Procesamiento del Lenguaje Natural, Revista nº 71, septiembre de 2023, pp. 97-108 recibido 30-03-2023 revisado 08-05-2023 aceptado 10-05-2023

NegesAPI: An API for Negation Cue Detection and Scope Identification in Spanish

NegesAPI: Una API para la Detección de Claves de Negación y para la Identificación de su Ámbito en Español

José Valle-Aguilera, Salud María Jiménez-Zafra, María Teresa Martín-Valdivia, L. Alfonso Ureña-López
Computer Science Department, SINAI, CEATIC
Universidad de Jaén, Campus Las Lagunillas, 23071, Jaén, Spain
`{jvalle, sjzafra, maite, laurena}@ujaen.es`

Abstract: This paper presents NegesAPI, an API for negation detection in Spanish. This API receives as input a text, analyzes it and automatically annotated it with the negation cues present in the text and their respective scopes of influence for their subsequent incorporation in Natural Language Processing systems. For the development of it, the SFU Review SP-NEG corpus and the negation model of Jiménez-Zafra et al. (2020a) have been updated. Moreover, it has been created a web application with the API documentation, a tutorial and a user management system to monitor the API usage.
Keywords: Negation, negation processing, cue detection, scope identification, natural language processing.

Resumen: Este artículo presenta NegesAPI, una API para la detección de negación en Español. Esta API recibe como entrada un texto, lo analiza y lo anota automáticamente con las claves de negación presentes en él y con sus respectivos ámbitos de influencia para su posterior incorporación en sistemas de Procesamiento del Lenguaje Natural. Para el desarrollo de la misma se ha actualizado el corpus SFU Review SP-NEG y el modelo de negación de Jiménez-Zafra et al. (2020a). Además, se ha creado una aplicación web con la documentación de la API, un tutorial y un sistema de gestión de usuarios para monitorizar el uso de la API.
Palabras clave: Negación, procesamiento automático de la negación, detección de claves, identificación de ámbito, procesamiento del lenguaje natural.

1 Introduction

Negation is a grammatical phenomenon that is present in all languages to express the negative form of a sentence or statement (Jiménez-Zafra et al., 2020b). It relates one expression e with another that is in some way opposed to the meaning of e (Horn and Wansing, 2015). Negation can be accomplished through a variety of elements, including negation particles, negative affixes or negative words.

- Negation particles are related with syntactically independent elements, such as *no, nunca, nadie,...* This type of negation is called syntactic negation and is the most common in all languages.
- Negative affixes are morphemes that are added to a word to create a negative form, for example: ***im**pensable, **i**legible, **im**posible,...* This form of negation is known as lexical negation.
- Negative words are words or expressions that implies a negative sense. This is called morphological negation. For example, the expression *en la vida* (In my life) implies a negation of the next action: ***En la vida** cruzaré ese puente* (In my life I will cross that bridge).

Negation in Spanish can also be found in multiple ways, including the use of simple, continuous and discontinuous negation. In simple negation, there is only one negation cue: ***No** sé cocinar* (I don't know how to cook). Continuous negation appears when there are two or more consecutive negation cues: ***Casi no** llego a la presentación* (I almost don't make it to the presentation). Finally, when two or more negation cues appear but not consecutively, it is called discontinuous negation: ***No*** *tengo* ***nada*** *de dinero para la comida* (I don't have money for the food).

Moreover, in some cases, the appearance of a negation cue does not imply a negation. This phenomenon is very common in Spanish, meaning comparative or rhetoric sentences:

- Estás preparado, ¿**no**?
  Are you prepared, right?
- **No** hay **más que** dar la vuelta.
  Just turn around.
- **No** estudio **tanto como** mi hermano.
  I don't study as much as my brother.

Therefore, negation can be very challenging to detect and process (Jiménez-Zafra, 2019). 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 different approaches to handle negation in NLP, such as rule-based approaches (de Albornoz et al., 2012; Peng et al., 2018), and traditional machine learning (Cruz, Taboada, and Mitkov, 2016; Jiménez-Zafra et al., 2020a) and deep learning approaches (Fancellu, Lopez, and Webber, 2016; Pabón et al., 2022). 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.

