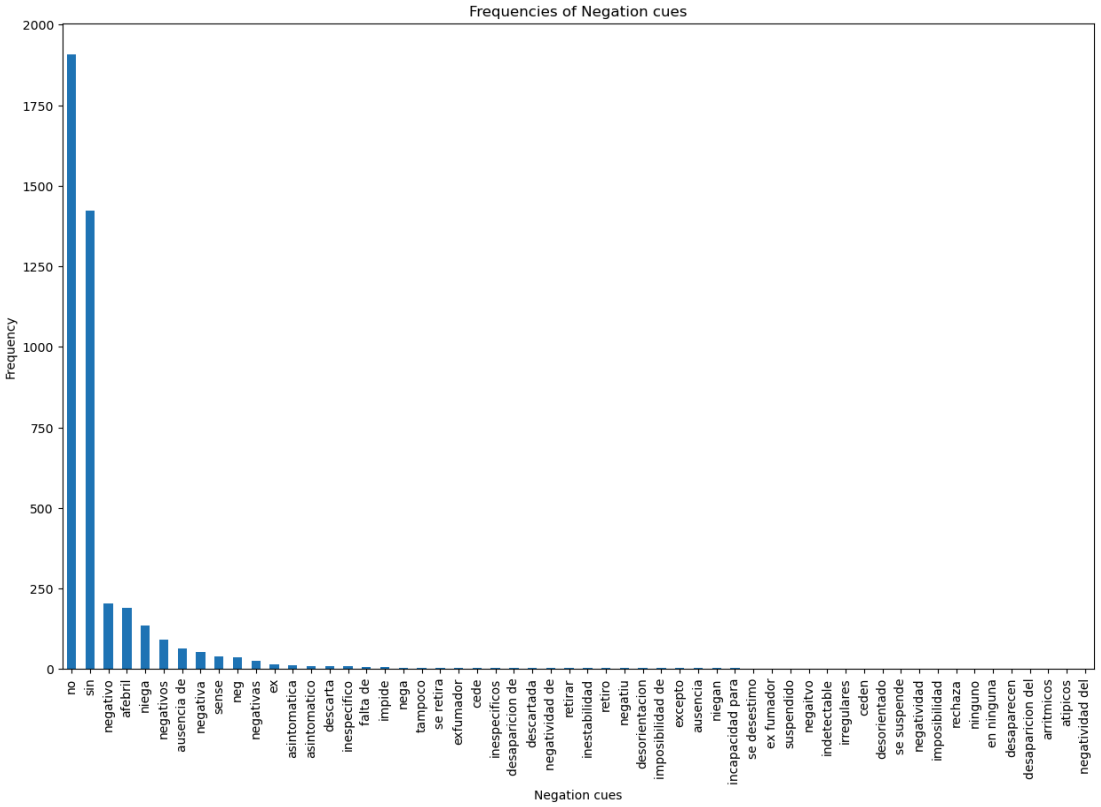
Report Natural Language Project

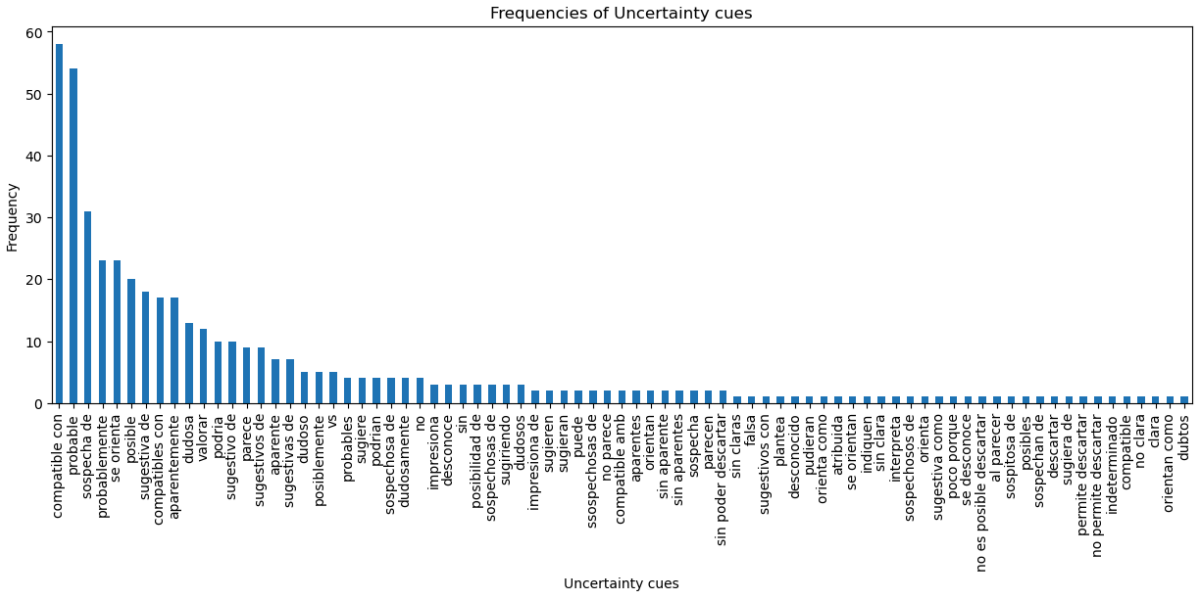
Alejandra Reinares, Álvaro Sáenz-Torre, Luis Domene, Joan Bayona

Abstract:
The objective of this project is to identify words expressing negation or uncertainty within a corpus of medical notes and determine their scope. The project explores three approaches: rule-based methods, machine learning, and deep learning. Rule-Based Method: A set of rules was developed based on linguistic patterns to identify the scope of negation and uncertainty cues.This method achieved moderate accuracy but was limited by its inability to generalise. Machine Learning Method: A Conditional Random Fields (CRF) model was trained using various linguistic features extracted from the medical texts. This approach significantly improved accuracy and provided better generalisation. Deep Learning Method: A bidirectional Long Short-Term Memory (LSTM) model was implemented, leveraging character and word embeddings, part-of-speech tags, and casing information. Despite the theoretical robustness, the model suffered from overfitting and data imbalance.

