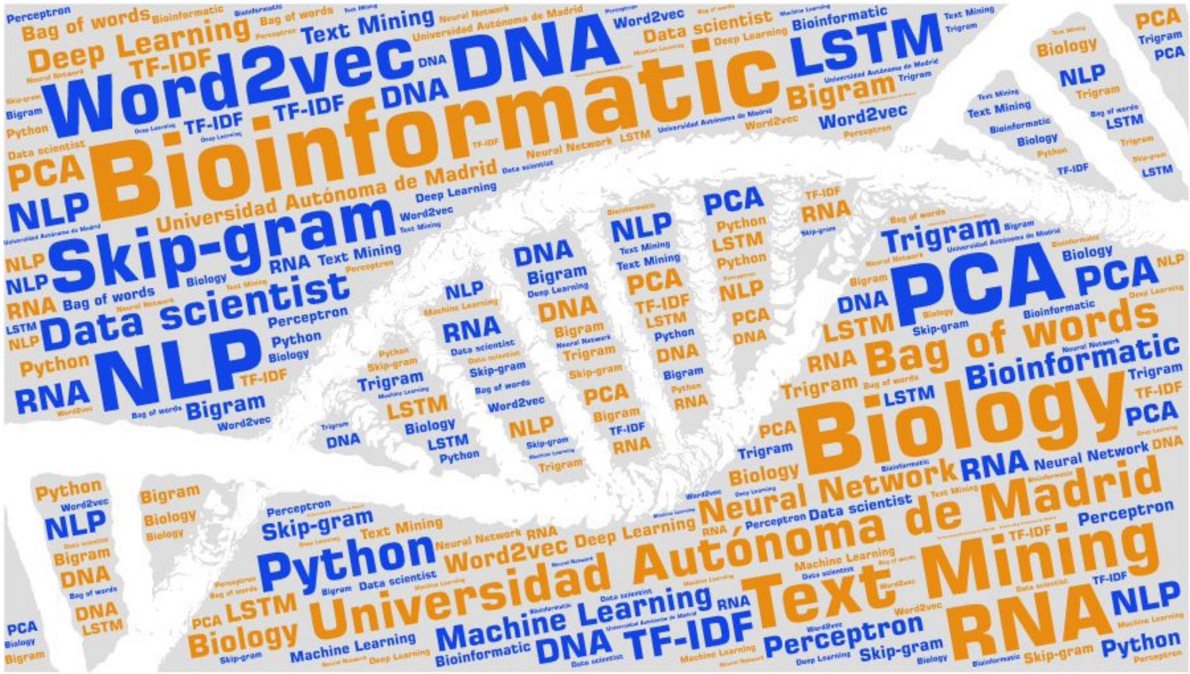
# Text Mining: Multi-Class Biomedical Text Classification

*The final project of the Text Mining course*



## Álvaro Huertas García

18/05/2020
Universidad Autónoma de Madrid - MSc Bioinformatic and Computational Biology
Social media: linkedIn, github

## Introduction

In this project we will proceed to classify biomedical texts belonging to 5 categories: Alzheimer's Disease, Bladder Cancer, Cervical Cancer, Breast Cancer and Negative. Two approaches will be used to carry out the classification. Firstly, machine learning models from Scikit-learn are used combined with appropriate text processing and coding. Secondly, recurrent neural networks with Keras are used. The aim of the work is to show the steps carried out in each approach, the problems faced and the combinations of architecture and pipeline that show the best results classifying biomedical texts. The code corresponding to the results shown in this project are available at:

- https://drive.google.com/drive/folders/1WJXBgIY5cCOCFdzYjo2qT2LVfpkc7GWi?usp=sharing
- https://github.com/Huertas97/Text_Mining

## Machine learning models with Scikit learn

The project starts with 5000 biomedical documents belonging to 5 different classes. The origin of the texts is unknown but not the category to which they belong. Therefore, a supervised learning approach is used, where a model is developed that fits the nature of the data and allows the extraction of the necessary information to correctly classify new texts in these categories.

Different stages are carried out:

1. Building and visualization of the data set
2. Pre-processing of the documents
3. Splitting in training and test sets
4. Features extraction and vectorization
5. Building machine learning method and testing

### Building and visualization of the data set

The data set of the work is constructed from 5 ".csv" files, each corresponding to a category (Alzheimer's Disease, Bladder Cancer, Cervical Cancer, Breast Cancer and Negative.). After the aggregation of all the documents we can observe that the distribution of samples by category is uniform, with 1000 documents by category (Figure 1).

