Describing the Architecture of NegesAPI, an API for Spanish Negation Processing

José Valle-Aguilera, Salud María Jiménez-Zafra*, María Teresa Martín-Valdivia and L. Alfonso Ureña-López

Computer Science Department, SINAI, CEATIC, Universidad de Jaén, Campus Las Lagunillas, 23071, Jaén, Spain

Abstract

In this paper, we present NegesAPI, an API for negation detection in Spanish texts using Machine Learning techniques, as well as its Web Application. We describe its architecture, the development technologies used and the integrated negation detection system. It is a powerful tool that can assist in different Natural Language Processing tasks.

Keywords

Negation, negation processing, cue detection, scope identification, natural language processing

1. Introduction

Negation is a grammatical phenomenon present in all languages that can change the polarity of a sentence. It expresses the negative form of a sentence or statement [1]. In Spanish there exist different types of negation depending on the elements used to negate: syntactic negation (*no, nunca, nadie,...*), lexical negation (***im**pensable, **i**legible, **im**posible,...*) and morphological negation (*en la vida*). Also, negation can be found in multiple ways, including the use of simple markers (one negation cue: ***No** sé cocinar*), continuous markers (two or more consecutive negation cues: ***Casi no** llego a la presentación*) and discontinuous markers (two or more non-consecutively negation cues: ***No** tengo **nada** de dinero para la comida*).

Therefore, negation can be very challenging to detect and process. In fact, the computational treatment of negation has not been solved due to its complexity. In Natural Language Processing (NLP) negation is a phenomenon of special interest, since it affects the polarity of texts, such as opinions, where the opinion of a product can be drastically altered if negation is present. It also has an impact on information retrieval systems, where searching for *"Películas que no sean de fantasía"* (Non-fantasy films) is not the same as searching for *"Películas que sean de fantasía"*(Fantasy films). It is also of special relevance for entity recognition in biomedicine, where for example if a patient "*no tiene cáncer*" (does not have cancer) and the entity *"cáncer"* is recognised, but the presence of negation is not detected, it changes the meaning of the sentence and the diagnosis of the patient.

There are various approaches to handle negation in NLP, such as rule-based approaches [2, 3], and traditional machine learning [4, 5] and deep learning approaches [6, 7]. Rule-based approaches use a set of pre-defined rules to detect negation cues and modify the meaning of the sentence accordingly. Traditional machine learning and deep learning approaches, on the other hand, use annotated data to train models that can automatically identify negation cues and their scope in a sentence.

In this paper we describe the architecture of NegesAPI, an API to detect negation cues and scopes in Spanish texts, that uses the machine learning approach of Jiménez-Zafra et al. [5] and that is trained in the SFU Review SP-NEG corpus [8], which consists of 9,455 sentences from reviews on 8 domains (cars, hotels, washing machines, books, mobile phones, music, computers and movies). This API can analyze any text in Spanish and annotate its negation automatically, to assist in other NLP tasks.

The rest of the paper is organized as follows: In Section 2 we provide a description of the API and its architecture. In Section 3 we describe the components in the back-end. The front-end components are set out in Section 4. In Section 5 we present the negation system used by NegesAPI. Finally, Section 6 summarizes the conclusions and future work.

2. System Description

NegesAPI has two main modules, the API itself, and a Web Application, built on top of the API. Additionally, these two modules rely on different components, each one responsible of an specific task:

- Data model: NegesAPI uses a relational database

SEPLN-PD 2023: Annual Conference of the Spanish Association for Natural Language Processing 2023: Projects and System Demonstrations

*Corresponding author.

jvalle@ujaen.es (J. Valle-Aguilera); sjzafra@ujaen.es (S. M. Jiménez-Zafra); maite@ujaen.es (M. T. Martín-Valdivia); laurena@ujaen.es (L. A. Ureña-López)

0000-0003-3274-8825 (S. M. Jiménez-Zafra); 0000-0002-2874-0401 (M. T. Martín-Valdivia); 0000-0001-7540-4059 (L. A. Ureña-López)

system that stores the user list and information about them, in order to implement a login/register system. It should be noted that it does not store any information about the processed and analyzed texts. It is built in MariaDB as it is open source and it is the default database system in most Linux distributions.