Data Exploration:
Before training any model with the train set that was provided, it is necessary to do data exploration in order to know more about the structure and distribution of the data.
The test set consists of a dictionary that contains several negation and uncertainty cues, and their respective scopes. Specifically, it contains 4307 negation cues and 458 uncertainty cues, although some items can appear more than once. If repeated items are not taken into account, 56 unique negation cues and 79 unique uncertainty cues can be counted. However, there are only 4103 negaction scopes and 451 negation scopes, due to the fact that certain words do not require a scope. An example of such a cue is the word “afebril”.
The following chart shows the amount of times each negation cue can be found in the train test. In it, it can be easily seen that some cues are far more common than others, such as the words “no”, “sin”, or “negativo”. Meanwhile, many others barely appear once or twice in the whole set.
Furthermore, when dividing the amount of times the word “no” appears by the total number of negation cues, the result is 0,44. If the same calculation also takes into account the amount of times “sin” appears, the result is 0,77. This shows that there is not much variation, since ¾ of the cues are actually the same two words.

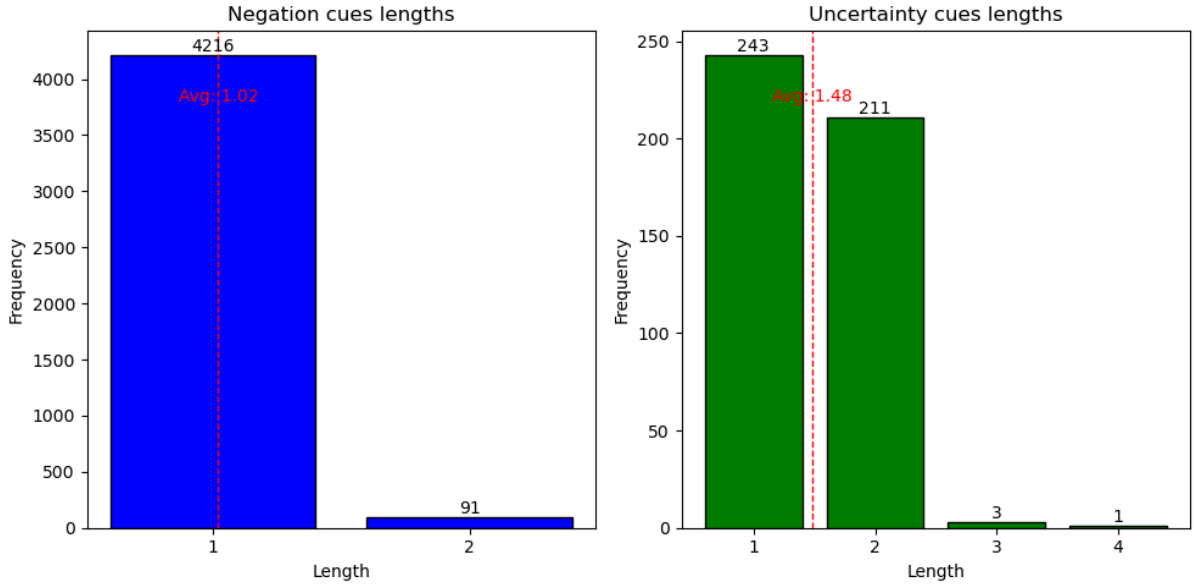


A similar thing happens with the uncertainty cues, in which chart we can see that some words appear much more often than others. The main examples of this are “compatible con”, which appears 58 times; and “probable”, which appears 54 times.