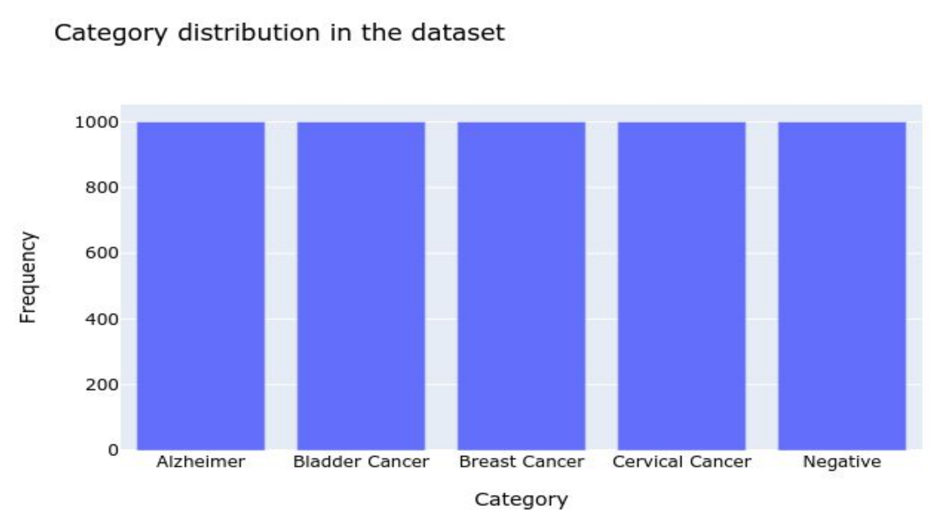


**Figure 1:** Distribution of the documents by category in the data set

It is important to check that the different categories are equally balanced because standard algorithms have difficulties leading with unbalanced data sets. As Susan Li points out in her article titled Multi-Class Text Classification with Scikit-Learn [1]:

> *“Conventional algorithms are often biased towards the majority class, not taking the data distribution into consideration. In the worst case, minority classes are treated as outliers and ignored. For some cases, such as fraud detection or cancer prediction, we would need to carefully configure our model or artificially balance the dataset, for example by undersampling or oversampling each class”*

## Pre-processing of the data

The texts are pre-processed in order to clean up the information present in them. The quality of the information received by the machine learning models is a bottleneck of the classification. Pre-processing allows for the elimination of any distortion present in the data and thus ensures correct feature extraction.

The cleaning of the documents is done on the whole of the texts. The criteria used for this stage of pre-processing are:

- Text is transformed to Unicode codification
- Introduction words for each section are deleted

- Convert all text to lower case
- Word substitution:
    a. P values higher than 0.05 are replaced by the term “lppv” (Low Priority P-value)
    b. P values lower or equal to 0.05 are replaced by the term “hppv” (High Priority P-value)
    c. Isolated numbers are replaced by the term “NUM”
- Text tokenization:
    a. Punctuation marks are deleted from tokens
    b. English stop words are deleted from tokens
- Text lemmatization

To mention some of the steps carried out in the pre-processing, in the first step all terms are encoded to Unicode. This allows terms present in scientific texts in ASCII encoding to be treated uniformly. For instance, the mathematical symbol plus-minus sign (±) (177 ASCII codification) is transformed to “+ -”, as this code box shows:

```
import unidecode
text = chr(177)
print("ASCII character:", text)
text = unidecode.unidecode(text)
print("Unicode character:", text)
```

```
>> ASCII character: ±
>> Unicode character: +-
```

**Code Box 1:** Distribution of the documents by category in the data set

The substitution of the p-values is made to facilitate the substitution of the isolated numbers and to group all the statistically significant or non-significant results in a single term. Detailed examples of the effect of these substitutions on texts are shown in the notebook of "RNN_Multiclass_Biomedical_text_classfication.ipynb".