- Views: The views contains all the information that is presented in the client side of the web application, as well as the design and functionalities. The information is written in HTML, the design is defined with CSS and minor functionalities executed in the client are implemented in JavaScript.
- Controller and routing: As routing for the web application we use Flask, a powerful Python framework with which we have created the different endpoints. Flask uses a routes file, where data validation is implemented and forms are created or retrieved. This routes file serves as controller, communicating with all other components whenever it is necessary. The API is also written in Flask, but it has been previously generated using Swagger Documentation.
- Negation system: The negation system is implemented in the API */analyze* route. It uses the Machine Learning approach of Jiménez-Zafra et al. [5], with an updated version of the corpus SFU Review SP-NEG [1].
- Request management: Flask already implements a WSGI server, the toolset of Werkzeug. Although it is more than enough for a development environment, it falls short in production, and it is necessary to apply several technologies and tools for a better user experience and system scalability. To achieve that, NegesAPI uses the Gunicorn WSGI HTTP server to replicate our Flask application and NGINX as reverse proxy and application gateway.

These components are classified into Back-end and Front-end, the former being in charge of logic, routing and security, and the latter in charge of presenting information, design and functionality on the client side.

3. Back-end

The back-end is composed by all the server-side code, frameworks, tools and database management systems that are running in the server where the web application is allocated. It receives requests from the front-end, processes the data, and sends responses back to the front-end.

3.1. Request Management

The general request management architecture is presented in Figure 1 . First, when any request is sent, it is received by NGINX, serving as reverse proxy. Then, NGINX sends the request to an specified port of the machine, where Gunicorn is listening. Gunicorn is a Python WSGI HTTP Server for UNIX, and it can replicate instances of our flask application, named as workers. The requests are forwarded to an specific instance of the Flask application, where the request is resolved.

3.1.1. NGINX web server

NGINX is a web server and reverse proxy that is fast, reliable, and secure. It can manage HTTP connections and traffic by using various web acceleration techniques like load balancing, SSL termination, connection and request policing, static content offload, and content caching. NGINX can also act as a secure application gateway that passes traffic from users to applications using built-in interfaces.

The reason to use NGINX as an application gateway is that it provides an all-in-one solution for HTTP connection management, load balancing, content caching, and traffic security. This makes it easier to manage and secure the application backend, improve scalability and performance, and build highly available applications by clustering application instances behind NGINX.

3.1.2. Gunicorn WSGI server

Gunicorn (short for "Green Unicorn") is a popular open-source Python Web Server Gateway Interface (WSGI) HTTP server. It allows Python web applications to be served by communicating with web servers like NGINX or Apache. Gunicorn is designed to be a lightweight and simple HTTP server that can handle multiple requests concurrently. Gunicorn is commonly used in combination with popular web frameworks like Flask, Django, Pyramid, and others to serve web applications with high performance and reliability.

We will use Gunicorn to replicate our Flask application, improving our API reliability by providing redundancy in case of server failures. Also, Gunicorn can improve the performance of a web application by using multiple worker processes to handle requests concurrently. This allows the application to handle a larger number of requests without slowing down or crashing. Scalability of the system must be taken into account, with Gunicorn we can easily adjust the number of workers, allowing more instances of the application to run concurrently.

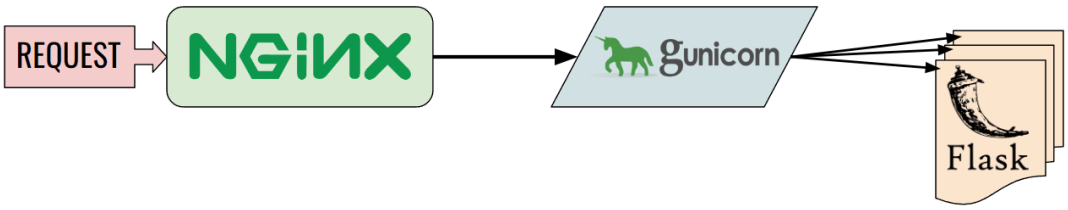

Figure 1: General API Architecture

3.2. Flask Application

Finally, the main Flask application will have all the endpoints and user management, it will be in charge of loading the models and the negation system, as well as connecting the database with the application and displaying the different views that will be available in the web.