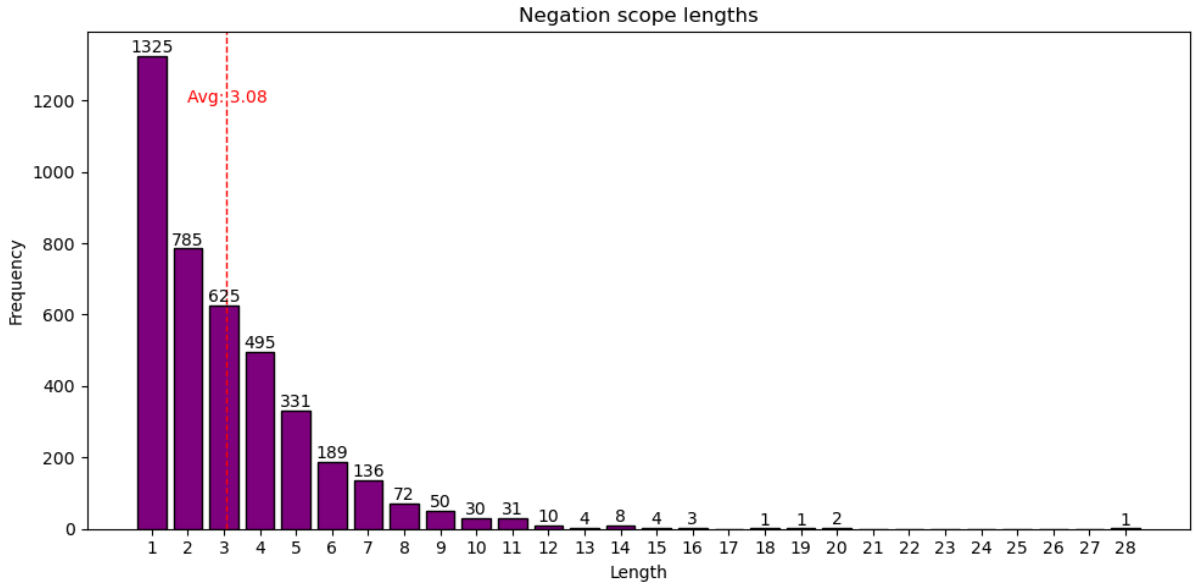
We also wanted to explore the length of cues and scopes, in order to see if there’s one predominant length. As we can see in the first plot, the 6 most popular negation cues appearing in the documents are: no, sin, negativo, afebril, niega and negativos. That’s an indicator already that most negation cues will be of length 1. In contrast, when we check the

frequencies of uncertainty cues, we notice that in the top 5 most popular cues, 3 are of length 2 and 2 are of length 1, so there's not a clear conclusion about the length of these.

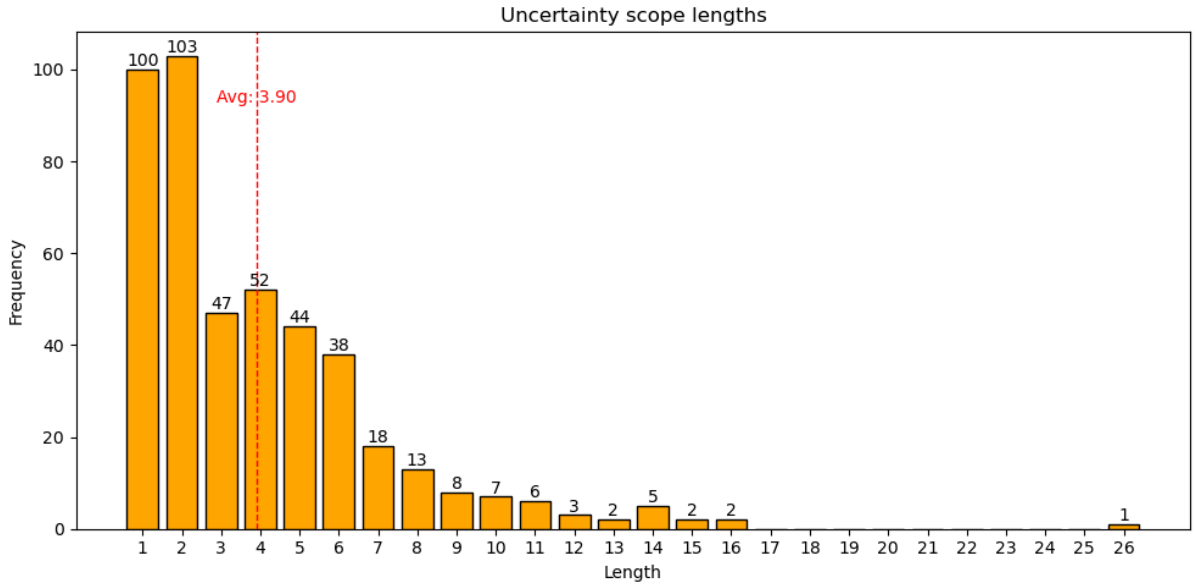


By checking the results, negation cues tend to be of length 1, but uncertainty cues are mixed between length 1 and 2, not having a really established length.

The next we checked were the lengths of the scopes. We knew that here there was gaping to be much more variance.



We observe that the average is around 3, but ⅓ of the negotiation scopes are of length 1. Investigating now the lengths of uncertainty scopes, we found this:



We observe that the average is close to 4. Almost ½ (45%) of all the uncertainty scopes are of length 1 and 2. After looking at all of these plots we can come up to the conclusion that the predominant lengths in everything is basically 1, and 2 is quite close behind. The most popular length in cues is 1, and also in negation scopes, even in uncertainty scopes it is just 3 points below length 2. After length 1 the most popular one is length 2, occupying 2nd position in cues and in negation scopes, only winning in uncertainty scopes. Looking at the averages we can also say that we can not discard totally lengths of 3 or more, we can in the case of negation and uncertainty cues, but in the case of scopes, they appear way less than length 1 and 2, but they still appear many times in total. ½ (48%) of negation scopes are of length >= 3 and in the case of uncertainty scopes also in ½ (54%) of cases, the length of the scope is bigger or equal than 3. Coincidentally, in both of the scopes graphs, ½ are of length 1 or 2 and the other ½ are of length >= 3.

Preprocessing:

The first step needed in order to work with the data consists of eliminating all of the patients' personal information, such as name, gender or weight. Therefore, everything contained in the documents before the actual medical report begins is deleted. A similar thing occurs with the information in the end of the document, which mainly contains data about the hospital and the patients' lives.