The goal of this work is to provide an API, called NegesAPI, to detect negation cues and scopes in Spanish texts, using the machine learning approach of Jiménez-Zafra et al. (2020a). The main contributions of this work are:

1. Update of the SFU review SP-NEG corpus (Jiménez-Zafra et al., 2018): This corpus uses a deprecated tag set generated with FreeLing 4.0 (Padró and Stanilovsky, 2012; Marimon, Padró, and Turmo Borras, 2018), which needs to be updated to the most recent version, FreeLing 4.2.
2. Generation of pre-trained models: Using the updated corpus, we need to adapt the cue and scope detector models of Jiménez-Zafra et al. (2020a) so that they can be loaded in the negation system.
3. Building an API: Using the negation system and the pre-trained models, we have created an API that receives a text, analyzes it, and returns the text automatically annotating it with negation.
4. Building a web application: On top of the API, we have built a simple web application with the API documentation, a tutorial and a user management system to monitor the API usage.

The rest of the paper is organized as follows: Section 2 presents previous negation tools. Section 3 provides a description of the corpus SFU Review SP-NEG and the negation system used by the NegesAPI. In Section 4, the methodology followed for the development of the NegesAPI is shown. In Section 5 it is described the API calls and how to use the NegesAPI. Finally, the conclusion and future improvements or functionalities are set out in Section 6.

2 Background

To the best of our knowledge, there is only a previous Spanish negation tool, called NegEx-MES[1] (Sanamaría, Jesús, 2019), which is based on the rule-based NegEx algorithm (Chapman et al., 2001). It is a negation detection system, implemented in Java, to detect negation in clinical texts written in Spanish. NegEx-MES detects if a given term is negated or not within a given sentence. If it is negated, it returns the type of the modifier: *negPhrases*, for adverbs, negative predicates

[1]`https://github.com/PlanTL-GOB-ES/NegEx-MES`

or prepositions; *postNegPhrases*, for words that negate but also includes some doubt, *pseNegPhrases*, as previous one but including the doubt directly and *conjunctions*, for adversative conjunctions. It should be noted that it returns the type of the modifier but not the negation cue.

In languages other than Spanish, we find other tools such as NegEx[2] (Chapman et al., 2001). It is a rule-based system that is available in Python and Java. NegEx receives a sentence in English and, optionally, a clinical condition and returns two types of outputs. If a clinical condition is specified in the input, returns whether the condition is negated or possible. If no condition is specified, returns the scope of the negation cues present in the sentence. There is another tool, called DEEPEN[3] (Mehrabi et al., 2015), which is based on NegEx, but takes into account the dependency relationship between negation words and concepts within a sentence to reduce the number of incorrect negation assignments. DEEPEN is implemented in Java. It receives a sentence with a clinical condition and returns the negation status of the given concept in the sentence. The negation status can be: *Affirmed*, if the concept is not negated, *Affirmed confirmed*, if NegEx found the concept negated but DEEPEN considers it affirmed, and *Negation confirmed*, if both NegEx and DEEPEN found the concept negated.

There are also other English Python tools, such as the library "negation-detection"[4] (Gkotsis et al., 2016). It is a rule-based tool that detects the negation associated with a Mental Health term, such as suicide, in a given sentence written in English. It returns *True* for affirmed terms, *False* for negated and *None* for terms not found. We also find Negtool[5] (Enger, Velldal, and Øvrelid, 2017), a machine learning tool to detect negation cues and scopes in raw or parsed texts in English, also available for Python. Finally, there is also a negation detection pipeline[6] integrated in spaCy tool (Honnibal and Montani, 2017), which is based on NegEx algorithm. With *negspacy* we can determine whether Named Entities are negated or not in a given English text.

As we can see, despite the amount of articles and research on this phenomenon, we do not find many tools available in languages other than English.

Most of the existing tools are designed for the clinical domain and focus on detecting whether a term is negated or not. NegesAPI does not focus exclusively on a text domain, instead it can be applied to any desired domain because it is trained with texts from different areas. In addition, it is a machine learning-based tool, in contrast to the rule-based tools mentioned above, with the exception of Negtool. Finally, it should be noted that existing tools do not provide the full scope of a negation cue (except Negtool). However, NegesAPI automatically detects the negation cues and scopes without the need of specify a term on which to perform the analysis. Although Negtool is a machine learning tool and identify negation cues and scopes, it is designed for English and NegesAPI is specialized in Spanish, in which negation can be found in more different ways (Martí et al., 2016) as it has been explained in the Introduction section.

3 *Corpus and Negation Detection System*

NegesAPI uses the SFU Review SP-NEG corpus (Jiménez-Zafra et al., 2018) and the negation detection model of Jiménez-Zafra et al. (2020a), which will be described below.