The tokenization of the text is carried out to facilitate the removal of punctuation marks and the well-known stop words. In this step, the punctuation list from the string module is used. Moreover, four different stop word lists are also used: the one belonging to NLTK (179 stop words), the one belonging to Spacy (326 stop words), the one extracted Ganesan, Lloyd & Sarkar, 2016 in "Discovering Related Clinical Concepts Using Large Amounts of Clinical Notes" (811 stop words, named as “Clinical”) and the one available in this web resource (https://countwordsfree.com/stopwords) (853 stop words, named as “Long”).

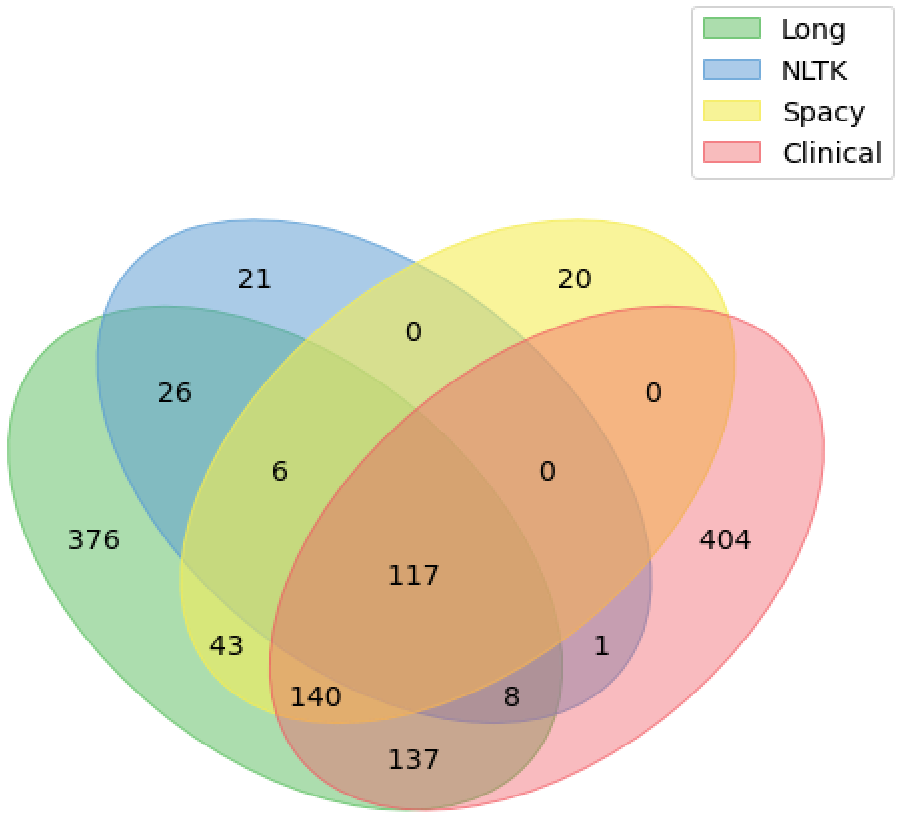


**Figure 2:** Venn diagram showing the similarity between the different stop word sources

As can be seen in Figure 2, the different stop word lists have 117 words in common. Of the 179 stop words in NLTK 123 are contained by Spacy. However, there are differences. For example, the two largest lists (Clinical and Long) differ from each other, presenting 404 and 376 unique terms with respect to the rest of the lists; sharing only 137 terms with each other. Thus, as discussed later in this project, we study which of the lists above performs a better classification result.

Finally, lemmatization is performed to reduce the syntactic variability present in the texts. Consequently, we manage to reduce the dimensions of the vectorized texts. Lemmatization is used instead of stemming because lemmatization provides existing words, whereas stemming provides the root of the word, which may not be an existing one [2].

## Splitting in training and test sets

To ensure that the classifier results are not biased, the pre-processed texts are separated into a training set and a test set. The SciKit-Learn `model_selection.train_test_split` function is used. The proportion of testing corresponds to 30% of the total (1500 documents), leaving 3500 documents for training. At this point, it is also interesting to check the proportion of the documents in each set.