Also, some of the patients' personal information is already encrypted with the symbol * in the middle of the texts. These symbols are deleted, as they do not contain any relevant information for the task, but do make the algorithm less efficient.

Implementation:

Three different approaches can be used for this purpose:

1. Rule-Based
2. Machine Learning
3. Deep Learning

Now, we are going to explain these three approaches one by one:

- Rule-Based Method:

  The first approach that can be used in order to detect negation and uncertainty is a rule-based method. The idea behind this type of method is discovering regularities in the data that can be expressed in the form of an IF-THEN rule. It requires domain-specific knowledge (of the Spanish and Catalan languages, in this case) since the rules are created based on the domain.

  After doing some research on several papers from people who have tried to do a similar thing to what we intend to do, we have decided to try to replicate part of the solution shown in the paper “Integrating Speculation Detection and Deep Learning to Extract Lung Cancer Diagnosis from Clinical Notes”. The choice to replicate those instead of other solutions was made due to their simplicity, relevance and similarity to the task, and similarity to what an actual person would do to decide the scope of a word. While replicating these rules, adjustments were made to better suit our data.

  To begin with, two lists were compiled: one of them containing all words (or sets of words) that express negation, and another one containing those that express uncertainty. Both lists were extracted from the examples in the training data. A custom function was developed to scan text sentences one by one for these words using regex, enabling their position retrieval within sentences. The decision to extract these words directly from the data instead of using some other database was made so that those lists would also include commonly misspelled negation/uncertainty words.

  For this model, not much preprocessing of the data was needed, as these rules are designed to perform well even if some words are misspelled and most of them do not focus on the words themselves, but on their position in the sentences.

  The sentences on a text are separated and passed through the rules one by one. This is done because the scope of a word must be included in the same sentence as the word, and it simplifies the implementation of the rules. If none of the keywords are found in a sentence, no more work is done on that sentence, since it does not contain the scope of any keywords. However, those sentences that contain any of the words are processed in order to find their scope. For this, the rules suggested in the paper are used.

  Rule 1: Identification of Termination Terms
  The first one of these rules focuses on detecting words that signal the end of a scope, such as "but" or "however". These termination terms are crucial indicators that the scope of a negation or uncertainty cue ends before them in a sentence. An example of a sentence where this could be applied is “cistoscopia negativa para lesiones malignas pero se objetiva estenosis de uretra”.
  The rule involves creating a list of termination terms gathered from various sources, including internet searches and sample data. All of these words are stored in a list. While the initial idea was to use a list of possible termination words found on the internet, the results needed to be improved. That led to an active search over the

sample data in order to identify these words in it. The need for this extra list to store the termination words is the main problem related to this rule, since there is no fixed amount of words that can be considered termination terms.
When a sentence containing a negation or uncertainty cue is encountered, the rule checks if any termination terms appear after the cue. If found, the scope of the cue ends just before the termination term. This approach effectively captures the boundaries of scope within sentences. Due to the good results of this rule, not many changes with respect to the original needed to be made.

Rule 2: Handling Enumerations

The second rule focuses on enumerations of several negations or uncertainties. This rule addresses this by examining sequences of cue words occurring consecutively. If this case is observed, it is considered that the scope of one of them ends right before the next of these words starts. An example of where this is intended to be used is a sentence like “No pain, no inflammation, no joint pain, no fever”.
However, it's essential to differentiate between true enumerations and instances where multiple cues appear without an enumerative intention. An example of the second are sentences like “falsa vía a nivel de uretra peneana, siguiendo la uretra se detecta gran estenosis que no permite el paso de una guía”. This is not done in the original rule, which is the reason why many false positives occur. Therefore, this was one of the changes that was implemented after careful consideration and observation of the function´s output.
To achieve this, the rule checks for the presence of a comma shortly before the next cue word (specifically, in the 4 previous characters), indicating an enumeration. If such a comma is found, the scope of each cue extends until the next cue word, effectively determining their respective scopes. This refinement minimizes false positives and enhances the accuracy of scope determination.

Rule 3: Scope of short sentences

The third rule focuses on sentences that end shortly after the negation/uncertainty word.In sentences ending shortly after a negation or uncertainty cue, determining scope becomes straightforward. This rule identifies sentences where the cue is closely followed by sentence termination punctuation, a period. In such cases, the scope of the cue extends until the end of the sentence.The idea is for this to be applied on sentences like “posible cáncer pulmonar.”
To implement this, the rule examines a limited number of words following the cue, typically one to three words, checking for sentence termination punctuation. If found, the rule marks the end of the cue's scope. This rule simplifies scope determination in these sentences, ensuring accurate boundary identification.
The idea behind this rule was so simple that not many changes needed to be made.

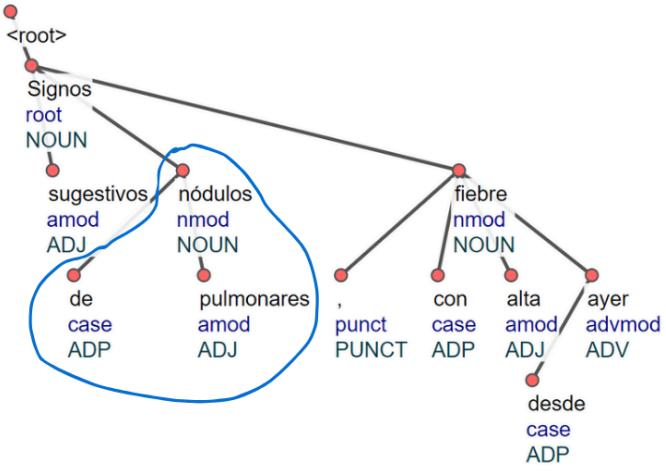
Rule 4: Part-of-Speech Analysis

The fourth rule is based on the PoS of a sentence. This rule involves tagging each sentence with PoS labels using trained taggers for both Spanish and Catalan languages (specifically, trained with corpuses from ntkl called cess_esp and cess_cat). The need to train one tagger with both corpuses came from the fact that texts in the train data include sentences written in both languages, even mixing them in the same sentence at times.