3.1 Corpus SFU Review SP-NEG

This corpus is composed of 221,866 words and 9455 sentences, out of which 3022 sentences contain at least one negation structure. It consists of 400 reviews divided into eight groups of 50 files each. Each group has 25 negative and 25 positive reviews containing a user's opinion about a product. The corpus has annotations on negation cues, scopes, and events. It also shows how negation affects the words in its scope, such as changes in polarity or value, which is useful for sentiment analysis systems. Additionally, the corpus was annotated with morphological information, such as lemma and PoS tags using Freeling 4.0 (Carreras et al., 2004).

The reviews are organized in 8 files classified in several domains: cars, hotels, wash-

[2] `https://code.google.com/archive/p/negex/`

[3] `http://svn.code.sf.net/p/ohnlp/code/trunk/DEEPEN/`

[4] `https://github.com/gkotsis/negation-detection`

[5] `https://github.com/marenger/negtool`

[6] `https://spacy.io/universe/project/negspacy`

```
hoteles_no_2_6  20   1 La   el   da0fs0    article    spec   2 -    La    -
hoteles_no_2_6  20   2 calidad calidad ncfs000 common   suj  4    -  calidad  -
hoteles_no_2_6  20   3 no   no   rn    negative   mod   4 no  no   -
hoteles_no_2_6  20   4 corresponde corresponder    vmip3s0    main   sentence  0    -  corresponde  corresponde
hoteles_no_2_6  20   5 a    a sps00   preposition   creg  4    -  a    -
hoteles_no_2_6  20   6 un   uno  di0ms0   indefinite    spec   8 -   un    -
hoteles_no_2_6  20   7 4    4 z    - z    6 -    4 -
hoteles_no_2_6  20   8 estrellas   estrella  ncfp000  common   sn    5  -    estrellas  -
hoteles_no_2_6  20   9 .    . fp   -    f  4    - -    -
```

Figure 1: Corpus output example.

ing machines, books, mobile phones, music, computers and movies. Each file contains all reviews associated to a domain annotated in CoNLL format (Farkas et al., 2010). Each review can have one or more sentences, and one sentence can have one or more tokens. Each token corresponds to a row in the file and its information is distributed in columns. Now we will describe more in depth the rows and columns of the corpus.

3.1.1 Corpus rows

Each row corresponds to a token, each one of the words or symbols in which the sentences of a comment are separated. Each row has multiple columns with the token info, this will allow us to analyze the dependencies and relations that exist between the tokens of each sentence. The sentences are separated with a blank row, and each comment is also separated with a blank row.

3.1.2 Corpus columns

Each row is composed by a minimum of 10 columns. Each column corresponds to the data associated to the token in row. The reason for not having a fixed number of columns is the negation info, which will be described later in this section. In Figure 1 we can see an example of the corpus. The meaning of each of the columns is described below:

1. Domain filename: File where the comment is located. A comment has the same Domain filename in all its tokens and sentences. It allows us to determine if a sentence belongs to one comment or another.
2. Sentence Number within Domain filename: Order of the sentence within Domain filename. It starts at 1, increasing by one number if the following sentence belongs to the same comment. If not, it restarts at 1.
3. Token number within sentence: Token order within the sentence. It starts at 1, and behaves as above, increasing by 1 in the next row if the next token belongs to the same sentence.
4. Token: original word or symbol.
5. Lemma: It is the simplest form in which the word can be represented.
6. Part of Speech: Tag that represents the grammatical category.
7. Part of Speech type: Grammatical category type.
8. Dependency relation: Relation type that this token has with the token id marked in Dependency head.
9. Dependency head: Id of the token with which the current token maintains a dependency relationship.
10. (to last) Negation: Negation info, it can be one of the following:
    (a) Three asterisks (***) if negation is not present in the sentence.
    (b) Three tokens (*cue scope event*) separated by tabulators if negation is present in the sentence, where "cue" is the lemma of the negation cue and "scope" is the lemma of the token that belongs to the scope of the negation.
    (c) More than three tokens -but always multiple of three- (cue1 scope1 event1 cue2 scope2 event2 ...), if there is more than one negation key in the sentence.

3.2 Negation detection system

The negation detection system models the negation processing as a sequence labelling task, making use of the CRF algorithm (Lafferty, McCallum, and Pereira, 2001), as it has shown its effectiveness in this task (Councill, McDonald, and Velikovich, 2010; Loharja, Padró, and Turmo Borras, 2018; Domınguez-Mas, Ronzano, and Furlong, 2019). 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.

