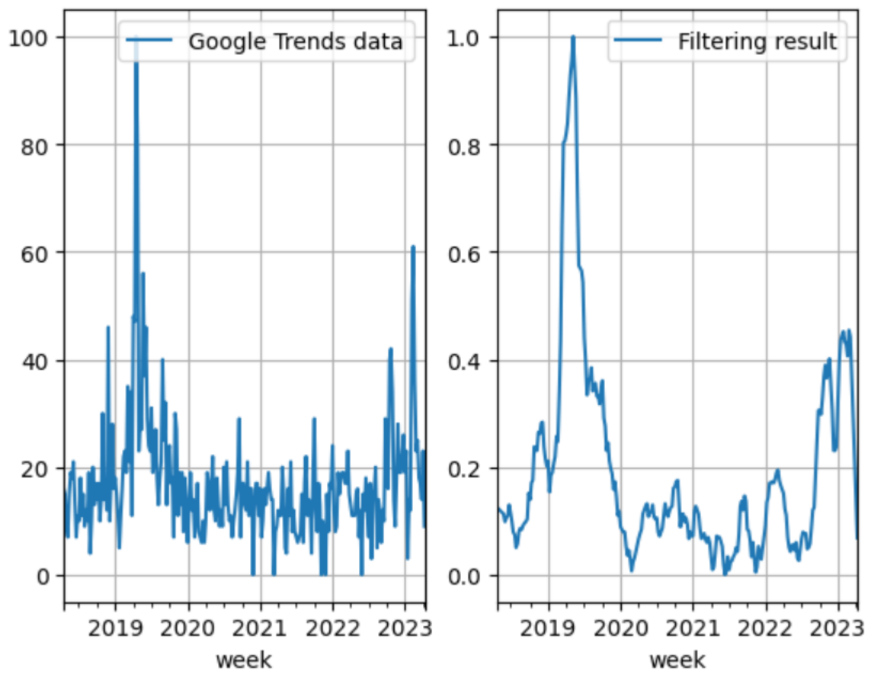


Figure 4.13: Filtering results compared with Google Trends curve for neologism “Animalista”

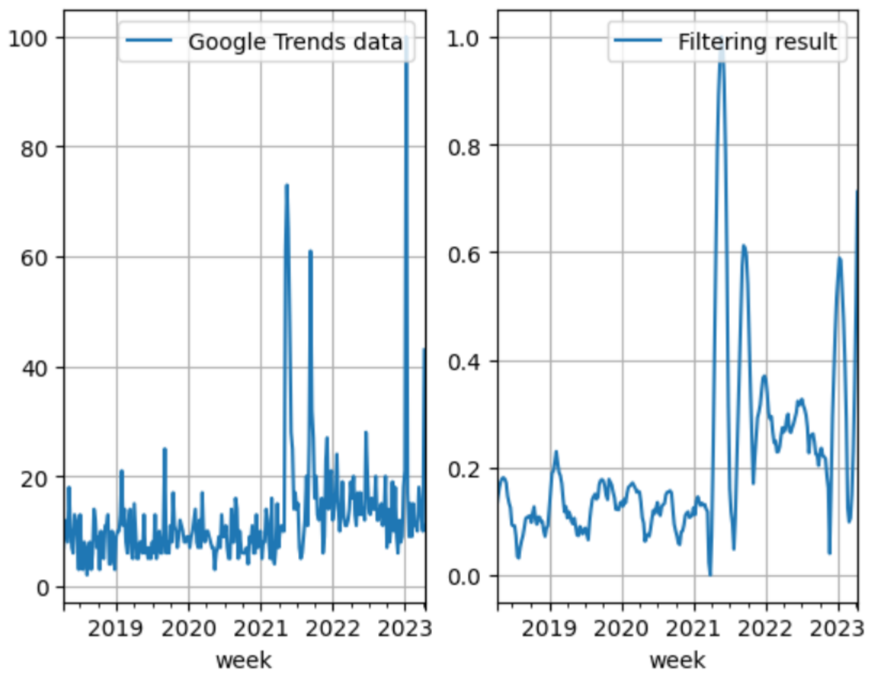


Figure 4.14: Filtering results compared with Google Trends curve for neologism “Poliamor”

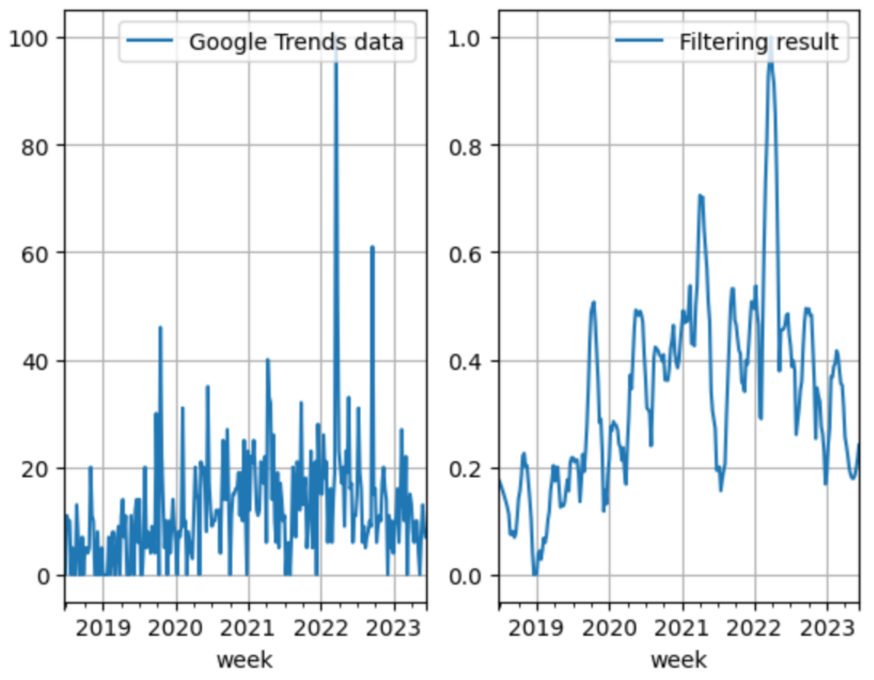


Figure 4.15: Filtering results compared with Google Trends curve for neologism “Estafa Piramidal”