After tagging a sentence (one by one), the rule examines words following each negation or uncertainty cue. If a word after the cue is identified as a verb or conjunction, indicating a change in sentence structure, the scope of the cue extends until the beginning of the verb or conjunction.
However, some problems arise when implementing this rule in Spanish and Catalan, in sentences like “el paciente no ha mejorado”, where the first word after the negation is a verb. Therefore, exceptions are made for cases where the first word after the cue is a verb, considering it part of the cue's scope. Also, gerunds and participles are not taken into account at all. This rule leverages linguistic structure to refine scope identification, enhancing overall accuracy.

Rule 5: Parse Tree Analysis

The fifth rule is the last recourse when the first 4 rules haven’t found the scope of a negation/uncertainty cue. It is based on the parse tree analysis of a sentence. The original rule shown in the paper suggests that the scope of a sentence is given by the sub-tree that contains the uncertainty or negation cue. However, several problems arise when putting this rule into practice, which is what led to the creation of another version. One common problem is the fact that the cue has no sub-tree under it, since it is a leaf in the tree. Therefore, this rule cannot be applied correctly to this specific problem and needs modification.
This rule utilizes a parse tree generated using UDPipe, a tool for syntactic analysis. If the cue is part of the children of the root we grab as a scope the next sub-tree coming from the same parent as the cue. The cue's position within the parse tree is examined, along with its relationships with other elements. If the cue is a child of the root, the next subtree coming from that root is taken as the scope. In the following example: “Signos sugestivos de nódulos pulmonares, con fiebre alta desde ayer”, the cue is “sugestivos”, which is a child of “signos” (the root). Therefore, “de nódulos pulmonares” is considered the scope, since it is the next subtree coming from “signos”.
If the cue is not a child of the root, but a grandchild or even lower, the rule changes. In that case, it is considered that the scope consists of the neighbors of the cue and their children, and the head of the cue. e.g. If the cue was “con”, the scope would be “fiebre (the head) alta desde ayer (brothers and their children)”.



- Conclusions: After calculating some metrics, we finally saw how well our model performed on both the train and the test set. Our model got a 0.88

accuracy, recall and F-1 score on the train set and 0.86 on all values on the test set. While this might seem like a good result, we are not very satisfied with it since the simplified NegEx adaptation we created as a pipeline (which is explained below) got better results (specifically, 0.89 in the train set and 0.88 in the test set). We believe this is a sign that our model is not as good as we thought it was and probably needs some changes.

However, even after modifying some of the rules and actually making some progress, we were not able to come up with better rules in order to improve our model even more. For the most part, we believe we have been able to minimize false positives by changing some rules in order to fit our data better, after regularly checking the outputs of our functions and observing the most common mistakes. We believe that some of the mismatches might have been caused by the way some of our functions return the output (scope), since some of them might return an extra space or dot in comparison with the data given to us.

Although we did not get the results we expected, we are satisfied with the work we have done since we have worked hard on the implementation of all the rules and actually improved the original version (at least on our data). All we can say is that we trust the following models (machine learning and deep learning based) will give us better results, since this was the simplest model between the three of them.

- Contributions:
  Alejandra: Implementation of rules 2 and 5, writing the report, design of plots, testing
  Álvaro: Implementation of rule 1 and 4, training loop, design of plots, testing
  Luis: Implementation of rules 2 and 5, training loop, implementation of baseline
  Joan: Implementation of rules 3 and 5, revision of training loop code

- <u>Machine Learning Method:</u>
  The second approach consists of a machine learning method. The ideas applied in this model are an imitation of those shown in the paper “Negation Cues Detection Using CRF on Spanish Product Review Texts”, by Henry Loharja, Llus Padro, and Jordi Turmo.

  In this case, the approach relies on a supervised learning model using Conditional Random Fields (CRF). They are a type of statistical model widely used in natural language processing.
  CRFs are designed to work with sequential data, where each observation is part of a sequence. For example, a sentence can be represented as a sequence of words, and each word may be associated with a part-of-speech tag or a named entity label. They model the conditional probability distribution of a sequence of labels given the observed data. CRF´s rely on features extracted from the input data to make predictions. These features capture relevant information about the input sequence and its context.

