Detection of negation and uncertainty in medical data

Amritpal Singh, Óscar Arrocha, Mustapha El Aichouni, Eric López

May 14, 2024

Abstract

In medical data analysis, accurately interpreting text, like patient information, is crucial. Negation and uncertainty are particularly challenging features, and misinterpreting them can lead to errors in medical decision-making. For this reason, this project aims to achieve a high precision detecting negation and uncertainty cues and their context, by developing and comparing three methods: a rule-based algorithm, a machine learning and a deep learning-based algorithm.

Rule-based method

The goal is to build a rule-based algorithm to detect uncertainty and negation cues, and their corresponding context, in our dataset, which consists of 318 medical documents with patient information and a variety of medical terms. First we implement a baseline algorithm, which tags the entire sentence after or before the negation word as its context. Then we present an implementation of NegEx **negex**, which uses a simple heuristic rule to determine a more accurate context.

Baseline algorithm (Blind)

Our first approach is to create a baseline model with no information about the words. That is, after finding a negation/uncertainty word, the model will tag as scope what is behind or before the negation (depending on its tag) until reaching at the end/start of the sentence. We call this method "blind" because it does not take into account (it doesn't "see") words, only the delimiters of the sentence. Despite this approach appearing too simplistic, it's expected to work mostly well for most of the cases, as many sentences have been observed to consist only of the negation and the scope, like "no hábitos tóxicos.".

Implementation

First, a negation list is created, where all the negations of the training set are grouped and given a tag, based on where the position of its scope is usually found on the text (behind or after):

- The tag [PREN] means that the negation is before the scope and the scope has to be tagged as [NSCO]
- The tag [PREP] means that the negation is before the scope and the scope has to be tagged as [USCO]
- The tag [POST] means that the negation is after the scope and the scope has to be tagged as [NSCO]
- The tag [POSP] means that the negation is before the scope and the scope has to be tagged as [USCO]

Then, this information is used to create the algorithm. First, for a given sentence, all the negations that are spotted. Then, for each negation everything that is before/after the negation is tagged as the corresponding scope. A weak point of this implementation is that for sentences with more than one negation, the scopes will be overlapped.

Results

Evaluating this algorithm in the test set yields a precision of 0.82. Recall and F1 score result in the exact same value. As expected, the model performance at detecting the scope is reasonably high for its such simple approach. This is because the evaluation algorithm is considering the whole text when comparing the ground truth with the prediction, and detected negation cues and scopes are only a small part of it. This must be taken into account for future methods explained in this report, we are aiming to achieve a high precision relative to this one.

NegEx

Once implemented a baseline model, we proceed to implement the NegEx algorithm as explained in the original paper **negex**. NegEx proposes a simple strategy to detect negation words and their scopes in medical documents:

- First, the negation and uncertainty words are spotted in the sentence.
- Second, the medical terms are identified in the sentence. A predefined vocabulary [1] of these terms is necessary for this step.
- Finally, the negation scope is defined as the next words after the negation word until the medical term, included. The number of tokens that can appear in between is limited to 5. Moreover, depending on the negation or uncertainty word, the scope will be before or after.

Implementation

Before implementing the NegEx algorithm, we found it beneficial to first preprocess the data. Each document is split into sentences, and then tokenized into words. For this last step, the tokenizer works better if the language is provided, so a first language detection step is performed over each sentence, distinguishing Catalan from Spanish.

Next, the process consists of, for each sentence, identifying the negation or uncertainty words, then the medical terms. These steps are done by simply matching each word in our vocabulary of the corresponding terms. Finally the scope is detected, following the definition previously explained, using an efficient and straightforward approach: a binary mask of the sentence is created, where medical terms are 1, the rest 0. Also, for each negation and uncertainty word a mask is created where words with a distance equal or below 5 to that word are 1, the rest 0. For each word, both masks are matched using an overlapping criteria, where only sets of 1 that intersect or are externally connected remain in the resulting mask, which represents the scope for that word. This ensures that the scope includes at least a medical term within a distance of 5 words maximum. The scope is saved as spans in format [start index, end index], corresponding to the indices of the characters in the original text. In this way, sentences for each document are processed.