Regarding the security aspect of our application, with Flask we can make use of different libraries to implement forms, data validators, session data, login, logout and CSRF protection:

- Flask-Login: Flask-Login is a Python library for Flask web applications that enables user authentication and authorization. This library helps in managing user sessions, securing user data, and restricting access to specific parts of the application for authenticated users only. Flask-Login allows defining a User model that represents the users in the application and implementing login and registration views to handle user authentication. It also enables protecting views and endpoints by requiring users to log in before accessing them.
- Flask-WTF: Flask-WTF is a Python library that simplifies the process of creating and validating web forms in Flask applications. With Flask-WTF, forms can be created using Python classes and rendered in HTML templates. It includes various form fields such as text fields, checkboxes, radio buttons, and dropdown menus, among others, and supports form validation that helps prevent common errors like missing required fields and invalid data types. Flask-WTF's built-in security features protect against CSRF attacks and enable secure file uploads.

3.3. Database system and database model

As database management system we will use MariaDB. MariaDB Server is a widely used open source relational database system that was created by the original developers of MySQL. It is designed to be high-performing, stable, and open-source, and is included as the default database system in most Linux distributions and many cloud services. To be able to communicate between the database and the Flask application, we will create a component called Database Model.

The Database Model is a class written in Python that implements all the necessary functions to communicate with the database. It builds the SQL statements needed to create, read, update and delete any element in the database. This way the connection and communication with the database is abstracted from the other components, leaving a cleaner and more maintainable code.

3.4. API security

To ensure the API security, we need to implement an authentication system in the web application and link it to the API requests, so that every call is identified. In each call to an endpoint where you need identification, a JSON Web Token must be added in the request header.

3.4.1. JSON Web Tokens

A JSON Web Token (JWT) is a secure method of transmitting information a JSON object between parties in web applications. It is a compact and self-contained format consisting of three parts: header, payload, and signature. The header includes information about the type of token and algorithm used to sign it, while the payload contains the claims or information that the token carries, such as user ID or role. The signature is a cryptographic signature that verifies the token's integrity and ensures it has not been tampered with. The stateless nature of JWTs means that NegesAPI do not need to track user sessions or login credentials and can rely on JWTs to validate a user's identity for each request.

NegesAPI implements JWTs as an API Key, provided in the */profile* page of each user, with an expiration date