  In this case, several features were used, all of them inspired by the paper and extracted word by word. The first one, PoS (Part of Speech) returns the information

about the Part of Speech of a given word. This was implemented by using the langdetect module of SpaCy, which works both in Spanish and Catalan. Even though the function only returns the PoS of the given word, it classifies the whole sentence in order for it to take context into account and be more accurate.
Then, Init_Cap returns a boolean that reveals whether a word starts with a capital letter or not. Alphanum also returns a boolean that corresponds to whether a word contains only alphanumeric characters or not. Has_Num checks if a token contains numeric characters, and Has_Cap looks for any capitalized letter in the word. Also, Has_Dash checks if a word contains a dash and Has_Us checks if it has an underscore. Lastly, Punctuation looks for punctuation inside the word.
For the next two features, two lists are needed. SufN looks for common sufixes in the ending of the word, which is the reason why a list of those suffixes in both Spanish and Catalan is needed. The same thing happens with PrefN, which looks for prefixes in the beginning of words.
The rest of the features focus not only on the current word, but in its context. 2GrammarBefore returns a list of lists containing the bigrams of the 6 words previous to the current, if they exist. The 2GramAfter function does the analogue thing with up to one word after the current.
BeforePos and AfterPos return the PoS of the 6 words before and 1 word after the current, respectively.
Last of all, the Special function focuses on a special type of negation that is very frequent in the Spanish language: double negations. It uses a list of special words such as “nada”, “ni” or “nunca”, which have a tendency to be part of negation cues of multiple words.
All of these features are extracted one by one for each of the words in the text. When all of them have been correctly extracted and stored in the corresponding database, they can be used to train the CRF model.

Although the training and test data that is used indicates the scope of the negations and uncertainties based on characters, CRF models cannot work with this. Therefore, a function called tagger is needed in order to change this representation and work with words.

For this project, the chosen CRF implementation is the one provided by the sklearn library. The function receives both the features of the training data, extracted with the help of the previously described functions, and the correct tags of this training data, in order to train the model. After doing so, it can predict the tags of the words contained in new and unseen sentences.

Even though the theoretical part of this project can appear to be simple, several problems can arise during the implementation. Some of them are the following:

- The most time consuming part of the project is to process the json file in order for it to have an adequate format for the task. The original data is transformed into a format that represents the different sentences, so that the model can have a notion of sentence. For this reason, the features and tags are divided in lists that represent each sentence. The following image shows an example of how the feature vector of each word would look like and the tagged sentence.

```
[['NOUN', False, True, False, False, False, False, False, '', 'in', [],
['informe', "d'"], [], False], [None, False, False, False, False, False,
False, True, '', '', [], None, ['VERB'], False], [None, False, False, False,
False, False, False, True, '', '', [], ["d'hospitalitzacio", 'motiu'], [],
False], ...]
```

```
['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
'O', 'O', 'O']
```

- The first approach taken in this project was simply using BIO tagging, which failed to give good results because the step of computing the BIO tags from the json was not being performed correctly. The fact that the accuracy of the model was below 80%, and mainly that the accuracy of the tags without including the tag “O” was only 20%, gave a clue that something was not right. Moreover, it could be seen more clearly when carefully checking the tags, that many of them were not in the correct place.
- An alternative to solve the BIO model problems was to directly perform the classification using the labels given in the training set. Another problem arose when trying to implement this: the sklearn module does not accept anything but strings as features. This creates the need for another function that changes each feature vector to a vector of strings.

In the end, and in order to properly start evaluating the model, a portion of the training set (1500/11782 sentences) was used as a validation set, to perform adjustments in the hyperparameters of the model. The final values for each parameter were decided after manually traversing through every possible combination of those parameters, keeping the ones with best accuracy. In the end, the model was tested in the test set, in order to check the performance on unseen data.

- Conclusions: The results of this model are significantly better than the Rule-Based approach. After some hyperparameter adjustment, the final model reaches an accuracy of 95%, a precision of 94,3%, a recall of 95% and a F1 score of 94,4%. It is important to highlight that the accuracy, when excluding the "O" tag which skews the result, is 69%.
  We are satisfied with this model because the results are solid and it improves the results of some of the papers we have read. We can see that there is no overfitting, and the overall results are good. The model works well with negations and negation scopes. However, there is much room for improvement when talking about uncertainties and uncertainty scopes, due to the lack of enough data for the model to correctly learn the patterns..

- Contributions:
  Alejandra: Extraction of features, writing the report
  Álvaro: Revising and putting all the features together, research of CRF, changing tagger from character based to word based, implementation of CRF
  Luis: Extraction of features, research of CRF, implementation of CRF, testing
  Joan: Extraction of features

- <u>Deep Learning Method:</u>
  The last approach taken is a deep learning method. The ideas applied are based on the paper “Deep Learning approach for Negation Cues Detection in Spanish”, by Hermenegildo Fabregat, Juan Martinez-Romo and Lourdes Araujo.

  The main idea is to use a bidirectional LSTM, which is trained with the following features: a representation of the characters, a representation of the words, the Part of Speech tag of each word and the case tagging of each word. A Long Short-Term Memory (LSTM) model is a type of recurrent neural network that aims at dealing with the vanishing gradient problem present in traditional RNNs.

  Part of the preprocessing for this specific LSTM model was to apply padding to the original sentences in order to get sentences that have the same length (377). Otherwise, the LSTM cannot work with them.