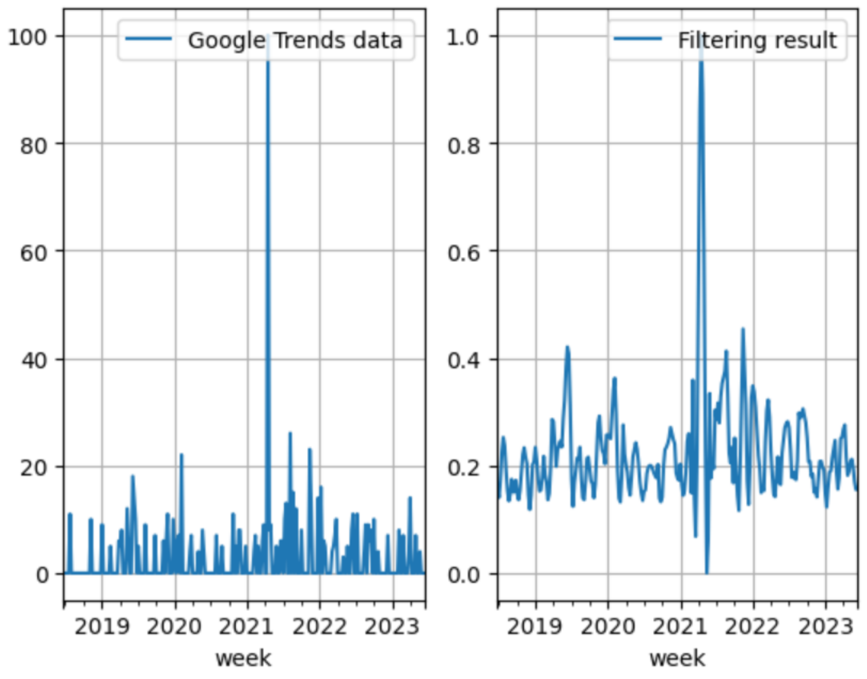


Figure 4.16: Filtering results compared with Google Trends curve for neologism “Tripanofobia”

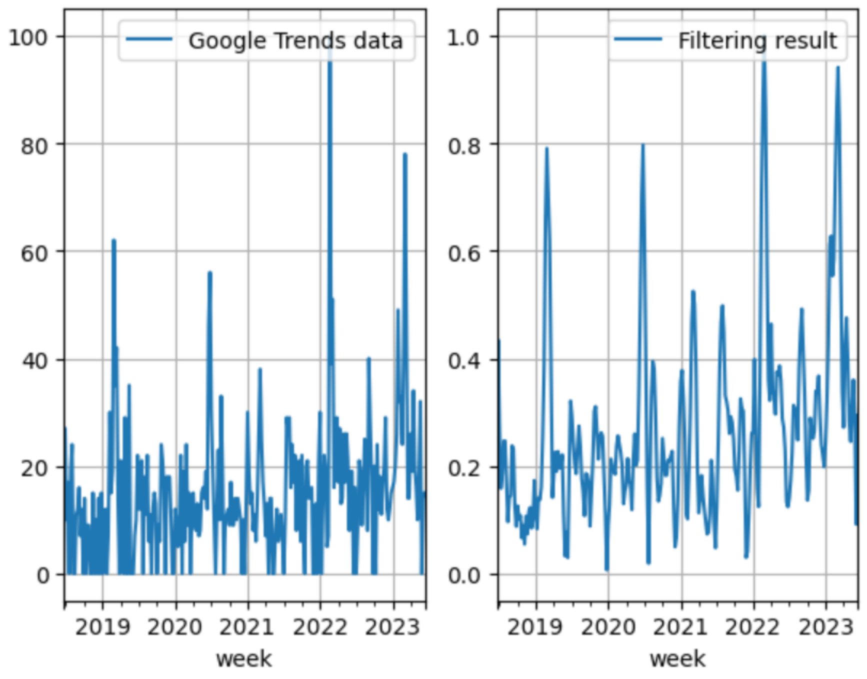


Figure 4.17: Filtering results compared with Google Trends curve for neologism “Mansplaining”

4.4 Curve Segment Selection

At this moment and as we commented in Sec. 3.4, we need to select a curve segment, which corresponds to a rise in terms of popularity, as in an epidemic increases the number of infected people. We have already commented on the algorithm for doing this process, so in this section we will explain the code that made it possible and the results of the process for each neologism we have selected in this chapter.

The first step consists of getting all the prominences from all the peaks of the curve in order to select the highest one. The code 4.8 is as follows:

Listing 4.8: Get the prominneces of the curve

```
peaks, _ = signal.find_peaks(x)
prominences = signal.peak_prominences(x, peaks)[0]
```

We can see that first we use the method *find_peaks* for finding all the peaks of the curve and then get the prominences for those peaks using the method *peak_prominences*. As a reminder, Fig. 4.18 illustrates an example of these prominences for the neologism “Animalista”.

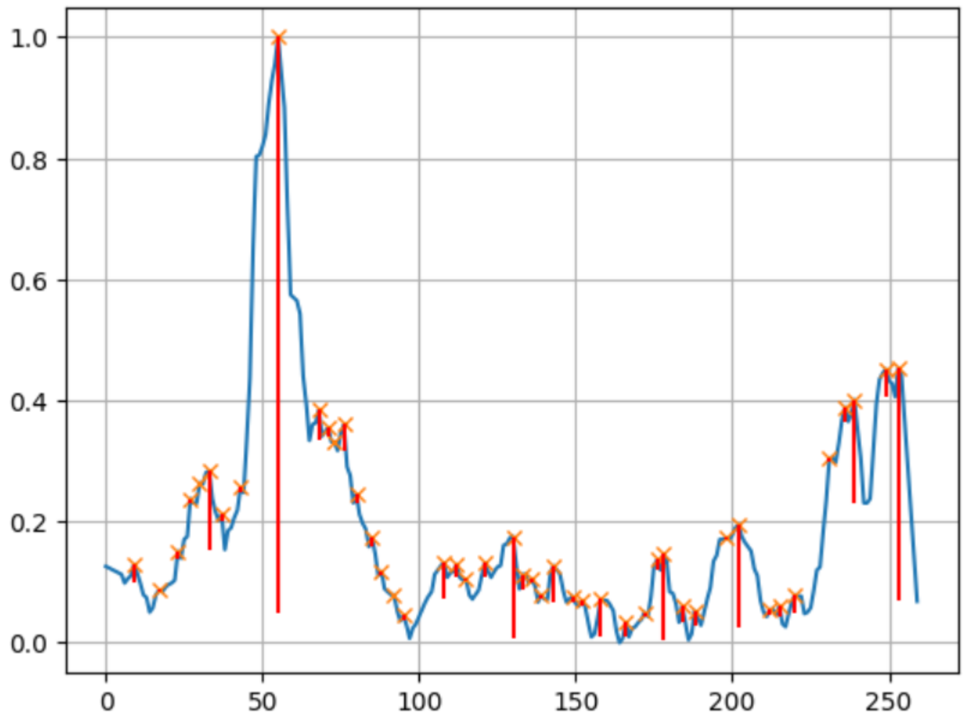


Figure 4.18: “Animalista” prominences

The second step is to obtain the highest prominence and its peak from all the ones we have. The code 4.9 illustrates this process:

Listing 4.9: Obtain highest prominence

```
out = np.array((peaks, prominences)).T
best_prominence = [0, 0]
for peak, prominence in out:
  if prominence > best_prominence[1]:
    best_prominence[0] = int(peak)
    best_prominence[1] = prominence

best_prominence
```

In this case, we create an array of tuples, each tuple composed of a prominence and its peak. Then, we create a predefined array with two elements, both zero, called *best_prominence*. Finally, we run through a for loop in order to find the highest peak, and when we find it, we store it in the array, as same as its peak.

The final step consists of getting what we called in Sec. 3.4 *base peaks*. As a reminder, these *base peaks* are the relative minimums at the right and left of the peak. These points are useful because with them we are able to know when the curve started to grow and when it stopped to descend. The code 4.10 is as follows:

Listing 4.10: Peak_bases

```
def left_base(peak):
  first_value = scaled_model.iloc[peak].value
  second_value = scaled_model.iloc[peak - 1].value
  if (((second_value > first_value) and (abs(second_value - first_value) >=
      0.01))) or (second_value == 0) :
    return peak
  else:
    return left_base(peak - 1)

def right_base(peak):
  first_value = scaled_model.iloc[peak].value
  second_value = scaled_model.iloc[peak + 1].value
  if (((second_value > first_value) and (abs(second_value - first_value) >=
      0.01))) :
    return peak
  else:
    return right_base(peak + 1)
```

Both functions (left_base and right_base) are based on recursion. They start from the peak (first_value) and from it, they take a point to the left or right of it (second_value), depending on the function. Once we have both values, we check a condition. If the condition is true, we have found our minimum and stop; but if it does not, we call again the function, but this time our reference value (the first_value) will be the second_value we have just taken.

The condition we might verify is different for both functions. In case of left_base function, we find our left relative minimum if the second_value is is at least ten percent higher than the first one or it is zero. The percentage is because as we commented in Sec. 3.4 even though the filter has removed most of the impurities from the curve, there are still some minuscule ones that are considered as noise and we do not have to take care of them. Therefore, we select this percentage to skip these impurities.

On the other hand, the condition for right_base function is exactly the same, except that we do not care if the second_value is zero or not.

Once we have explained the code of this process, we present the results of it for each neologism in the following figures.