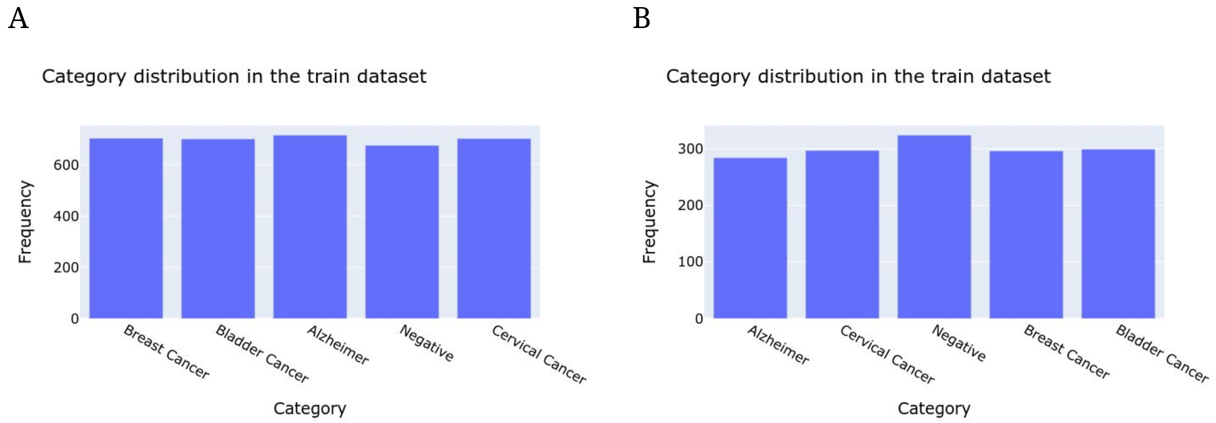


**Figure 3:** Distribution of the different categories in the training set (A) and test (B).

## Features extraction and vectorization

One of the most remarkable features of the NLP is the structuring of the data. Indeed, text documents are considered to be unstructured data. However, these unstructured text sequences must be converted into a structured feature space when using mathematical modeling as part of a classifier [3]. After the data has been cleaned, feature extraction methods can be applied. In this project, the techniques used for feature extractions are Term Frequency-Inverse Document Frequency (TF-IDF) and Word2Vec.

Term frequency, also called bag-of-words, is the simplest technique of text feature extraction. This method is based on counting the number of words in each document and assigning it to the feature space [3]. The TF-IDF technique allows overcoming the problem of bag-of-words (BoW) to represent properly the meaning of each processed document by penalizing the word frequency over the documents, in order to reduce the effect of common words in the corpus.

However, in the TF-IDF method the semantic meaning of the words and the order that they appear in the text is not considered. This issue raises problems to understand sentences within the model [3]. Moreover, this problem becomes more important in neural networks. One might think that the use of n-grams would solve this problem, but n-grams does not solve this problem so a similarity needs to be found for each word in the sentence. To solve this problem one approximation is the word embedding.

Word embedding is a feature learning technique in which each word or phrase from the vocabulary is mapped to a N dimensional vector of real numbers [3]. The Word2Vec approach uses shallow neural networks with two hidden layers, continuous bag-of-words (CBOW), and the Skip-gram model to create a high dimension vector for each word [3].

| **Feature Extraction Technique** | **Parameters** | **Source** | **Notebook** |
|---|---|---|---|
| TD-IDF | 1-<br>Lemmatizer = ?<br>Stop words = ?<br>max_features=5000,<br>min_df=3,<br>max_df=0.2,<br>strip_accents='unicode',<br>analyzer='word',<br>ngram_range=(1, ?)<br>2-<br>max_features= ?,<br>min_df=3,<br>max_df=0.2,<br>strip_accents='unicode',<br>analyzer='word',<br>ngram_range=(1, 2)<br>3-<br>max_features= ?,<br>min_df=?<br>max_df=?<br>strip_accents='unicode',<br>analyzer='word',<br>ngram_range=(1, 2)<br>+<br>TextBlob:<br>polarity sentiment<br>subjectivity sentiment<br>spelling correction | `sklearn.feature_extraction.text TfidfVectorizer`<br>`textblob` | 1, 2- ML_biomedical.ipynb,<br>3- `TF-IDF_Pol_sub.ipynb` |
| Word2Vec | size=vec_dim,<br>min_count=10, window=10,<br>iter=10 | `gensim.models.word2vec Word2Vec` | ML_biomedical |
| **Table 1:** Parameters used in the different approaches used to extract features information | | | |

In this project, both techniques will be used in combination with different classification models in order to compare which gives the best result. Table 1 shows the different strategies used to extract the characteristics of both techniques. The red question marks refer to the parameter explored.