Results

By implementing NegEx as proposed in the paper, we raise all our metrics (precision, recall and F1 score) from 0.821 to 0.897 in the best case of the hyperparameter exploration, shown in Figure1. The best value for maximum context size is 1, in contrast to 5, used in the original paper. We believe that in most sentences, the scope is rather close to the negation or uncertainty word, and a smaller maximum context size reduces the chances of having overlapping scopes.

The efficiency of this algorithm is reflected in the time taken to process the documents: an average of 3.1 seconds to process the test set, 0.05 each document. The simplicity and lower precision of this method, compared to the next ones, is compensated with the high speed and efficiency of its implementation.

Note that in these type of natural language problems, the closer we get to a perfect accuracy, the harder it becomes to increase the performance of the model.

[1] The original paper used a union of terms from the UMLS and from ICD10. As we don't have access to such private databases, our workaround consists of creating a vocabulary of medical terms by extracting all words from the scopes in the training set, listing them and removing those words with 3 characters or less. This is an heuristic to remove common determinants and conjunctions that may appear in the scopes, leaving only the medical terms, which are usually longer.

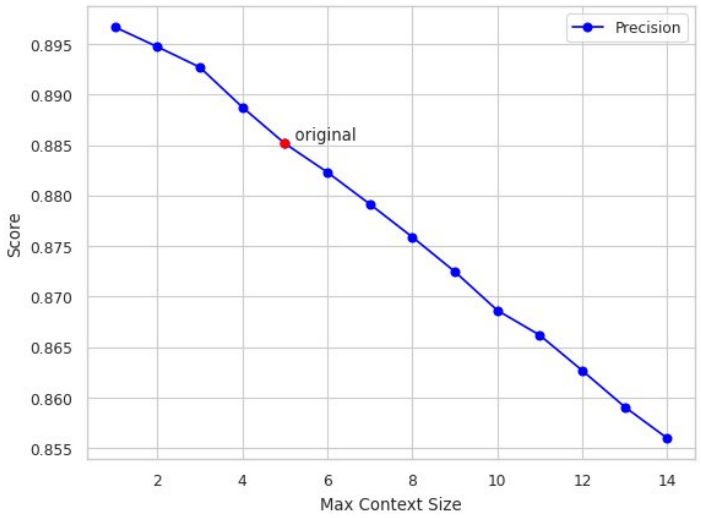

Figure 1: Performance score of NegEx for different values of the maximum number of words between the negation or uncertainty word and the medical term. In red, the value (5) used in the original paper. Note that only the precision is shown, because the recall and F1 score resulted in the exact same value.

Machine learning method

Machine learning (ML) significantly enhances healthcare technology by interpreting complex data sets and training algorithms to recognize patterns with minimal human intervention. This project applies ML to improve the processing of medical data, specifically for detecting negation and uncertainty in patient records.

This section details the use of ML models, algorithm selection, data preprocessing, and performance metrics. By automating the detection process and reducing misinterpretation risks, this approach will support doctors in delivering precise medical care and enhance the quality and reliability of medical documentation, ultimately improving patient outcomes.

Conditional Random Field

In this study, a Conditional Random Field (CRF)-based approach is employed to enhance the detection of linguistic nuances such as negation and uncertainty within text data. The problem is treated as a Named Entity Recognition (NER) challenge, focusing on the identification and classification of words related to negation and uncertainty. Specifically, a tagging scheme is designed based on the BIO format to classify words under negation and uncertainty categories:

- B-NEG: Indicates beginning word of negation cue.
- I-NEG: Indicates inside word of negation cue.
- B-NSCO: Indicates beginning word of negation scope.
- I-NSCO: Indicates inside word of negation scope.
- B-UNC: Indicates beginning word of uncertainty cue.
- I-UNC: Indicates inside word of uncertainty cue.
- B-USCO: Indicates beginning word of uncertainty scope.
- I-USCO: Indicates inside word of uncertainty scope.
- O: Tags all other words that do not belong to any specific category of negation or uncertainty.

Using the above tagging, we have trained two different CRF models that differ from the features used. The models are trained to predict these tags by learning from context-dependent features within the data, including lexical, syntactic, and positional indicators.