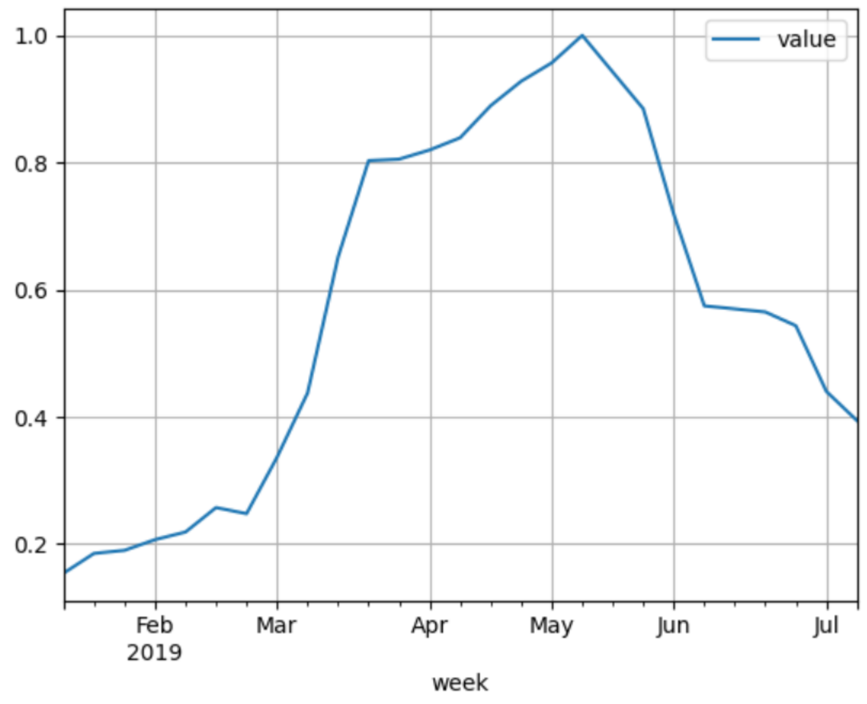


Figure 4.19: "Animalista" curve segment

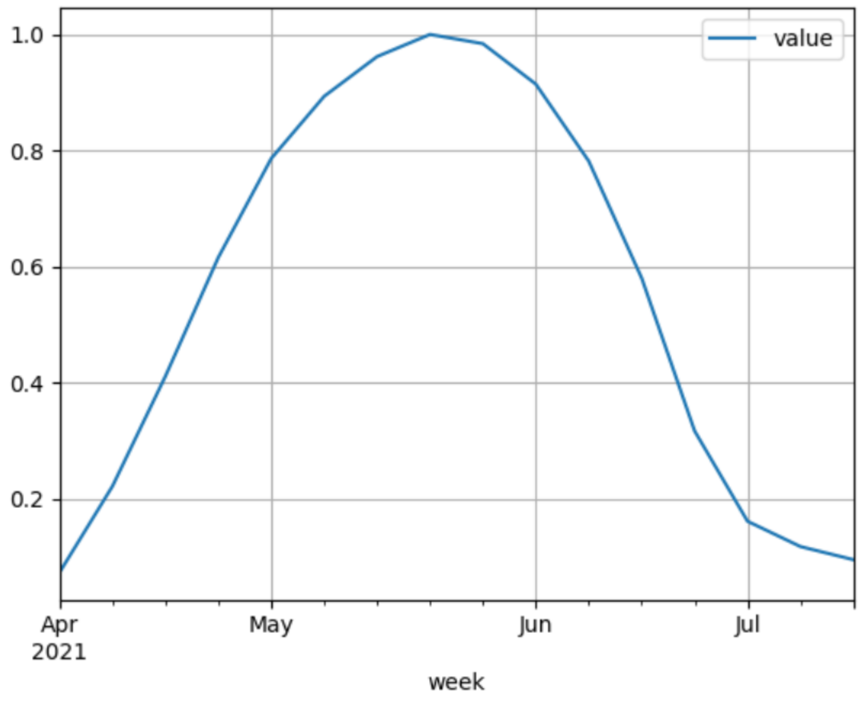


Figure 4.20: “Poliamor” curve segment

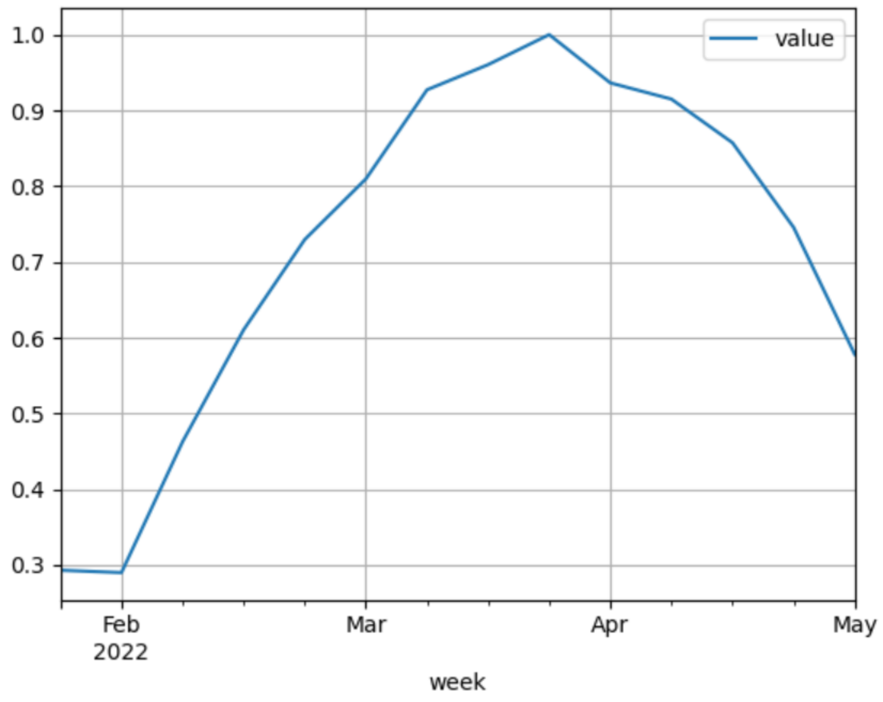


Figure 4.21: “Estafa Piramidal” curve segment

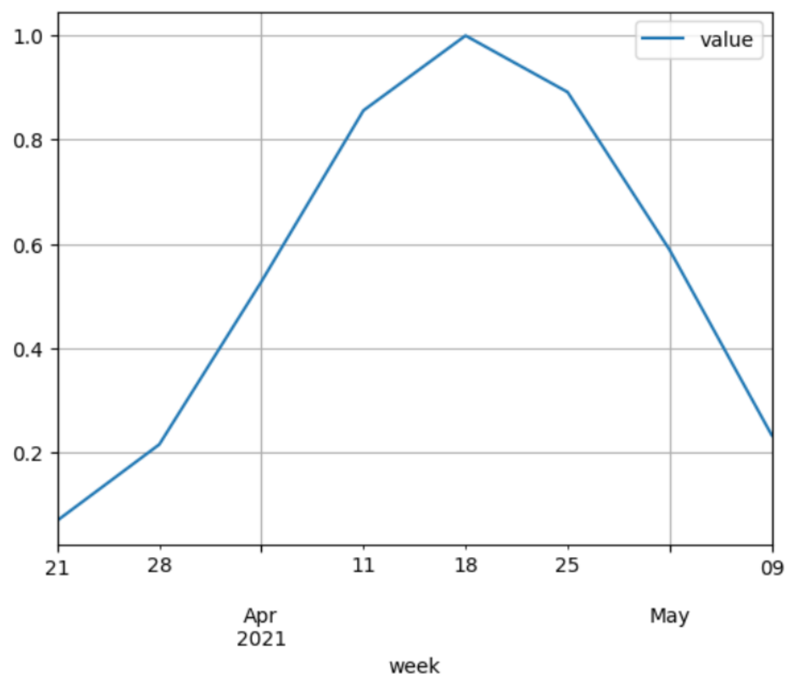


Figure 4.22: "Tripanofobia" curve segment

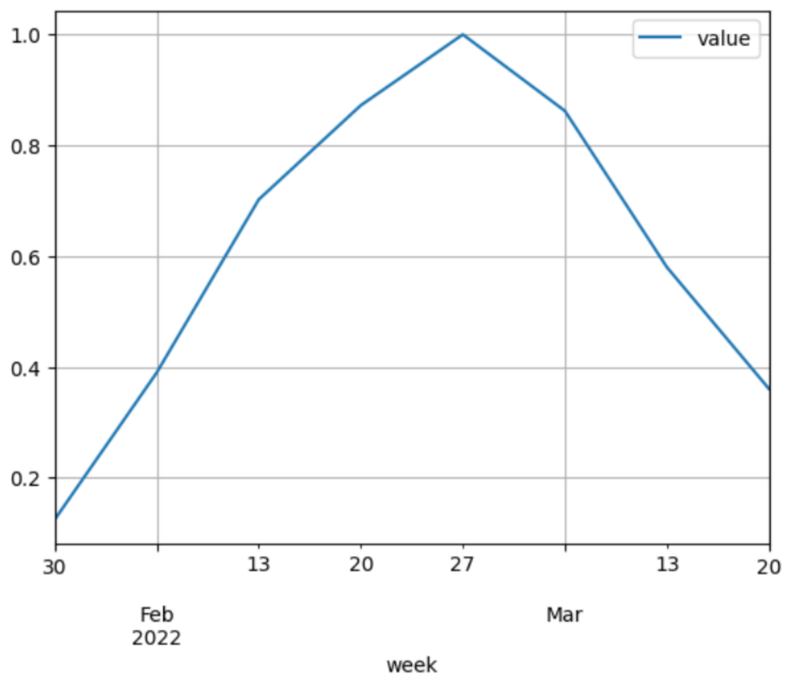


Figure 4.23: "Mansplaining" curve segment

4.5 Fit to SIR Model

In this last section, we will comment on the code that allows us to fit the curve segments selected above with the infected curve of the SIR model. In addition, we will show the results of these fittings, along with the *R-squared* parameter for each neologism.