3.2.1 System description

The CRF algorithm is implemented using CRFsuite (Okazaki, 2007) and scikit-learn (Pedregosa et al., 2011), and has the default training algorithm, the LBFGS, and Elastic Net (L1 + L2) as regularization. Specifically, the system trains two classifiers. The first one takes as input a sentence and predicts the negation cue BIO labels to decide whether each token in the sentence is the beginning of a cue (B), the inside (I), or no cue (O). The second one takes as input a sentence along with information about the predicted cues and predicts the scope BIO labels identifying which of the tokens in the sentence are at the beginning (B), inside (I) or outside (O) the scope of the cue. Each token is represented with a set of features for each task by a selection process conducted by the authors of the negation processing system (Jiménez-Zafra et al., 2020a).

The feature set for the negation cue detection classifier is composed of 31 features: lemma and PoS tag of the token in focus as well as those of the seven tokens before and after it (features 1-30), and a string value stating whether the token in focus is part of any cue in the training set and whether it appears as the first token of a cue in the training set (B), as any token of a cue except the first (I), as both the first token of a cue and other positions (B_I), or if it does not belong to any cue of the training set (O) (feature 31). The motivation of this last feature is that many cues appear as a single token (e.g., *ni*, 'neither') and are also part of multiword cues (e.g., *ni siquiera*, 'not even').

The feature set for detecting the scope of the negation cues consists of 24 features: lemma and PoS tag of the current token and the cue in focus (features 1-4), location of the token with respect to the cue (feature 5) (before, inside or after), distance in number of tokens between the cue and the current token (feature 6), chain of PoS tags and chain of types between the cue and the token (features 7-8), lemma and PoS tags of the token to the left and right of the token in focus (features 9-12), relative position of the cue and the token in the sentence (features 13-14), dependency relation and direction (head or dependent) between the token and the cue (features 15-16), PoS tags of the first and second order syntactic heads of the token (features 17-18), whether the token is ancestor of the token and vice versa (features 19-20), dependency shortest path from the token in focus to the cue and vice versa (features 21-22), dependency shortest path from the token in focus to the cue including direction (up or down) (feature 23), and length of the short path between the token and the cue (feature 24).

3.2.2 System evaluation

The evaluation of the negation system is performed in terms of precision (P), recall (R) and F1-score (F) measures using the SFU review SP-NEG corpus and 10-fold cross-validation. In Table 1 we can see the result values for negation cue detection and scope identification.

4 *NegesAPI development: Methodology*

The development of the NegesAPI has been divided into different stages. We will briefly describe all of them and next, each stage in more depth:

1. Updating the corpus: It is necessary to update the corpus SFU Review SP-NEG for the negation system to work. The negation system uses FreeLing 4.2, and the corpus is generated with a previous version, FreeLing 4.0.

2. Generating the models: The negation system will make use of pre-trained models created with the updated corpus, so the next step is to generate the different lists and corpus formats the model generation tool needs.

3. Definition of the API requirements: Before starting the API development, it's essential to determine the API requirements. This includes defining the functionality, endpoints, parameters, response codes, and any other details required for the API.

4. Choosing the technology stack: Based on the requirements and previous works, such as the negation system of Jiménez-Zafra et al. (2020a), we need to choose the appropriate technology stack for building the API. This includes selecting the programming language, frameworks,

	Cue			Scope		
	P	R	F	P	R	F
Books	87.67	81.24	84.33	84.69	63.23	72.40
Cars	93.01	82.10	87.22	90.65	59.88	72.12
Cell phone	95.51	84.83	89.85	94.12	63.87	76.10
Computers	94.43	83.93	88.43	91.59	64.26	75.53
Hotels	92.97	80.83	86.48	91.47	65.561	76.38
Movies	91.84	81.73	86.49	89.96	65.06	75.51
Music	91.98	79.89	85.51	89.34	58.45	70.67
Washing machines	95.19	82.20	88.22	94.31	68.84	79.59
All	92.70	82.09	87.07	90.77	63.64	74.79

Table 1: System results for cue and scope detection using 10-fold cross-validation.

libraries, and any other tools needed for development.

5. API architecture design: Once the technology stack is decided, it's time to design the API architecture. This involves creating a high-level design of the API, including its components, interfaces, and data models.

6. Building the API negation system: Now we need to load the models previously generated and build the negation system in the analyzer endpoint, taking in mind to return the analyzed data in the correct format and analyzer type.

7. API security: API security is crucial to protect the data being exchanged between the API and its consumers. It is important to implement security measures such as authentication and encryption to ensure the API is secure.