Thus, in a first step (TF-IDF Parameter 1), the performance of the classifiers are compared according to the lemmatizer (NLTK or Spacy), the list of stop words mentioned above and the appropriate N-grams (uni, bi, trigram). This is how we select the right lemmatizer, stop words and N-grams. In the following strategy (TF-IDF Parameter 2), we develop the dimensions of the vocabulary. Finally (TF-IDF Parameter 3), we perform a feature extraction including the sentiment of polarity and subjectivity, and correct the spelling with TextBlob. In the results section below, the optimal parameters found are commented on.

### Building machine learning method and testing

Eight different classifiers are used to classify the texts:

1. Naïve Bayes Classifier (NB)
2. Logistic Regression (LR)
3. Logistic Regression with Lasso Regularization (LRS, Log. Reg. L1)
4. Logistic Regression with Ridge Regression (LRR, Log. Reg. L2)
5. K-Nearest Neighbor (KNN)
6. Random Forest (RF)
7. Support Vector Machine (SVM)
8. Stochastic Gradient Descent (SGD)

All classifiers employed follow a supervised learning strategy. This variety of classifiers has been used since there is no gold standard and they have been used previously in text classification [3]. Each of them has advantages and disadvantages:

| Model | Advantages | Disadvantages |
|---|---|---|
| Naïve Bayes | • It works very well with text data<br>• Easy to implement<br>• Fast in comparison to other algorithms | • A strong assumption about the shape of the data distribution (assumes independence) |
| Logistic Regression | • Easy to implement<br>• Does not require too many computational resources<br>• It does not require any tuning | • It cannot solve non-linear problems<br>• Prediction requires that each data point be independent |
| Logistic Regression Lasso Regularization | • L1 norm<br>• Introduce sparsity in the data as the | • It does require tuning of the hyperparameter of regularization "C" |

| | | |
|---|---|---|
| | coefficient can be equal to 0. Feature selection.<br>• Reduce risk to overfitting | |
| Logistic Regression Ridge Regression | • L2 norm<br>• Reduce weights of coefficients<br>• Reduce risk to overfitting<br>• Helps to reduce the model complexity | • It does require tuning of the hyperparameter of regularization "C"<br>• It does not introduce sparsity. No feature selection |
| K-Nearest Neighbor | • Effective for text data sets<br>• Non-parametric<br>• More local characteristics of text or document are considered<br>• Naturally handles multi-class data sets | • Difficult to find optimal value of k<br>• Constraint for large search problems to find nearest neighbors<br>• Finding a meaningful distance function is difficult for text data sets |
| Random Forest | • Ensembles of decision trees are very fast to train in comparison to other techniques<br>• Reduced variance (relative to regular trees) | • More trees in forest increases time complexity in the prediction step<br>• Not as easy to visually interpret<br>• Overfitting can easily occur<br>• Need to choose the number of trees at forest |
| Support Vector Machine | • SVM can model non-linear decision boundaries<br>• Performs similarly to logistic regression when linear separation<br>• Robust against overfitting problems (especially for text data set due to high-dimensional space) | • Lack of transparency in results caused by a high number of dimensions (especially for text data).<br>• Choosing an efficient kernel function is difficult (susceptible to overfitting/training issues depending on kernel)<br>• Memory complexity |
| Stochastic Gradient Descent | • It has unbiased estimate of gradients.<br>• The more the examples, the lower the standard error.<br>• Straight trajectory towards the minimum | • Sometimes calculating the gradient can be very expensive if the size of the data is large.<br>• Necessity of picking a correct step size |

**Table 2:** Advantages and disadvantages of the classification model used

## Results and Discussion

### TF-IDF

Firstly, the selection of the optimal lemmatizer, stop words and N-grams is carried out. This is done by comparing the results of the different classifiers for the different combinations of these parameters.