	Train			Test		
Class	Precision	Recall	F1	Precision	Recall	F1
B-NEG	0.96	0.94	0.95	0.96	0.93	0.95
I-NEG	0.99	0.78	0.87	1.00	0.86	0.92
B-NSCO	0.96	0.91	0.94	0.97	0.90	0.93
I-NSCO	0.91	0.90	0.90	0.88	0.88	0.88
B-UNC	0.95	0.78	0.86	0.95	0.67	0.70
I-UNC	0.96	0.82	0.88	0.94	0.69	0.80
B-USCO	0.96	0.78	0.86	0.87	0.69	0.81
I-USCO	0.89	0.72	0.80	0.82	0.74	0.78

Figure 2: Train and test set results, detailing the metrics computed for each of the classes.

Features - First Approach

This approach focuses on providing basic features about each word to the model. Here is the list:

1. TOKEN: the word itself
2. LEMMA: the base of the word
3. POS: the information of part of speech of the word.
4. CHUNK: the syntactical group of the word
5. BIAS: bias term added to the feature vector

Additionally, the features "token", "pos" and "chunk" of the previous word and the next are also included in the feature vector of the current word, as well as beginning and end of sequence tokens.

Results - First Approach

Overall, this model performs well on both the training and test sets (with better performance on the training set, as expected), with high precision scores indicating accurate predictions for most classes (above 0.89 in the training set and above 0.82 in test set). However, there are challenges in capturing all instances of certain classes, as reflected in lower recall scores. The F1-scores generally demonstrate a good balance between precision and recall, but some classes show imbalances, particularly in the test set.

Features - Second approach

In this approach we have used a total of 16 features as described in the original paper **crf**. Here are the complete set of features used in the training:

1. WORD: the words itself.
2. POS: the information of part of speech of the word.
3. INIT CAP: word starts with capitalization.
4. ALPHANUM: word consists of alphanumeric characters.
5. HAS NUM: word contains number.
6. HAS CAP: word contains capitalized letter.
7. HAS DASH: word contains dash.
8. HAS US: word contains underscore.
9. PUNCTUATION: word contains punctuation.
10. SUFn: suffixes in the n character length ranged from two to four.
11. PREFn: prefixes in the n character length ranged from two to four.
12. 2GRAMBEFORE: bigram of up to 6 words before the observed word.

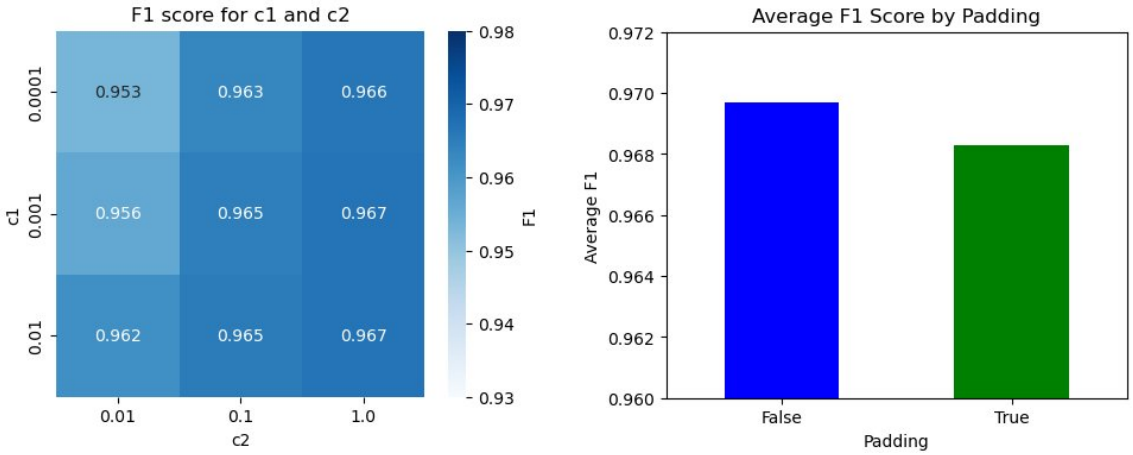

Figure 3: To the left, heatmap showing the F1 score for different combinations of values for parameters c1 and c2. To the right, F1 score for using padding or not in context windows, an average is computed across different window sizes for a more stable score.