8. API release: Now the API is ready to be released to production so that everyone can try it out.

4.1 Updating the corpus

In order to analyze a text with the negation detection system, the key points are: the pre-trained models with the SFU Review SP-NEG corpus, the machine learning classifiers for negation and scope detection, and a method to represent each token with the features required by each classifier. The pre-trained models and the feature extraction method have to use the same set of tags. The SFU Review SP-NEG corpus, as previously stated, was automatically annotated with FreeLing 4.0, but the tag set of Freeling tool has been modified in the latest version, FreeLing 4.2, so it is necessary to update the SFU Review SP-NEG tags in order to correctly generate the pre-trained models and use them to predict the negation info.

To update the corpus we have analyzed it with Freeling 4.2, obtaining the new tags but without losing the manual annotations related to negation. For this, we have created one file per domain with the tokenized texts, leaving only one token in each line and we have used the following Freeling command, which works with tokenized files:

```
> analyze.bat --inplv splitted
--input freeling -d txala -f es.cfg
--output conll
--outlv dep < input\_file.txt
```

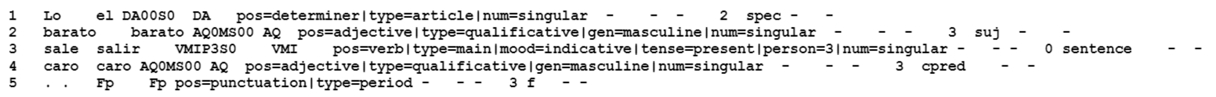
4.1.1 FreeLing Output

The output format provided by FreeLing 4.2 is different from the format of the annotated corpus, but it has all the information we need to update the corpus.

As we can see in Figure 2, despite of having a different output format, we can obtain the fields Lemma (3rd column), PoS (4th column) PoS type (6th column, attribute type), Dependency head and Dependency relation (10th and 11th respectively)

4.1.2 Generating the new corpus

To generate the new version of the corpus we have to merge the information that is present in the corpus with the new information retrieved from FreeLing 4.2. However, this join is not as simple as it may seem since FreeLing 4.2 improved the analyzer compared to FreeLing 4.0. In Figure 3 we can see how the same sentence has two ways of expressing token relationships, what leads to inconsisten-



```
1   Lo    el DA00S0  DA   pos=determiner|type=article|num=singular  -    - -    2  spec -   -
2   barato    barato AQ0MS00 AQ  pos=adjective|type=qualificative|gen=masculine|num=singular  -    - -    3  suj  -    -
3   sale  salir    VMIP3S0    VMI    pos=verb|type=main|mood=indicative|tense=present|person=3|num=singular -   - -   0 sentence    -  -
4   caro  caro AQ0MS00 AQ  pos=adjective|type=qualificative|gen=masculine|num=singular  -    - -    3  cpred    - -
5   . .   Fp    Fp pos=punctuation|type=period -   - -   3 f    - -
```

Figure 2: FreeLing output example.

cies. In this case, the corpus breaks down this sentence in 26 tokens, while FreeLing 4.2 breaks it down in 25 tokens. The token that is "missing" is located in line 22, where FreeLing 4.2 joins *15* with *días* in a single row: *15_días*, as a temporal marker: *d:15* (Lemma in 3rd column).

However, this is not the only type of inconsistency we found, we also discovered other types of problems:

- Tokens that are joined: It is the result of two tokens joining to represent a single token. This problem appears with temporal markers: *"4 minutos", "5 HORAS", "veinte días"*; length measurements: *"mil kilómetros", "10.000 km"*; or even currency markers: *"4.000.000 de pesetas", "300 euros"*. This change will result in less rows in the updated corpus.
- Tokens that are split: It occurs when tokens are split, such as the contractions: *"del" (de + el), "al" (a + el)*. This case separates a token in two, allowing its analysis separately. This will result in a new line in the updated corpus.
- Tokens with negation annotations: When two tokens are joined or a token is separated into two and they have annotations on negation, the annotation of those tokens must be checked. If two tokens are joined, the columns representing whether the token is cue or scope will become the lemma resulting from the join. On the other hand, if one token is split into two, the negation information should be reflected in the two new tokens with their respective lemmas.

4.2 Generating the models

NegesAPI has the option to analyze and detect negation cues, or negation cues and scopes, so we created a model for each analyzer type trained on the annotated data from the updated version of the SFU Review SP-NEG corpus.