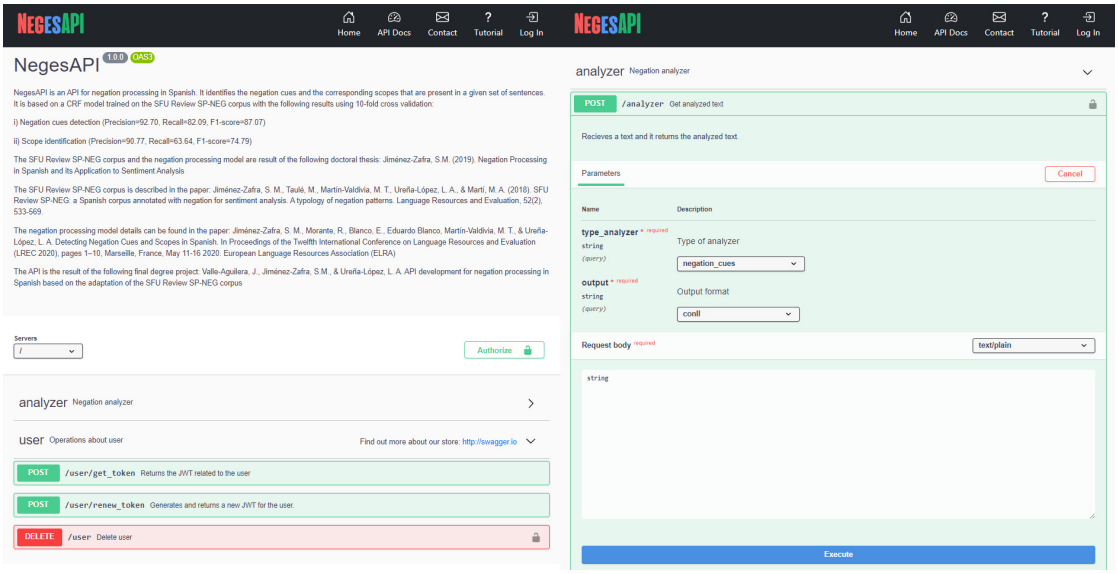

Figure 2: NegesAPI Documentation Page. On the Left, main documentation page. On the right, analyzer demo.

of 24 hours and the option to regenerate a new one. This can easily be implemented with the *jwt* Python module, and starting each authorized request with the header *"Authorization: Bearer userJwt"*, being *userJwt* the user's token.

4. Front-end

The code executed in the client-side, normally interpreted by web navigators, is called front-end. It is the responsible to provide an interface with which the user can interact. NegesAPI offers a simple and intuitive interface, with OpenAPI 3.0.0 documentation standards. In addition, it includes a test environment for the negation analyzer, a user profile tab and a tutorial on how to use the tool.

4.1. API documentation and test

The Documentation page (Figure 2) provides a easy to use test environment to try the different endpoints, the mandatory data, formats and response codes, without writing any code or command. This page is perfect to try the different output formats and it also provides a cURL example of how to create this request.

This documentation page has been generated using a Swagger tool, named Swagger Editor. Swagger is an open-source set of tools that provides a standardized way of documenting RESTful APIs. It comes with tools that can generate API documentation, perform API testing and debugging, and create client code in various programming languages. Swagger uses a machine-readable format called OpenAPI Specification, which describes the structure of the API, including endpoints, parameters, request/response formats, and authentication mechanisms.

4.2. Profile tab

The profile tab (Figure 3) contains all user information, such as name, email, password and API key. Here we can display, copy and regenerate our API key, change the user's password and delete the account.

The API Key is the JWT generated in the back-end, necessary to make API requests that needs identification.

4.3. Tutorial tab

NegesAPI also offers a step-by-step guide on how to use the API, either from the browser if you want to test it or if you want to implement it a programming language using the cURL tool.

It also adds code snippets, output examples and descriptions of the different analyzer options, negation annotations and output formats.

5. Negation system

NegesAPI uses an updated version of the corpus SFU Review SP-NEG [8] and the negation detection model of Jiménez-Zafra et al. [5].