13. 2GRAMAFTER: bigram of up to 1 word after the observed word.
14. BEFOREPOS: the information of part of speech of up to 6 words before the observed word.
15. AFTERPOS: the information of part of speech of up to 1 word after the observed word.
16. SPECIAL: word is one of the special words in the special dictionary. The words we included as special words are: "nada", "ni", "nunca", "ningun", "ninguno", "ninguna", "alguna", "apenas", "para nada", and "ni siquiera". These words have more tendency to be part of a negation cue with multiple words, in which other words may appear in between.
17. BIAS: bias term added to the feature vector.

With this second approach, our CRF reaches higher performance, so we decide to move forward with these features, find optimal hyperparameters and evaluate the model.

Hyperparameter tuning - Second Approach

Once designed the CRF and the training method, it is still possible to improve the model performance by tuning the hyperparameters. To do so, our approach was to make a grid search to analyze the best parameter combination. The parameters our model can take are c1 and c2 (L1 and L2 regularization, respectively), maximum number of iterations, the context window length before and after a given token and whether to use padding. Note that a CRF can observe the information from previous and posterior tokens. The original paper proposes those context windows to be of a maximum of 6 tokens before and 1 after.

Our grid search analysis is not exhaustive, but heuristic. The search is done in groups of parameters, so first the optimal value is found for those parameters, and then another search is done for another group, fixing the previous parameters to their optimal values. For this evaluation, the F1 score is used as the performance metric, as it leverages precision and recall. Moreover, these other metrics have been observed before to be of identical value.

First, the search is performed for parameters c1 and c2. As shown in Figure 3, the optimal values are a low c1 (0.001) and a high c2 (1.0). These values are then fixed. Next, the optimal choice between using padding or not is also found doing a grid search, this time not only with padding but also for a small range of context window lengths before and after a token. This allows us to make a more confident choice. The average of the F1 score is computed across these lengths, and not using padding results as the best choice. Lastly, we confirm, as shown in Figure 4 that the values for the context window length limit mentioned in the original paper, 6 before and 1 after, are the optimal [2].

[2]Despite the grid search showing a length limit 2 to perform slightly better for context before, we decide to stay with the value of 6 for more stable and informed predictions in future inferences.

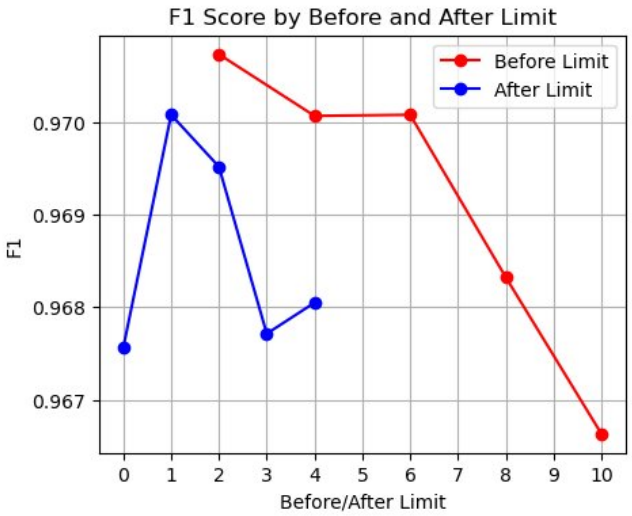

Figure 4: F1 score by context window length limit before and after a given token. Despite both plotted together, their value search has been computed separately.

Results - Second approach

Now the hyperparameters are the optimal. A 5-fold cross validation is performed to ensure the training performance, which results in a F1 score of 0.973. Finally, the model is trained on the entire training set and evaluated on the test set, achieving a F1 score of 0.971. Comparing it with the training performance, we confirm the model had no overfitting.

Results comparison: CRF, both approaches

When comparing the two methods in terms of F1 score, the second approach achieved a higher performance, with an F1 score of 0.971, while the first one only managed an average F1 score of 0.846. Therefore, we can conclude that the features used in the second approach are more suitable for the task of negation detection than the ones used in the first one.

Results comparison: Negex and CRF second approach

At inference time, the CRF model took 1.24 seconds to process the test set, half the time that NegEx took. However, we must consider that the CRF required a first training step which lasted over a minute, and additional data preprocessing (BIO labels precomputation for training set and POS tags for both sets).

Despite CRF needing extra treatment, we show in Figure 7 how the performance on the task of identifying negation cues and their context is significantly increased, almost to perfection.