For the generation of the pre-trained model for negation cue detection:

1. The negation cue annotations of the corpus in CoNLL format are converted to BIO format.
2. Three lists containing the negation tokens present in the training corpus are compiled, which are used to generate feature 31.
   - B_cue_list, for tokens appearing as the first token of a cue (B).
   - I_cue_list, for tokens that are part of a negation cue but not are the first token (I). This is useful for continuous and discontinuous negation cues.
   - B_I_cue_list, for tokens that can appears both at the beginning of or inside a negation cue (B_I). This is useful for tokens that appears as a single cue and as part of multiword cues.
3. Finally, a pre-trained CRF model fitted with the negation cue features of the training corpus is generated.

For the development of the pre-trained model for scope identification:

1. The scope annotations of the corpus in CoNLL format are converted to BIO format also linking the information of the negation cue to which each scope is related to.
2. Finally, a pre-trained CRF model fitted with the scope features of the training corpus is generated.

4.3 API requirements

NegesAPI needs to be light and fast, ready to analyze any text. The user can choose the type of analyzer, between detection of negation cues or detection of negation cues and scopes. In addition, the user can choose the output format, between CoNLL and BIO. In order to monitor the usage of the API, the users must be authenticated, so the web page must have a registration, login and user management. In addition, both the API and the

```
#CORPUS EXAMPLE
1       En      en      sps00   preposition
2       el      el      da0ms0  article
3       primer  1       ao0ms0  ordinal
4       año     año     ncms000 common
5       lo      lo      pp3cna00 personal
6       tuve    tener   vmis1s0 main
7       que     que     cs      subordinating
8       llevar  llevar  vmn0000 main
9       4       4       z       -
10      veces   vez     ncfp000 common  cc
11      a       a       sps00   preposition
12      el      el      da0ms0  article
13      taller  taller  ncms000 common
14      y       y       cc      coordinating
15      cada    cada    di0cs0  indefinite
16      vez     vez     ncfs000 common
17      que     que     pr0cn000 relative
18      lo      lo      pp3cna00 personal
19      llevé   llevar  vmis1s0 main
20      me      me      pp1cs000 personal
21      quedé   quedar  vmis1s0 main
22      15      15      z       -
23      días    día     ncmp000 common
24      sin     sin     sps00   preposition
25      coche   coche   ncms000 common
26      .       .       fp      -
```

```
#FREELING EXAMPLE
1       En      en      SP      SP
2       el      el      DA0MS0  DA
3       primer  1       AO0MS00 AO
4       año     año     NCMS000 NC
5       lo      lo      PP3MSA0 PP
6       tuve    tener   VMIS1S0 VMI
7       que     que     CS      CS
8       llevar  llevar  VMN0000 VMN
9       4       4       Z       Z
10      veces   vez     NCFP000 NC
11      a       a       SP      SP
12      el      el      DA0MS0  DA
13      taller  taller  NCMS000 NC
14      y       y       CC      CC
15      cada    cada    DI0CS0  DI
16      vez     vez     NCFS000 NC
17      que     que     PR0CN00 PR
18      lo      lo      PP3MSA0 PP
19      llevé   llevar  VMIS1S0 VMI
20      me      me      PP1CS00 PP
21      quedé   quedar  VMIS1S0 VMI
22      15_días TM_d:15 Zu      Zu
23      sin     sin     SP      SP
24      coche   coche   NCMS000 NC
25      .       .       Fp      Fp
```

Figure 3: Sentence differences between corpus and FreeLing 4.2.

web application must be secure against any the most common cyber-attacks, and generally robust against any unwanted connection or behaviour. At the moment, NegesAPI does not need to have a large amount of endpoints, and they should not be too complex to develop, so we will use the standard OpenAPI 3.0.0 specification to make an scheme of the API requests, parameters and response codes, and to easily generate the proper API documentation. To do this in the simplest and efficient way, we will use 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.

4.4 Choosing the technology stack

The most used programming language in NLP and text processing is Python, as it is powerful, simple and has a lot of libraries and tools to make NLP tasks easier. There are also two main frameworks written in Python that we can use for building an API, Flask and FastAPI. Both frameworks are perfectly acceptable options, but only Flask is particularly suited to the requirements set out above.

Flask depends on Jinja templates, one of the widest used in Python. It also adds auto-scaping, preventing XSS attacks and SQL injections. Flask also provides the toolset of Werkzeug, the WSGI (Web Server Gateway Interface) to describe how the server communicates to the web application, what allows to have ready our server with only a few steps. Flask can be easily scalable and maintainable, adding more functionalities and endpoints in a comfortable and easy way.