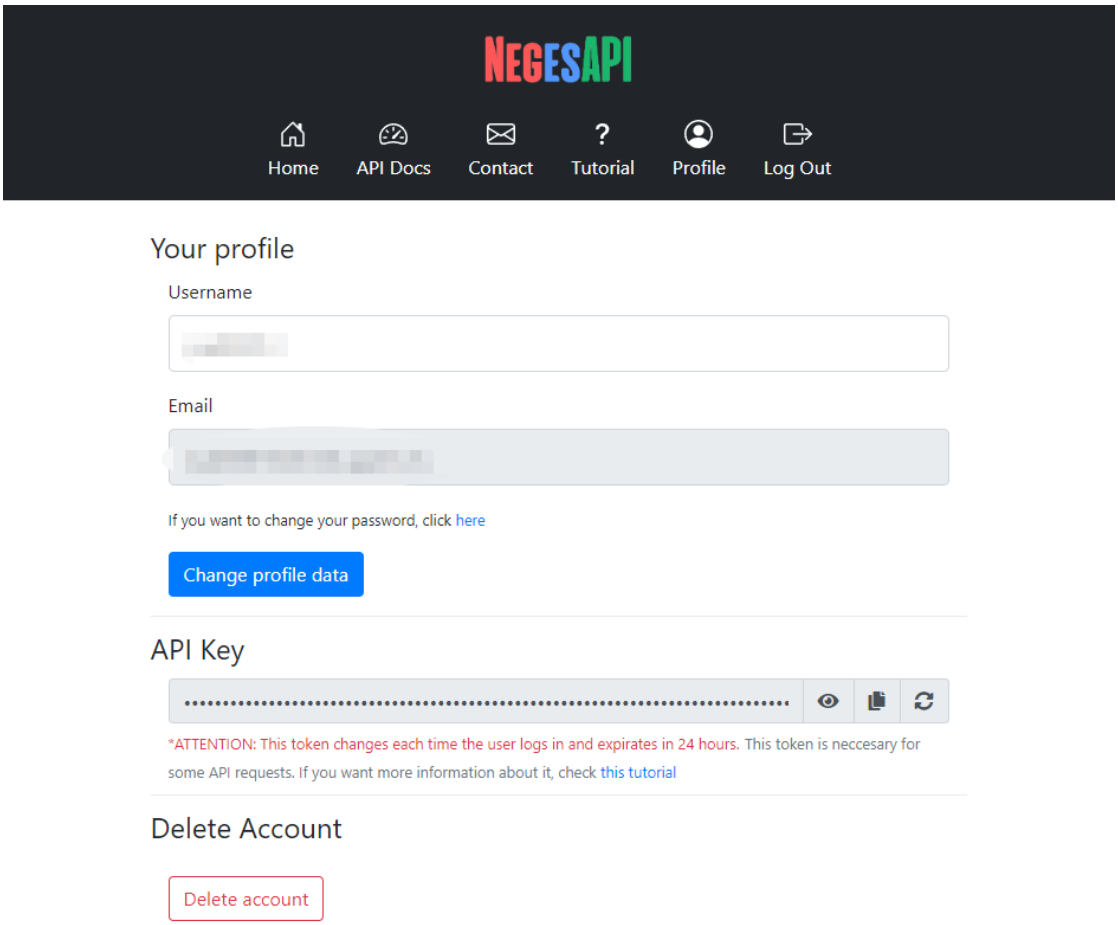

Figure 3: NegesAPI Profile page

The corpus SFU Review SP-NEG has been updated to match the latest improvements in FreeLing 4.2, since we use this tool as a feature extraction method. This corpus used a deprecated tag set generated with FreeLing 4.0 [9, 10], which needed to be updated in order to make the negation system work.

The negation detection system models the negation processing as a sequence labelling task, for that reason, it makes use of a CRF algorithm [11], as it has shown its effectiveness in this task [12, 13, 14]. This is because CRF makes predictions based on the elements in the sequence, not only in the current one, and negation cues and scopes behaves the same way. The results of its effectiveness using 10-fold cross validation are: i) Negation cues detection (Precision=92.70, Recall=82.09, F1-score=87.07), ii) Scope identification (Precision=90.77, Recall=63.64, F1-score=74.79).

More information on the corpus and the system development can be found in [15].

6. Conclusion and future works

NegesAPI is a powerful API that can assist different tasks in NLP and can be easily implemented in multiples scenarios, providing a fast and precise prediction of the negation cues and scopes in any Spanish text. In future works, we can improve NegesAPI so that it will add the negation cues in the scope, prefixing the negator to the negation word. It is also important that NegesAPI can be used in any type of algorithm and technology, therefore a future work would be to adapt the output suitable for Deep Learning algorithms. In addition, we plan to test its accuracy in different types of texts (reviews, posts, tweets, clinical texts, etc.). NegesAPI can provide a lot of relevant information to information retrieval systems, to detect negation in the searches that can be made, for

better effectiveness. This knowledge could be used, for example, to distinguish "Películas que sean de fantasía" from "Películas que no sean de fantasía", or any other search with negation.