The first step will be to create a Python function that is consistent with the three differential equations of the SIR model. It is worth mentioning once again these equations:

$$\begin{cases} \frac{dS(t)}{t} = -\beta * I(t) * S(t) \\ \frac{dI(t)}{t} = \beta * I(t) * S(t) - \gamma * I(t) \\ \frac{dR(t)}{t} = \gamma * I(t) \end{cases} \tag{4.1}$$

The function created for these equations is called *deriv* and the Python code 4.11 is as follows:

Listing 4.11: Differential equations code

```
def deriv(y, time, beta, gamma):
    S, I, R = y
    dSdt = -beta * S * I
    dIdt = beta * S * I - gamma * I
    dRdt = gamma * I
    return dSdt, dIdt, dRdt
```

Once we have defined this function, the next step will be to use the *odeint* method from the sub-package *integrate*, in order to get a result for the *deriv* function for the three equations, assigning the initial conditions. The function that describes this proccess is called *ode_Model* and its code 4.12 is the following:

Listing 4.12: ode_Model function

```
def ode_Model(test_model, beta, gamma, S0, R0):
  I0 = test_model.iloc[0]
  y0 = S0, I0, R0 #CI
  t = np.linspace(0, len(test_model), len(test_model))
  ret = integrate.odeint(deriv, y0, t, args=(beta, gamma))
  S, I, R = ret.T

  return S, I, R
```

As we can see, the initial condition for the infected curve is static and equal to the first value of the curve segment. However, because we do not know the initial values for β, γ, S0 and R0, we have to assign them a predefined value (as we saw in Sec. 3.5) and pass them as parameters of the function.

Finally, we create another function that we introduce in a model created by the library *Lmfit*, to modify the different initial conditions in order to obtain the best fit between the data of the different neologisms and the previously described infected curve.

This function is called *fit_odeint*, in which we use the *ode_Model* function, in order to get a result for each differential equation for some initial conditions and return just the result of the infected for each instant of time. The code 4.13 is as follows:

Listing 4.13: fit_odeint function

```
def fit_odeint(time, beta, gamma, S0, R0):
  ret = ode_Model(test_model, beta, gamma, S0, R0)
  I = ret[1]
  time = time.astype(int)
  return I[time]
```

Once we have defined this function, we create a model of the *Lmfit* library, pass it the predefined initial conditions, and use the *fit* method to fit the result of *fit_odeint* to the data of each neologism. The code 4.14 is described below:

Listing 4.14: Create Lmfit model and fit to neologism data

```
gmod = Model(fit_odeint)
params = gmod.make_params(beta=0, gamma=0, S0=test_model.max().values[0],
    R0=0)
result = gmod.fit(test_data, params, time=time)
```

To finish this chapter, the following figures illustrate the best fit of each neologism, along with the *R-squared* parameter for each figure.