Although Flask is more complex than FastAPI, it allows us a wide window of possibilities and libraries to build the desired application. On the other hand, it is important the scalability and maintenance of the tool, as well as its reliability and security. It is also important to take in mind that Swagger Editor can generate the backbone of our API in Flask with a single click, once that we have our documentation ready.

4.5 API architecture

We can explain the general architecture of the API by following the modules that a request goes through. The main modules are: NGINX, Gunicorn and the Flask application.

First, when any request is sent, it is received by NGINX, serving as reversed proxy and application gateway. Then, NGINX sends the request to an specified port of the machine, where Gunicorn is listening. With Gunicorn we can replicate our Flask application as many times as our system can manage. The requests are forwarded to an specific instance of the Flask application, where the request is resolved. 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.

For more technical information about NegesAPI, more details are provided in the work of Valle-Aguilera et al. (2023).

4.6 Building the API negation system

The negation system must be able to analyze the cues or the cues and scopes of a text. In addition, each analyzer must return the output in a specified format: CoNLL or BIO. It is important to take into account the size of the texts that can be sent. In order to analyze and convert formats, we need to think a way to store the files in a safe but not persistent location.

To temporally store the texts to be analyzed, we will use TemporaryFile interface from tempfile module. This module creates temporary files and directories and works on all supported platforms. All the interfaces that this module implements provide automatic cleanup and can be used as context managers.

Each time a new request is received, we create a new temporary file and store the file name. Then, every modification that is made to the text is made in the same file. With this approach, we ensure that the files of previous requests are destroyed and does not stay in the server memory.

4.7 API security

To ensure the API security, we need to implement an authentication system, as well as encrypt the sensible information of the database. In each call to an endpoint where you need authorization a JSON Web Token (JWT) must be added in the request header. 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 and authorization for each request.

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

4.8 API Release

NegesAPI is running in a personalized server owned by the University of Jaén, at CEATIC facilities. To provide the better user experience and prevent saturation, we run the Flask app in a WSGI server, such as Gunicorn, and NGINX as app gateway and reverse proxy.

5 *How to use the negation API*

NegesAPI is public at `http://s5-ceatic.ujaen.es:8081/` and can be accessed by anyone, only registration is required. Now, we will describe all the requests the API has:

- /analyzer: Receives a text, the analyzer type and the output format. This request returns the text analyzed with the negation annotated, in the output format specified. You must be authenticated via API Key.
- /get_token: Receives the user email and password and returns the Key without generating another.
- /renew_token: Receives the user email and password and returns a new Key.
- /delete_user: Deletes the user that makes the request. You must be authenticated via API Key.

The Key is a token that authenticates the user and allow the previous calls. The Key has an expiration date of 24 hours, starting when the token is created.

As an example of an API call, we will use the Python programming language, as it widely used in NLP.

The first step is to obtain the API Key, so we need to register in the NegesAPI web page (`http://www.s5-ceatic.ujaen.es/register`). Now that we have our email and password, we can obtain the Key in our profile (`http://www.s5-ceatic.ujaen.es/`

`profile`) or we can make a request to the API.

The following code in Python creates a file called *token.txt* in which we are going to store the API response. Please note that the variable *data* must be modified to make this code work as intended.

```
from urllib.parse import urlencode
import pycurl
from io import BytesIO

file = open('token.txt', 'wb')
crl = pycurl.Curl()
crl.setopt(crl.URL,
    'http://s5-ceatic.ujaen.es:8081/
    user/get_token')
data = {'email': 'test@negesapi.es',
    'password':'my.very.secret.password'}
pf = urlencode(data)

crl.setopt(crl.POSTFIELDS, pf)
crl.setopt(crl.WRITEDATA, file)
crl.perform()
crl.close()
file.close()
```

The next step is to prepare our text to be analyzed. It must be in plain text, and it is recommended to end each sentence with a punctuation mark. For example, we will create a file called *data.txt* and its content is: "*No me gusta la comida.*". Now, we are ready to analyze our text. We only need to make the following call:

```
import pycurl

output_file = open('output.txt', 'wb')
token = open('token.txt', 'r').read()
crl = pycurl.Curl()
type_analyzer = "negation_cues"
output = "conll"
crl.setopt(crl.URL,
    'http://s5-ceatic.ujaen.es:8081/
    analyzer?type_analyzer='+
    type_analyzer+'&output='+output)
crl.setopt(crl.HTTPHEADER,
    ['accept: text/plain',
    'Authorization: Bearer '+token])
crl.setopt(crl.HTTPPOST, [
    ('body', (
        crl.FORM_FILE, 'data.txt',
        crl.FORM_FILENAME, 'data.txt',
        crl.FORM_CONTENTTYPE, 'text/plain',
    )),
])
crl.setopt(crl.WRITEDATA, output_file)
crl.perform()
crl.close()
output_file.close()
```

The output is generated in a file called *output.txt*, which contains the text analyzed in the specified output.