Acknowledgements

This work has been partially supported by Project CONSENSO (PID2021-122263OB-C21), Project MODERATES (TED2021-130145B-I00) and Project SocialTox (PDC2022-133146-C21) funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR, Project PRECOM (SUBV-00016) funded by the Ministry of Consumer Affairs of the Spanish Government, Project FedDAP (PID2020-116118GA-I00) supported by MICINN/AEI/10.13039/501100011033 and WeLee project (1380939, FEDER Andalucía 2014-2020) funded by the Andalusian Regional Government. Salud María Jiménez-Zafra has been partially supported by a grant from Fondo Social Europeo and the Administration of the Junta de Andalucía (DOC_01073).

References

[1] S. M. Jiménez-Zafra, R. Morante, M. T. Martín-Valdivia, L. A. U. Lopez, Corpora annotated with negation: An overview, Computational Linguistics 46 (2020) 1–52.

[2] J. C. de Albornoz, L. Plaza, A. Díaz, M. Ballesteros, Ucm-i: A rule-based syntactic approach for resolving the scope of negation, in: * SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012), 2012, pp. 282–287.

[3] Y. Peng, X. Wang, L. Lu, M. Bagheri, R. Summers, Z. Lu, Negbio: a high-performance tool for negation and uncertainty detection in radiology reports, AMIA Summits on Translational Science Proceedings 2018 (2018) 188.

[4] N. P. Cruz, M. Taboada, R. Mitkov, A machine-learning approach to negation and speculation detection for sentiment analysis, Journal of the Association for Information Science and Technology 67 (2016) 2118–2136.

[5] S. M. Jiménez-Zafra, R. Morante, E. Blanco, M.-T. Martín-Valdivia, L. A. U. López, Detecting negation cues and scopes in spanish, in: Proceedings of the Twelfth Language Resources and Evaluation Conference, 2020, pp. 6902–6911.

[6] F. Fancellu, A. Lopez, B. Webber, Neural networks for negation scope detection, in: Proceedings of the 54th annual meeting of the Association for Computational Linguistics (volume 1: long papers), 2016, pp. 495–504.

[7] O. S. Pabón, O. Montenegro, M. Torrente, A. R. González, M. Provencio, E. Menasalvas, Negation and uncertainty detection in clinical texts written in spanish: a deep learning-based approach, PeerJ Computer Science 8 (2022) e913.

[8] S. M. Jiménez-Zafra, M. Taulé, M. T. Martín-Valdivia, L. A. Urena-López, M. A. Martí, Sfu review sp-neg: a spanish corpus annotated with negation for sentiment analysis. a typology of negation patterns, Language Resources and Evaluation 52 (2018) 533–569.

[9] L. Padró, E. Stanilovsky, Freeling 3.0: Towards wider multilinguality, in: LREC2012, 2012.

[10] M. Marimon, L. Padró, J. Turmo Borras, Coreference resolution in freeling 4.0, in: LREC 2018: 11th International Conference on Language, Resources and Evaluation: Miyazaki, Japan: May 7-12, 2018: proceedings book, 2018, pp. 376–381.

[11] J. Lafferty, A. McCallum, F. C. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001).

[12] I. Councill, R. McDonald, L. Velikovich, What's great and what's not: learning to classify the scope of negation for improved sentiment analysis (2010).

[13] H. Loharja, L. Padró, J. Turmo Borras, Negation cues detection using crf on spanish product review texts, in: NEGES 2018: Workshop on Negation in Spanish: Seville, Spain: September 19-21, 2018: proceedings book, 2018, pp. 49–54.

[14] L. Domınguez-Mas, F. Ronzano, L. I. Furlong, Supervised learning approaches to detect negation cues in spanish reviews, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019), CEUR Workshop Proceedings, Bilbao, Spain. CEURWS, 2019.

[15] J. Valle-Aguilera, S. M. Jiménez-Zafra, M. T. Martín-Valdivia, L. A. Ureña-López, NegesAPI: An API for Negation Cue Detection and Scope Identification in Spanish, Procesamiento del Lenguaje Natural 71 (2023).