In Table 3 it can be seen how the NLTK lemmatizer in combination with the longer stop word list and the use of unigrams and bigrams shows the best classification result with Random Forest (87.8%). Accuracy results for all models can be found in Appendix A.

| Stop words | Lemma | N-gram | Best Model | Accuracy |
|---|---|---|---|---|
| **Long** | **NLTK** | 1 | Log. Reg. L1 | 82.46 |

| | | | | |
|---|---|---|---|---|
| | | **1-2** | **Random Forest** | **87.8** |
| | | 1-3 | Log. Reg. L1 | 87.2 |
| NLTK | NLTK | 1 | Log.Reg. L1 | 82.47 |
| | | 1-2 | Log. Reg. L1 | 87.07 |
| | | 1-3 | Random Forest | 87.27 |
| Clinical | NLTK | 1 | Log.Reg. L1 | 82.4 |
| | | 1-2 | Random Forest | 87.4 |
| | | 1-3 | Log. Reg. L1 | 87.27 |
| Spacy | NLTK | 1 | Log. Reg. L1 | 82.46 |
| | | 1-2 | Log. Reg. L1 | 87.07 |
| | | 1-3 | Random Forest | 87.27 |
| Spacy | Spacy | 1 | Log. Reg. L1 | 83.27 |
| | | 1-2 | Log. Reg. L1 | 87.6 |
| | | 1-3 | Log. Reg. L1 | 87.4 |

**Table 3:** Results of the best classifier to determine the best combination of lemmatizer, stop words and N-grams. For this step, vocabulary size was set to 5000 words.

The next step in TF-IDF is to establish the optimal value of the vocabulary using the optimal parameters obtained previously. For this purpose, different vocabulary sizes were tested: 10, 500, 1000, 5000, 10000 and 50000. The results are shown in Table 4.

| Vocabulary Size | Best Model | Accuracy |
|---|---|---|
| 10 | Random Forest | 73.07 |
| 500 | Random Forest | 86.93 |
| 1000 | Log. Reg. L1 | 87.33 |
| **5000** | **Random Forest** | **87.8** |
| 10000 | Random Forest | 87.27 |
| 50000 | Random Forest | 86.8 |

**Table 4:** Classification results Vocabulary Size Selection

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Alzheimer | 0.94 | 0.95 | 0.95 | 284 |
| Bladder Cancer | 0.91 | 0.93 | 0.92 | 299 |
| Breast Cancer | 0.90 | 0.80 | 0.85 | 296 |
| Cervical Cancer | 0.89 | 0.86 | 0.87 | 297 |
| Negative | 0.77 | 0.85 | 0.81 | 324 |

**Table 5:** Random Forest Classifier metrics (vocabulary size = 5000, lemmatizer: NLTK and Long stop words).

It can be seen how the vocabulary size with the best result is 5000. The size of the vocabulary does not affect it to a great extent, although a drop in accuracy is observed when the size is reduced (size = 10).

Other metrics such as recall (proportion of actual text of a category are correctly classified) and precision (proportion of predicted category belongs truly to that category) are also shown in the classification results. Table 5 shows the metrics of Random Forest for the vocabulary size 5000.

The text classification is good except for the "Negative" group (f1-score: 0.81) and “Cervical Cancer” group (f1-score: 0.87). In the PCA display in Figure 4 you can see how both group are placed close in the space.