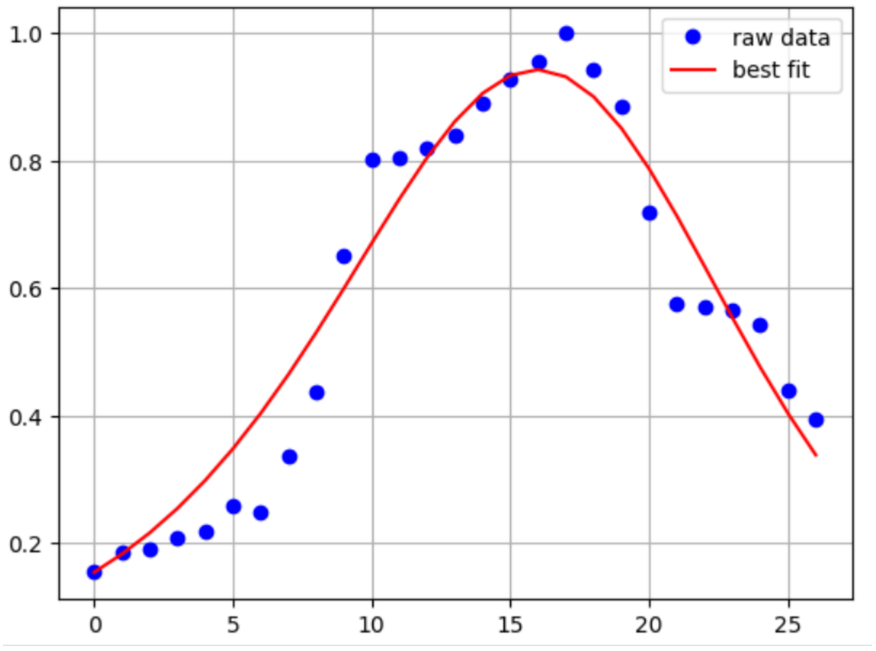


Figure 4.24: “Animalista” best fit with an *R-squared* of 93.53%

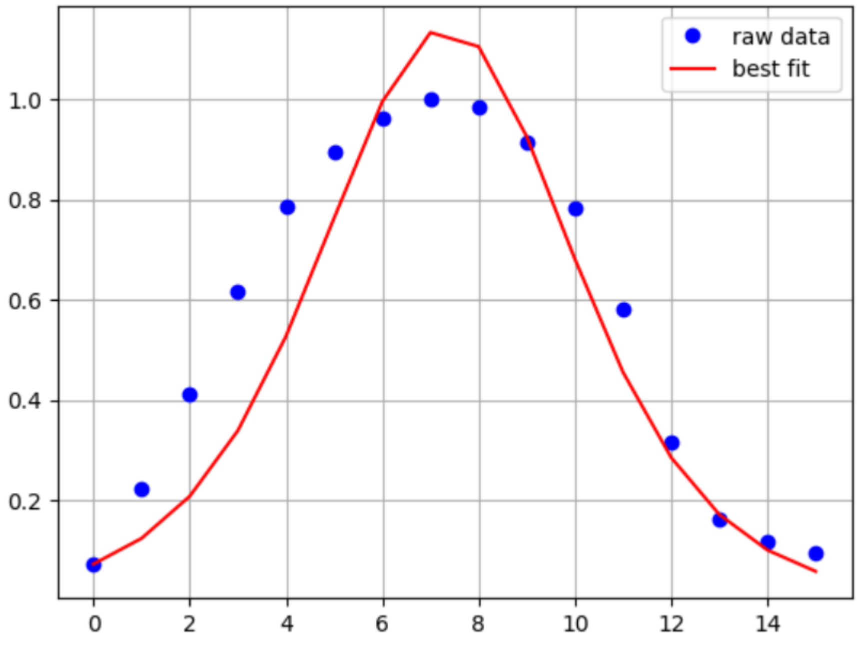


Figure 4.25: “Poliamor” best fit with an *R-squared* of 85.47%

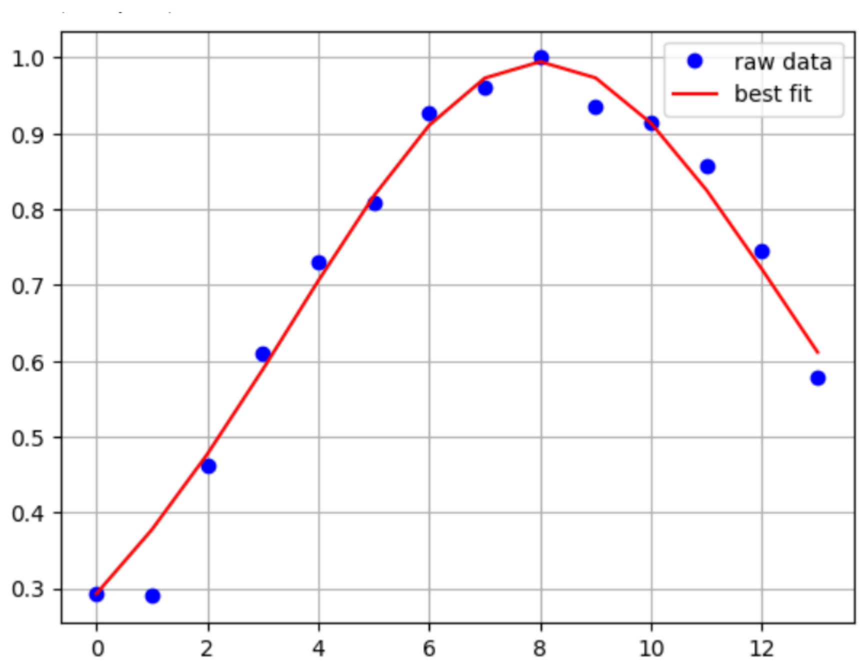


Figure 4.26: “Estafa Piramidal” best fit with an *R-squared* of 98.2%

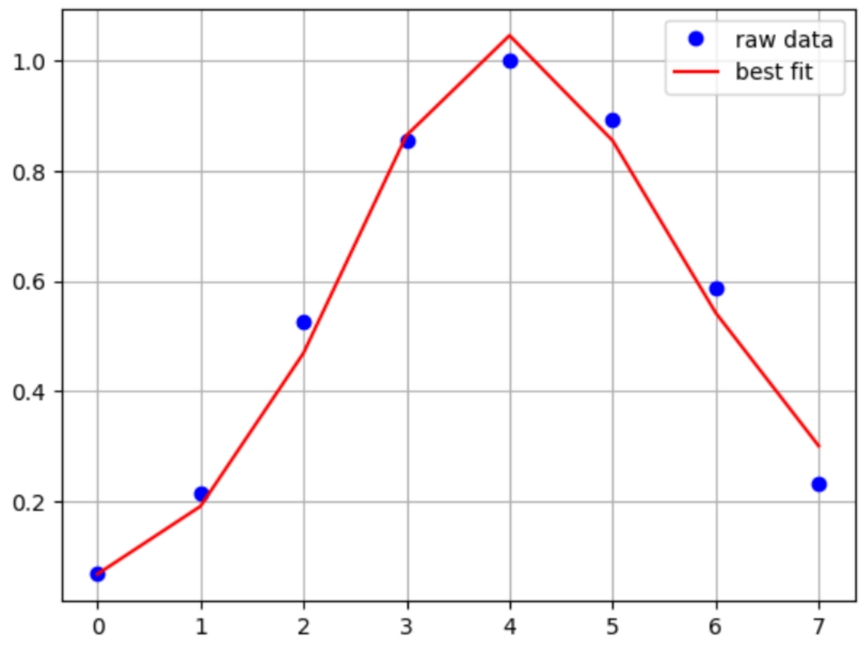


Figure 4.27: “Tripanofobia” best fit with an *R-squared* of 98.35%

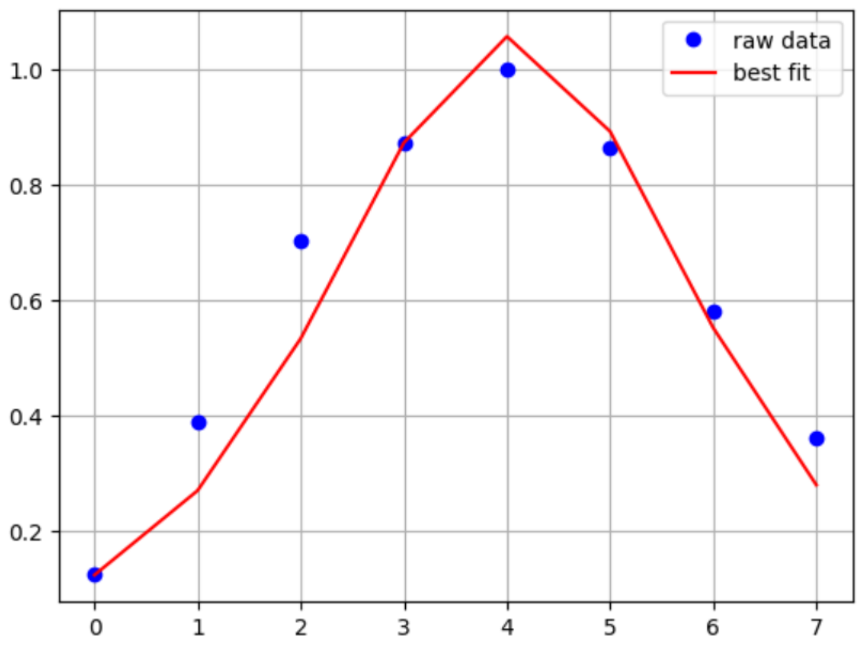


Figure 4.28: “Mansplaining” best fit with an *R-squared* of 91.58%

CHAPTER 5

Conclusions

This chapter will describe the goals achieved by the master thesis following some of the key points developed in the project.

5.1 Achieved Goals

As mentioned in Sec. 1.2, the main objective of this project is to fit as best as possible the popularity data of a neologism following an epidemiological model called SIR. As this objective has been fulfilled in the stages described above, it allows us to state that any type of word, whether existing words with a new meaning or totally new words, as long as at some point in its lifetime there has been a significant increase in its popularity, can be modeled according to the differential equations of the SIR model, and therefore, a priori it is able to be detected as a neologism.