6 Conclusions and future work

NegesAPI is a powerful API that can help in different NLP tasks and can be easily used to provide a fast and accurate prediction of the negation cues and scopes in any Spanish text. We plan to use NegesAPI in future work and test its applicability in different domains and different types of texts (reviews, posts, tweets, etc.). We also plan to improve it with more output formats, for example, in text format by prefixing the negation cue to all the words in the scope. Moreover, we could also study the most suitable output for Deep Learning algorithms.

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

Carreras, X., I. Chao, L. Padró, and M. Padró. 2004. Freeling: An open-source suite of language analyzers. In *LREC*, pages 239–242. Citeseer.

Chapman, W. W., W. Bridewell, P. Hanbury, G. F. Cooper, and B. G. Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of biomedical informatics*, 34(5):301–310.

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

Cruz, N. P., M. Taboada, and R. Mitkov. 2016. A machine-learning approach to negation and speculation detection for sentiment analysis. *Journal of the Association for Information Science and Technology*, 67(9):2118–2136.

de Albornoz, J. C., L. Plaza, A. Díaz, and M. Ballesteros. 2012. 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)*, pages 282–287.

Domınguez-Mas, L., F. Ronzano, and L. I. Furlong. 2019. 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.*

Enger, M., E. Velldal, and L. Øvrelid. 2017. An open-source tool for negation detection: a maximum-margin approach. In *Proceedings of the Workshop Computational Semantics Beyond Events and Roles*, pages 64–69, Valencia, Spain, April. Association for Computational Linguistics.

Fancellu, F., A. Lopez, and B. Webber. 2016. Neural networks for negation scope detection. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics (volume 1: long papers)*, pages 495–504.

Farkas, R., V. Vincze, G. Móra, J. Csirik, and G. Szarvas. 2010. The conll-2010 shared task: learning to detect hedges and their scope in natural language text. In *Proceedings of the fourteenth conference on computational natural language learning–Shared task*, pages 1–12.

Gkotsis, G., S. Velupillai, A. Oellrich, H. Dean, M. Liakata, and R. Dutta. 2016. Don't let notes be misunderstood: A negation detection method for assessing risk of suicide in mental health records. In *Proceedings of the third workshop on computational linguistics and clinical psychology*, pages 95–105.

Honnibal, M. and I. Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Horn, L. R. and H. Wansing. 2015. Negation.

Jiménez-Zafra, S. M. 2019. *Negation processing in Spanish and its application to sentiment analysis*. Ph.D. thesis, Universidad de Jaén.

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

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

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

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

Loharja, H., L. Padró, and J. Turmo Borras. 2018. 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*, pages 49–54.

Marimon, M., L. Padró, and J. Turmo Borras. 2018. 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, pages 376–381.

Martí, M. A., M. Taulé, M. Nofre, L. Marsó, M. T. Martín-Valdivia, and S. M. Jiménez-Zafra. 2016. La negación en español: análisis y tipología de patrones de negación. *Procesamiento del Lenguaje Natural*, (57):41–48.

Mehrabi, S., A. Krishnan, S. Sohn, A. M. Roch, H. Schmidt, J. Kesterson, C. Beesley, P. Dexter, C. M. Schmidt, H. Liu, et al. 2015. Deepen: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of biomedical informatics*, 54:213–219.

Okazaki, N. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

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

Padró, L. and E. Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality. In *LREC2012*.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

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

Sanamaría, Jesús. 2019. NegEx-MES (v1.0.1). Zenodo. `https://doi.org/10.5281/zenodo.2542567`.

Valle-Aguilera, J., S. M. Jiménez-Zafra, M. T. Martín-Valdivia, and L. A. Ureña-López. 2023. Describing the architecture of NegesAPI, an API for Spanish negation processing. In *SEPLN-PD 2023: Annual Conference of the Spanish Association for Natural Language Processing 2023: Projects and Demonstrations*. CEUR Workshop Proceedings.