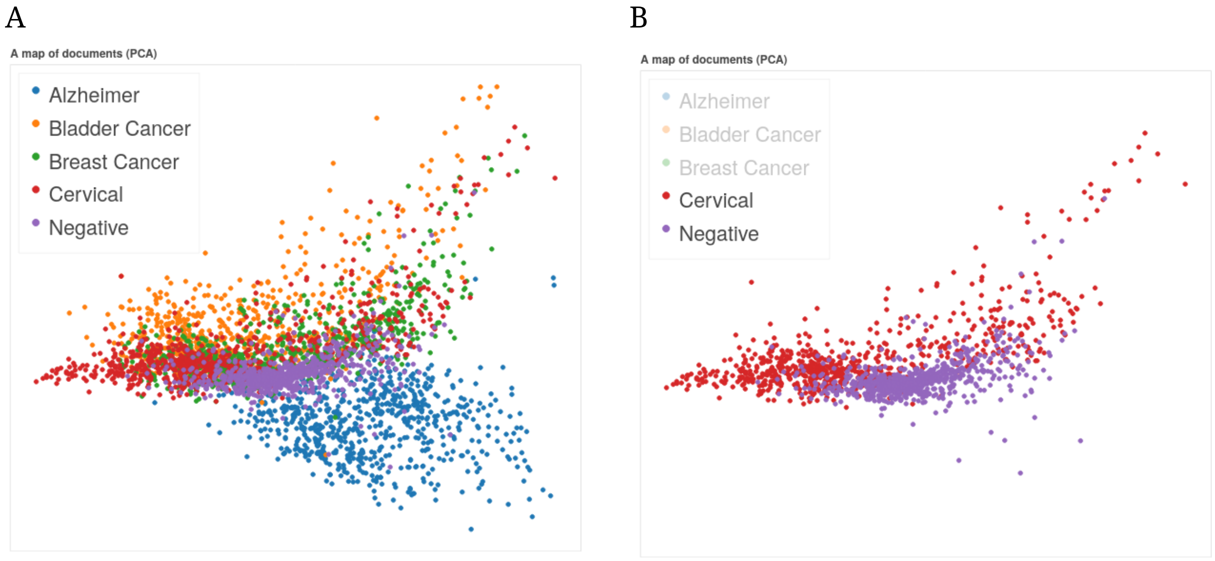


**Figure 4:** PCA visualization vocabulary size = 5000, lemmatizer: NLTK and Long stop word

To attempt to separate these groups and facilitate classification, the max_df and min_df parameters were modified along with the size of the vocabulary. The parameter max_df is used for removing terms that appear too frequently, while min_df is used for removing terms that appear too infrequently.

The table 6 shows different combinations of these parameters, the best being max_df = 0.7, min_df = 0 and vocabulary size = 8500. Also the best classifier is always Logistic Regression with Lasso Regularization. These results show that the inclusion of more frequent words in the vocabulary and the retention of infrequent words allows better separation of texts. The fact that reducing min_df improves the classification could be an indication that the cleaning on the texts is adequate, since lowering min_df increases the risk of introducing noise or stop words, but in our case it helps.

Once we have found the optimal parameters, we proceed to see the effect of sentimental inclusion and spell check. It is also interesting to show the top 100 bigrams identified with these parameters (Figure 5).

| Vocabulary Size | Max_df | Min_df | Best Model | Accuracy |
|---|---|---|---|---|
| 5000 | 0.2 | 3 | Random Forest | 87.8 |
| 5000 | 0.4 | 3 | Log. Reg. L1 | 90.33 |
| 5000 | 0.6 | 3 | Log. Reg. L1 | 90.40 |
| 6000 | 0.6 | 0 | Log. Reg. L1 | 90.53 |
| 7000 | 0.7 | 0 | Log. Reg. L1 | 90.70 |
| 8000 | 0.6 | 0 | Log. Reg. L1 | 90.8 |
| 8000 | 0.6 | 0.1 | Log. Reg. L1 | 88.47 |
| 8000 | 0.7 | 0 | Log. Reg. L1 | 90.80 |
| 8000 | 0.8 | 0 | Log. Reg. L1 | 90.60 |
| **8500** | **0.7** | **0** | **Log. Reg. L1** | **90.87** |
| 10000 | 0.7 | 0 | Log. Reg. L1 | 90.60 |

**Table 6:** Classification results for vocabulary size, max_df, min_df selection over NLTK lemmatizer and Long stop words.