Conclusions

In this study, we have implemented a Conditional Random Field (CRF)-based methodology to address the challenge of detecting negation and uncertainty in medical texts. Our approach treats the problem as a Named Entity Recognition (NER) task, where each word in the text is assigned a tag that indicates its role in expressing negation or uncertainty. The CRF model predicts one of these tags for each word in a sequence, leveraging comprehensive lexical, syntactic, and positional features for effective training.

Comparatively, the CRF model shows superior performance to the NegEx algorithm, demonstrating higher precision and recall in detecting nuanced linguistic elements. While the CRF model is faster in processing data once trained, it does require an initial training period which is not needed for the rule-based NegEx. Additionally, the CRF approach necessitated more extensive data preprocessing, including the identification and incorporation of specific lexical and syntactic features to adequately train the model. This initial investment in model training and data preparation allows the CRF model

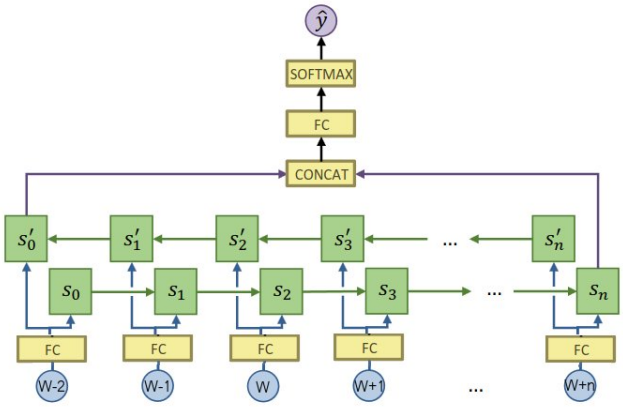

Figure 5: Model architecture of the deep learning approach. It consists of a dense linear layer at each time-step, followed by a bidirectional LSTM layer and another dense layer. It is a many-to-one model, taking a sequence of words as inputs but outputting only the tag corresponding to the central word.

to more accurately reflect the complexity of medical language, thereby providing more reliable and precise analysis than the simpler NegEx method.

This comprehensive approach not only enhances the quality of medical documentation analysis but also supports healthcare professionals by providing clearer insights into patient records, thus aiding in more accurate clinical decision-making.

Deep learning method

Deep learning (DL) has emerged as a powerful tool for natural language processing tasks. Its application in detecting negation and uncertainty within medical texts has shown promising results. By training models on large datasets annotated for these linguistic features, deep learning algorithms can learn to recognize complex patterns and contextual nuances that traditional rule-based systems often miss. This section details the use of DL models, algorithm selection, data preprocessing, and performance metrics.

Bidirectional Long-Short Term Memory (BiLSTM)

The model chosen for the deep learning approach is a bidirectional LSTM, which scientific literature has demonstrated the most prominent model for the task of negation detection. As in the ML section, the problem is treated as a Named Entity Recognition (NER) challenge, focusing on the identification and classification of words related to negation and uncertainty. Specifically, a tagging scheme is designed based on the BIO format to classify words under negation and uncertainty categories. This scheme has been explained in the ML section, and therefore, all the information can be found there.

Model architecture

The model, as shown in Figure 5, is composed of a dense linear layer, followed by a bidirectional LSTM layer and another dense layer. The LSTM is used to learn complex patterns and dependencies within the sequential medical text data. By processing the text in both forward and backward directions, the bidirectional LSTM captures context from both preceding and succeeding words, enhancing its ability to detect subtle cues and scopes of negation and uncertainty. The first dense layer is used to reduce the dimensionality of the input, while the latter is used to transform the high-dimensional output of the LSTM into a final set of prediction scores, which indicate the likelihood of each word being part of a negation or uncertainty category (a BIO class). Thus, it is a many-to-one model: it takes a sequence of words as inputs but the output corresponds only to the central word.