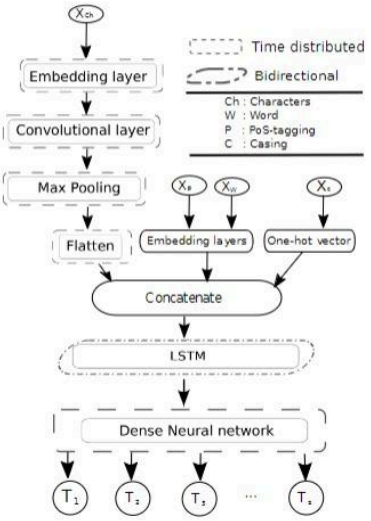
  The implementation of the model itself begins with the extraction of features. The first feature is the embeddings for each character, for which a pre-trained FastText model is used (https://fasttext.cc/). The FastText model returns a vector of length 300 as the representation for each character. In order for all characters to have the same vector

length, the vector is modified with PCA (Principal Component Analysis), which reduces its dimension to 100. After this, a convolution is performed on this feature, followed by a max-pooling layer. The objective of this is to obtain relevant features about the character distribution, given by FastText.
Similarly, the embedding representation of the words is extracted from a pre-trained Word2Vec model (Cardellino, 2016). This embedding representation also consists of a vector of length 300, which is then passed through a PCA that transforms it into a vector of length 100.
The PoS of a word is represented as a one-hot encoding that is later embedded as a vector of length 100.
Meanwhile, the casing feature represents several characteristics of a word. Tt is a one-hot vector of length 8, in which each position represents one of the following: if the first character is an uppercase letter, if the word is alphanumeric, if it contains numbers, if it contains uppercase letters, if it contains -, if it contains _, if it consists of a punctuation symbol, and if it is a previously stored special word (negation, for example). These vectors are also embedded into vectors of length 100.
The reason why all the features are transformed into vectors of the same length is so that they all have the same importance and are compensated, which facilitates the posterior training of the model.

The padding mentioned before does not have any of the previous features, since it is not a real word. However, they need some representation in order to be processed by the model. Therefore, the vectors that represent its features are filled with only 0. Also, its BIO tag is called padding. Even though the tag exists, it is not taken into account when computing the loss. This is the way to ensure that the model does not learn to classify everything as padding, since it is the most common tag.

Before this model, there was an attempt to create a GRU based model, which was an almost exact replica of the one presented on the first paper mentioned above. The general idea was the same as the LSTM model that was implemented in the end. Both models started with similar (but not equal) features, but GRU failed because of the computers' memory limitations.

As for the training part, the model was implemented by replicating the structure of the image. This idea is proposed in the paper "Extending a Deep Learning Approach for Negation Cues Detection in Spanish", by Hermenegildo Fabregat, Andres Duque, Juan Martinez-Romo and Lourdes Araujo.
As shown in the image, all the features are concatenated in one vector that is passed to the model. Then, a bidirectional LSTM receives the data and processes it. It uses a hidden state for processing data from the current step taking into account information of previous steps (in other words, the context). Last of all, a dense hidden layer is used to reduce the complexity of the bidirectional LSTM output.

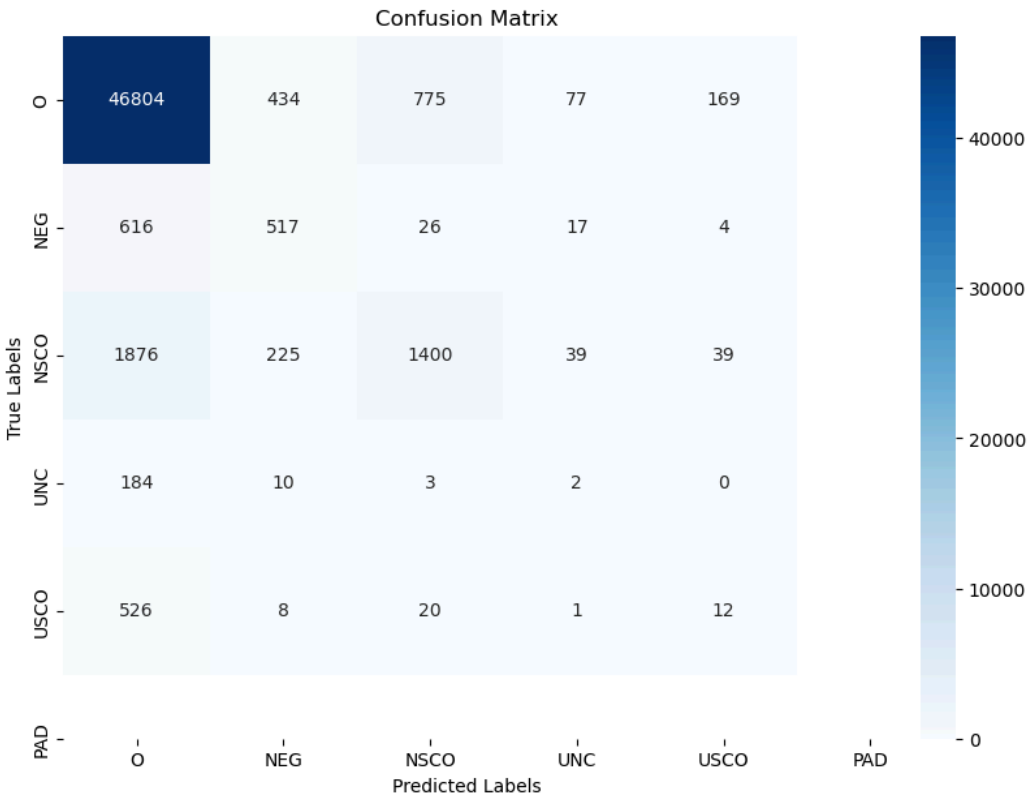


The objective is to get the LSTM to give each word a correct tag between the following, which are similar to a

BIO tagging approach: "NEG" (negation), "NSCO" (negation scope), "UNC" (uncertainty), "USCO" (uncertainty scope), "O" (out), "PAD" (padding).