In addition to this objective, others that complemented the main objective were described, and it is therefore important to mention whether or not they have been met, as follows:

- Correct guidelines were established for the selection of a neologism, since it is possible to achieve the main objective with any neologism that adjusts to the criteria.
- It has been possible to detail which curve segment should be selected for this study and why.
- An algorithm has been implemented that is subdivided into several phases, such as noise reduction or curve segment selection, to achieve a correct fit to the SIR model.
- Finally, although the final process is not completely automatic, since it depends on an individual observation of the different results obtained in each of the different stages mentioned to confirm that the task has been accomplished, it can be stated that each separate stage is fully automatic, and even some contiguous phases are automatic with respect to each other.

As commented in Chap. 3 and shown in Fig. 3.1, we use the elastic stack, to illustrate all the results we got from the neologism detection process. However, this last step is done during the production step and is not developed in this project, but it will be the first step to take to improve the quality of the project and the final product.

5.2 Conclusion

As conclusions of this project and related to the main objective of it, it is worth mentioning that this kind of neologism detection is totally different from what is normally done, such as natural language processing techniques. However, most of these neologisms, as we have commented, in terms of use or popularity generally have the same growth shape, and therefore we are able to perform this kind of process to detect them. In addition, this method is also suitable for all words that at a certain moment can arise because many people start using them, and, therefore, its popularity rises. If that happens, we can apply this process to verify that those words are behaving as an infection, in which many people use them.

Beyond the ability to fit neologisms to an infection curve, this project will allow us to make predictions about what behavior will have our words in the future, so we can get ahead of what is happening now and benefit ourselves.

Finally, using ElasticSearch and Kibana, we will be able to index our neologisms data to this stack and create a much more automatized process, in which introducing a neologism, Elastic will prompt the best solution for that neologism and its prediction for the future.

APPENDIX A

Project impact

This appendix reflects, quantitatively or qualitatively, the possible social, economic, and environmental impact jointly with ethical implications.

A.1 Social impact

This section describes the main social impacts that our work might have. As has been mentioned many other times in this project, knowing how people interact is fundamental to understanding them. In this way, we are able to get closer to them and therefore, to influence them.

Thanks to the use of neologisms and this type of work, we are able to know which are the most frequent words in use today, and thus be able to carry out different activities that are closer to the population, such as advertising acts, political speeches, etc.

A.2 Economic impact

In this section, we summarize the main economic impacts of our project. As in the case of social impact, being able to convey in the right words to the population what you are trying to communicate enables great economic growth. As an example, we can illustrate an advertising campaign in which using the right words makes it possible to get closer to customers, and thus obtain greater benefits.

A.3 Environmental impact

This section briefly describes the main environmental impacts of our project.

In this sense, the biggest environmental problem is the energy used to carry out the whole process.

Currently, there are no large energy expenditures, as the execution process takes only seconds. However, in case of implementing some future line such as the prediction of the trend of a neologism, or the use of Elastic, it could lead to a rise of energy when carrying out this process, which implies to be more harmful to the planet.

A.4 Ethical implications

Finally, this section presents the main ethical considerations.

Just as this work presents great opportunities to learn firsthand how people communicate and what kinds of expressions they use, it also presents a problem that needs to be discussed.

Having these kinds of tools makes it possible to gain greater power of conviction, since you are able to interact as someone today does, and therefore have greater influence over them.

Finally, you can also comment on the manipulation that you can have over another individual, and get them to do something they don't really want to do.

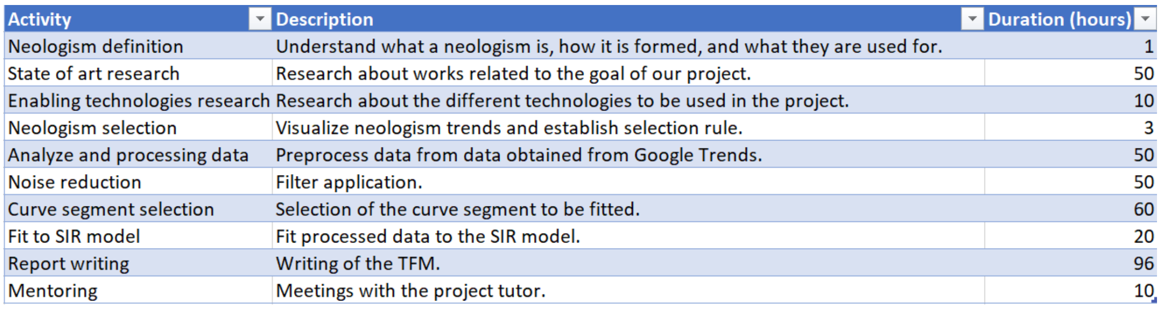
APPENDIX B

Project budget

This appendix details an adequate budget to carry out the project. The project structure is described along with the activities undertaken to complete it and costs are evaluated, including material and human resources, as well as taxes.

B.1 Project structure

In order of achieving the proposed goals, the project has been divided into the activities shown in Fig. B.1. The total duration of the project is approximately 350 hours.



Activity	Description	Duration (hours)
Neologism definition	Understand what a neologism is, how it is formed, and what they are used for.	1
State of art research	Research about works related to the goal of our project.	50
Enabling technologies research	Research about the different technologies to be used in the project.	10
Neologism selection	Visualize neologism trends and establish selection rule.	3
Analyze and processing data	Preprocess data from data obtained from Google Trends.	50
Noise reduction	Filter application.	50
Curve segment selection	Selection of the curve segment to be fitted.	60
Fit to SIR model	Fit processed data to the SIR model.	20
Report writing	Writing of the TFM.	96
Mentoring	Meetings with the project tutor.	10

Figure B.1: Project Structure Table

B.2 Costs evaluation

This section summarizes the material resources needed to develop the project, divided into software and hardware resources, as well as the human resources and taxes associated with it.

B.2.1 Material resources

B.2.1.1 Software

All the project has been developed using open-source software available on the Internet for free. For this reason, there is no cost associated to software.

B.2.2 Hardware

To carry out this project, a computer has been used. Its specifications are as follows:

- Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 2496 Mhz
- 16 GB RAM
- Hard disk of 250 GB

The cost of a PC with these requirements is approximately **500€**.