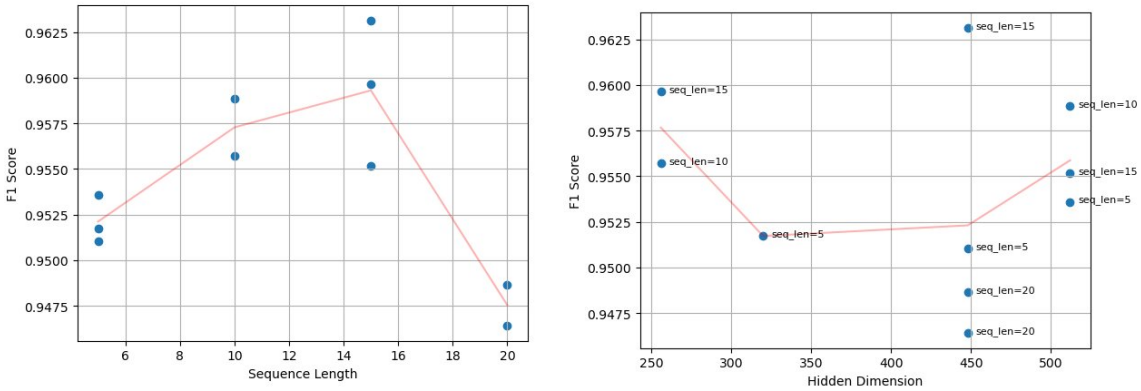

Figure 6: F1 scores obtained for different evaluations of the model, trying different sequence length values (left) and different hidden dimension values (right), using random search. Multiple F1 values for the same sequence lenght or hidden dimension indicate that those evaluations differ in other hyperparameter values, and thus the average (red line) is taken.

Model training

The input of the model consists of sequences of k words. Specifically, for each word a sequence is created with the 0.8*k previous words and the 0.2*k posterior words. The decision to include less posterior information than anterior is based on the success we had in doing so with the CRF, as well as in previous literature **lstm1 lstm2**. So, the sequence from A to B around a word W (and W itself) is passed to the model at one iteration. At the next iteration, the input will be a sequence from A+1 to B+1 and the word W+1. This means that the input sequences across iterations are overlapping.

For the embedding of the words, the following steps have been taken: first, the word lemma and the PoS tagging are computed for each word. Secondly, pretrained embeddings are extracted for the word itself and its lemma, obtaining two different vectors of 300 dimensions each. This is done with the help of the "fasttext" model **joulin2016bag joulin2016fasttext**. Next, a one hot vector is created consisting of the 17 grammatical categories a word can fall into (the PoS). Finally, the three vectors are concatenated into a vector of 617 dimensions.

This high dimensional vector of a given word is passed to the model at one time-step. It is first processed by the feed-forward layer and then by the LSTM block. This is done with all the sequence of words, each time-step producing two outputs (from the forward and backwards passes). Then, the last output from the forward pass and the last from the backward are concatenated into a single vector, which is finally passed to the second feed-forward layer, which produces some logits corresponding to each BIO class. During training, these are passed to a softmax layer and Cross Entropy is used as a loss function. At inference time, the argmax is computed on the logits to extract the most likely class.

The model was trained using the Adam optimizer, with an initial learning rate of 0.001, and a batch size of 32. The training was done for a single epoch, as the model was observed to converge quickly. The hardware used was a NVIDIA GTX 1050 Ti GPU, with 4GB of VRAM.

Hyperparameter tuning

Similarly to the CRF, with the deep learning model we have the opportunity to change some hyperparameters and attempt to improve the model performance on the task. In general, in deep learning it is not possible to know the best parameters in advance, and there is a long continuous range of values for some parameters to try (e.g. we explored values for hidden dimension from 64 to 512). For this reason, our decision was to make a random search this time, and explore 10 possible combinations of values for the hidden dimension of the LSTM layer, the number of layers of the same, and the sequence length. After a thorough analysis, we found a sequence length of 15 to be the best value, as well as a hidden dimension of 448 and a number of layers of 2. Figure 6 left supports this decision. Figure 6 right, on the other hand, shows how some evaluations with hidden dimension of 448 drop in performance, but we thought this may be due to the extreme sequence length values used on those evaluations.

Ablation studies

After finding the best hyperparameters, there are still some parts of either the model functioning and the data used to train it that can be modified in order to improve the performance. These are studies that don't guarantee to achieve better results, but in our case they did, and even exceed the CRF model.

The initial approach was to use, for the input, a context window around the word, defined by sequence length, of the same size before and after the word, that is, the same number of anterior words as posterior. By trying first a proportion of 60% anterior and 40% posterior size, and then 80-20%, the F1 score rises from 0.953 to 0.961. As another study, the data preprocessing is changed. For instance, by not removing the punctuation from the text and replacing all numbers by a constant token, the performance rises to 0.973.