Due to the fact that the dataset is very unbalanced, it is complicated for the model to learn how to classify the most uncommon examples. In this case, the vast majority of tags are “O”. Also, the words tagged as “UNC” and “USCO” are very few, which causes the model to not recognize many of them when working with new data.
In order to deal with this, we used a new approach when calculating the loss. In this case, some weights are assigned to the different tags in the loss calculation. These weights are inversely proportional to the frequency with which these tags appear, which causes an improvement in the classification of the most least common words. Indeed, the model went from classifying everything as “O” to a most accurate result.

- Conclusion:
  The model has a total accuracy of in the train set. If we check the classes individually, the accuracies are 93.76% for “O”, 95.02% for “NEG”, 94.90% for “NSCO”, 98.54% for “UNC”, 98.02% for “USCO” and 0.00% for “PAD”, since it was not taken into account. Although this may seem like a very good result, the accuracies in the test set are nowhere near these values.
  The overall Accuracy of the model on the test dataset is 90.61 %, with a 96.99 % accuracy for “O”, 43.81 % for “NEG”, 39.12 % for “NSCO”, 1.01 % for “UNC” and 2.12 % “USCO”.
  Our belief is that our model does not learn the right weights due to overfitting, but the different attempts to solve it have been unsuccessful. Regularization improved the results, but not nearly enough. Meanwhile, the attempt to use weights associated to the frequency of each tag when computing the loss was also an improvement, but did not solve the problem of the unbalanced dataset. Most tags present are “O”, while the rest are a minority in comparison. This is the reason why the overall accuracy is high although most of the accuracies are really low.
  Another significant problem is the low count of uncertainty cues. Deep learning models are very effective, but also need an enormous amount of data in order to learn the correct patterns. In this case, the data of this type is not enough.



- Contributions:
  Alejandra: Writing the report, implementation of previous model (GRU) that was not used in the end, creation of plots
  Álvaro: Embedding of words, padding, implementation of model, implementation of previous model (GRU) that was not used in the end, creation of plots
  Luis: Extraction features Fast-Text, PoS and casing, dimensionality reduction of embeddings, implementation of model, testing
  Joan: Fixing and adding plots, design of presentation

Simplified NegEx Adaptation:
In order to have another model to compare with, this simple approach that can be used as a baseline was created. The model searches for both negation or uncertainty cues in each sentence. For each word found, the closest medical term is searched between its neighbors. The scope of the tags is given by the words between the cue and the medical term. It is important to highlight that the order does not matter: ***Medical Term + Words[0-5] + Cue*** or ***Cue + Words[0-5] + Medical Term.***
The main drawback of this approach is the need for a list of medical terms that can be used as vocabulary.
The database used for this model comes from a GitHub repository. The medical terms in it have been translated to Spanish in order to work on this problem. It contains 32.431 medical terms.
https://github.com/PlanTL-GOB-ES/MeSpEn_Glossaries

In addition to the .txt file mentioned above, the lists of words in the labeled data used in the uncertainty and negative scopes also added new words to the vocabulary. The words added were those labeled as either a noun or an adjective.
The model works by searching for cues in the sentence and checking the 5 previous and 5 posterior words, in order to look for any medical term between them. If one is found, the scope is considered to be formed by the words between the cue and the medical term.
This causes another one of the drawbacks of the approach: It is very slow because it has to search in a huge amount of medical terms. This complication led to the decision of not using the complete least for the time being, and using only the list made with the labeled data. Even after this improvement, it takes considerably longer than the rule based implementation.

References:

Chapman, W. W., Bridewell, W., Hanbury, P., Cooper, G. F., & Buchanan, B. G. (2001). A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of Biomedical Informatics*.

Solarte Pabón, O., Torrente, M., Provencio, M., Rodríguez-Gonzalez, A., & Menasalvas, E. Integrating speculation detection and deep learning to extract lung cancer diagnosis from clinical notes.

Costumero, R., Lopez, F., Gonzalo-Martín, C., Millan, M., & Menasalvas, E. (Año). An approach to detect negation on medical documents in Spanish.

Beltrán, J., & González, M. (2014). Detection of Negation Cues in Spanish: The CLiC-Neg System. Proceedings of the Workshop on Language Analysis and Processing for Social Media, 19-24.

Morante, R., & Daelemans, W. (2010). A metalearning approach to processing the scope of negation. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (pp. 374-382). Association for Computational Linguistics.

Loharja, H., Padró, L., & Turmo, J. (2018). Negation cues detection using CRF on Spanish product review texts. In Proceedings of the Workshop on Negation in Spanish (pp. 49-54).

Enger, M., Velldal, E., & Øvrelid, L. (2014). An open-source tool for negation detection: A maximum-margin approach. Proceedings of the 14th Nordic Conference on Computational Linguistics (NoDaLiDa 2014), 217-224.

Fancellu, F., Lopez, A., & Webber, B. Neural networks for negation scope detection. School of Informatics, University of Edinburgh.

Fabregat, H., Martinez-Romo, J., & Araujo, L. Deep learning approach for negation cues detection in Spanish: Aplicación basada en deep learning para identificación de claves de negación en castellano.

Fabregat, H., Duque, A., Martinez-Romo, J., & Araujo, L. Extending a deep learning approach for negation cues detection in Spanish.