B.2.3 Human resources

For the development of this project, it can be considered the work of one person during the entire project. Taking into account that the person working in the project is an engineering graduate, the estimated salary is 24,108€/year, which translated into the worked hours is approximately 837€.

B.2.4 Taxes

In case the final product is sold to an interested company, taxes related to a software engineering project must be taken into account. The fees paid by the company would be the corresponding VAT established in the local country.

B.3 Conclusion

The project had a duration of **350 hours** and the total cost ascends to **1337€**.

Bibliography

[1] Coefficient of determination, r-squared. https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/coefficient-of-determination-r-squared.html. Published in Numeracy, Maths and Statistics - Academic Skills Kit, Accesed: 16th/06/2023.

[2] Anni Burchfiel. What is nlp (natural language processing) tokenization? https://www.tokenex.com/blog/ab-what-is-nlp-natural-language-processing-tokenization/, MAY 2022. Published in Tokenex, Accessed: 8th/06/2023.

[3] Ignacio Corcuera-Platas. Development of a Deep Learning Based Sentiment Analysis and Evaluation Service. Master thesis, ETSI Telecomunicación, Madrid, January 2018.

[4] Jordi de la Vega Puig. código qr. https://blogscvc.cervantes.es/martes-neologico/codigo-qr/, October 2022. Published in Martes Neológico, Accesed: 14th/06/2023.

[5] Ramón F. Zacarías Ponce de León. Tripanofobia. https://blogscvc.cervantes.es/martes-neologico/tripanofobia/, November 2022. Published in Martes Neológico, Accesed: 19th/06/2023.

[6] Carlos Ruiz Fernández. Metaverso. https://blogscvc.cervantes.es/martes-neologico/metaverso/, March 2023. Published in Martes Neológico, Accesed: 12th/06/2023.

[7] Henri Froese. Infectious disease modelling: Understanding the models that are used to model coronavirus. https://towardsdatascience.com/infectious-disease-modelling-part-i-understanding-sir-28d60e29fdfc, April 2020. Published in Towards Data Science, Accessed: 14th/06/2023.

[8] Irene Roldán González. Poliamor. https://blogscvc.cervantes.es/martes-neologico/estafa-piramidal/, September 2022. Published in Martes Neológico, Accesed: 19th/06/2023.

[9] Atul Harsha. Understanding part-of-speech tagging in nlp: Techniques and applications. https://www.shiksha.com/online-courses/articles/pos-tagging-in-nlp/, December 2020. Published in shiksha online, Accessed: 8th/06/2023.

[10] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[11] Maarten Janssen. Neotrack: semi-automatic neologism detection. In *APL Conference*, 2005.

[12] Maarten Janssen. Neotag: a pos tagger for grammatical neologism detection. In *LREC*, pages 2118–2124. Citeseer, 2012.

[13] Menghan Jiang, Xiang Ying Shen, Kathleen Ahrens, and Chu-Ren Huang. Neologisms are epidemic: Modeling the life cycle of neologisms in china 2008-2016. *Plos one*, 16(2):e0245984, 2021.

[14] Elisabet Llopart-Saumell. Animalista. `https://blogscvc.cervantes.es/martes-neologico/animalista/`, March 2018. Published in Martes Neológico, Accesed: 19th/06/2023.

[15] Susanna Pardines López. chatbot. `https://blogscvc.cervantes.es/martes-neologico/chatbot/`, January 2023. Published in Martes Neológico, Accesed: 12th/06/2023.

[16] Matthew Newville, Till Stensitzki, Daniel B. Allen, and Antonino Ingargiola. LMFIT: Non-Linear Least-Square Minimization and Curve-Fitting for Python, September 2014.

[17] Jorge Diz Pico. tetris. `https://blogscvc.cervantes.es/martes-neologico/tetris/`, April 2018. Published in Martes Neológico, Accesed: 12th/06/2023.

[18] Juan Miguel Monterrubio Prieto. Semana blanca. `https://blogscvc.cervantes.es/martes-neologico/semana-blanca/`, June 2017. Published in Martes Neológico, Accesed: 12th/06/2023.

[19] Kurtis Pykes. Stemming and lemmatization in python. `https://www.datacamp.com/tutorial/stemming-lemmatization-python`, February 2023. Published in datacamp, Accessed: 8th/06/2023.

[20] Gloria Guerrero Ramos. aporofobia. `https://blogscvc.cervantes.es/martes-neologico/aporofobia/`, September 2016. Published in Martes Neológico, Accesed: 12th/06/2023.

[21] Helena Sofia Rodrigues. Application of sir epidemiological model: new trends. *arXiv preprint arXiv:1611.02565*, 2016.

[22] Pedro J. Bueno Ruiz. Poliamor. `https://blogscvc.cervantes.es/martes-neologico/poliamor/`, July 2021. Published in Martes Neológico, Accesed: 19th/06/2023.

[23] Irene Vílchez Sánchez. Mansplaining. `https://blogscvc.cervantes.es/martes-neologico/mansplaining/`, September 2022. Published in Martes Neológico, Accesed: 19th/06/2023.

[24] Elastic Development Team. ¿qué es elasticsearch? `https://www.elastic.co/es/what-is/elasticsearch`, June 2023. Published in Elastic.co, Accesed: 29th/06/2023.

[25] Elastic Development Team. ¿qué es kibana? `https://www.elastic.co/es/what-is/kibana`, June 2023. Published in Elastic.co, Accesed: 29th/06/2023.

[26] The Pandas Development Team. Pandas. `https://pypi.org/project/pandas/`, May 2023. Publised in pypi.org, Accessed: 8th/06/2023.

[27] Wikipedia contributors. Matplotlib — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Matplotlib&oldid=1158310790`, 2023. Accessed: 16th/06/2023.

[28] Wikipedia contributors. Natural language processing — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Natural_language_processing&oldid=1158537250`, 2023. Accessed: 8th/06/2023.

[29] Wikipedia contributors. Numerical integration — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Numerical_integration&oldid=1148174789`, 2023. Accessed: 16th/06/2023.

[30] Wikipedia contributors. Numpy — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=NumPy&oldid=1151671618`, 2023. Accessed: 16th/06/2023.

[31] Wikipedia contributors. Scikit-learn — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Scikit-learn&oldid=1151444875`, 2023. Accessed: 16th/06/2023.

[32] Wikipedia contributors. Scipy — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=SciPy&oldid=1154857736`, 2023. Accessed: 16th/06/2023.

[33] Nasser Zalmout, Kapil Thadani, and Aasish Pappu. Unsupervised neologism normalization using embedding space mapping. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 425–430, 2019.