Finally, the architecture of the model is modified by adding a dense layer before the LSTM layer. This is done with the idea of reducing the input dimension for the LSTM, and thus the complexity of the model. By doing so, the performance barely increases, but the inference time slightly decreases.

Also, it was mentioned before that overlapping sequences is used as the strategy to pass sequences of words as input to the model, however, we also considered non-overlapping. The idea was discarded after realizing that this could potentially split actual negation scopes or other important information for the model to process. Likewise, despite finally using a many-to-one BiLSTM model, it was also thought of using a many-to-many architecture, by taking multiple words as input and predicting a tag for each one. But, as the input strategy followed was to pass overlapping sequences, this would result in overlapping outputs, and thus having different tags predicted for the same word. Although this can be solved by only taking the output tag corresponding to the central word in the input sequence, in practice this strategy resulted in no better performance, and more computation, than the many-to-one architecture, which is the one we finally implemented.

Results

All the modifications that raised the performance in both the hyperparameter analysis and the ablation studies are kept. Similar as with the CRF, a 5-fold cross validation confirms the training performance, this time of 0.976. By then evaluating on the test set, a F1 score of 0.977 is achieved, showing no overfitting. The training took 4 minutes, and the test set (punctuation included) is processed in 22 seconds, 0.34 seconds per document.

The BiLSTM shows a slight improvement over the machine learning model. The extensive hyperparameter tuning and ablation studies demonstrate the robustness and efficacy of the BiLSTM approach for detecting negation and uncertainty in medical texts.

Conclusions

In this study, we have implemented a bidirectional Long-Short Term Memory (BiLSTM)-based methodology to address the challenge of detecting negation and uncertainty in medical texts. Our approach treats the problem as a Named Entity Recognition (NER) task, where each word in the text is assigned a tag that indicates its role in expressing negation or uncertainty. The (BiLSTM) model predicts one of these tags for each word in a sequence, leveraging comprehensive lexical, syntactic, and positional features for effective training.

Comparatively, the BiLSTM model shows just a slightly superior performance to the CRF algorithm, demonstrating that traditional machine learning models can still be competitive in specific natural language processing tasks within the medical domain. Nevertheless, the enhancement of the dataset and the parameters number could imply a higher performance on the deep learning model.

On the whole, our study explored the efficacy of different methodologies for detecting negation and uncertainty cues in medical texts. The Conditional Random Field (CRF) model outperformed the rule-based NegEx algorithm, exhibiting higher precision and recall and better capturing the complexity of medical language. On the other hand, the bidirectional Long-Short Term Memory (BiLSTM) model showed only a slight performance edge over the CRF model. This indicates that while deep learning approaches like BiLSTM have potential, traditional models like CRF remain robust and competitive, particularly with well-prepared data and comprehensive feature sets. These findings highlight the

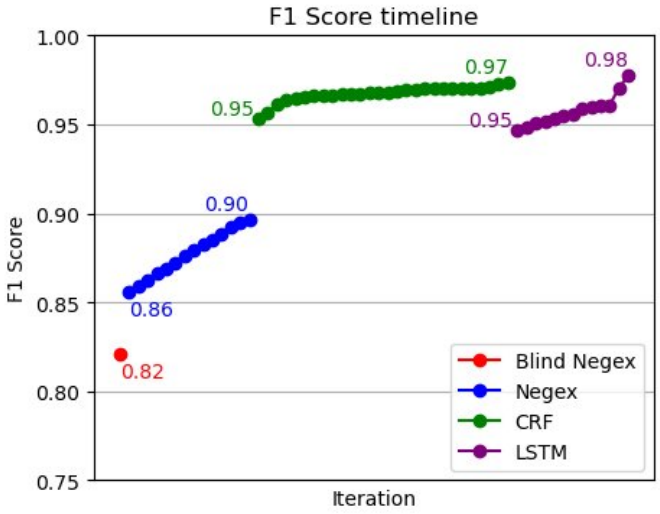

Figure 7: F1 scores obtained throughout this project for the different approaches, including the performance optimizations achieved by hyperparameter tuning. Training performance is included in this plot.

importance of tailored approaches in improving the accuracy and reliability of medical text analysis, ultimately supporting better clinical decision-making.