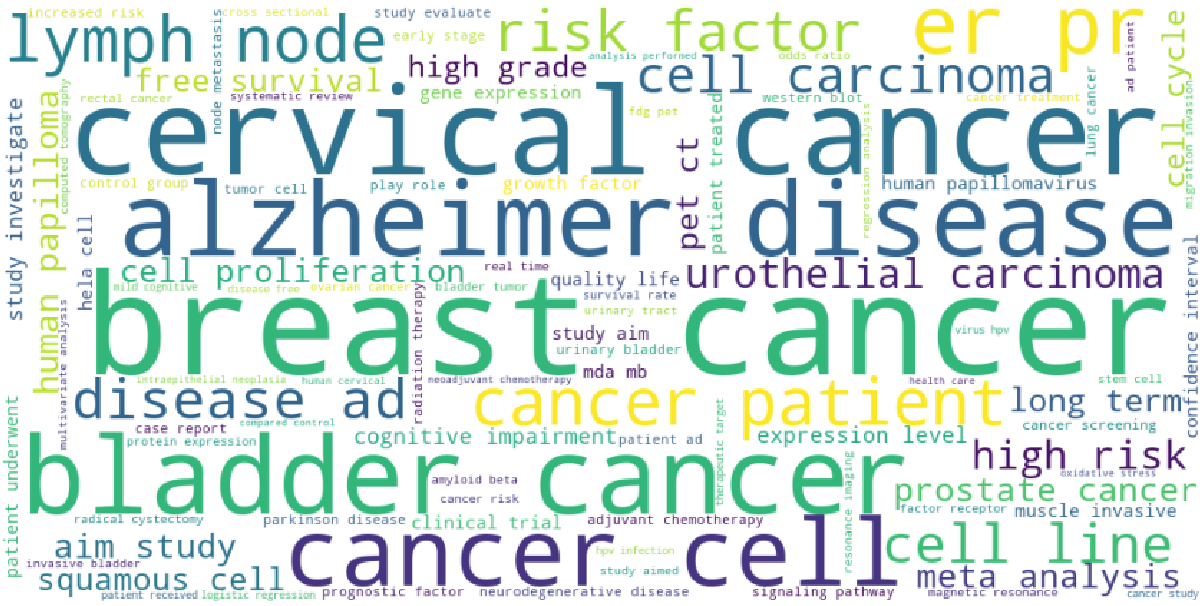
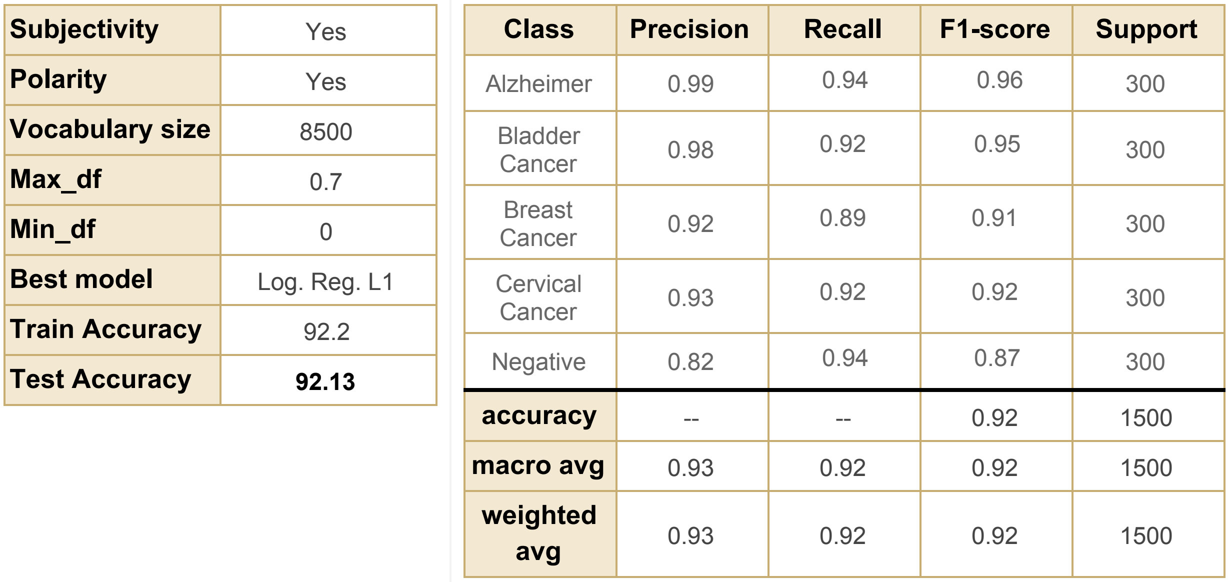


**Figure 5:** Cloud of words from top 100 spell-checked bigrams identified by TF-IDF with optimal parameters (vocabulary size = 8500, max_df = 0.7, min_df = 0, lemmatizer: NLTK, stop words = Long).

It is interesting to note that among the most important bigrams we find the categories of the texts we want to classify.

| | |
|---|---|
| A | B |



| | |
|---|---|
| **Subjectivity** | Yes |
| **Polarity** | Yes |
| **Vocabulary size** | 8500 |
| **Max_df** | 0.7 |
| **Min_df** | 0 |
| **Best model** | Log. Reg. L1 |
| **Train Accuracy** | 92.2 |
| **Test Accuracy** | **92.13** |

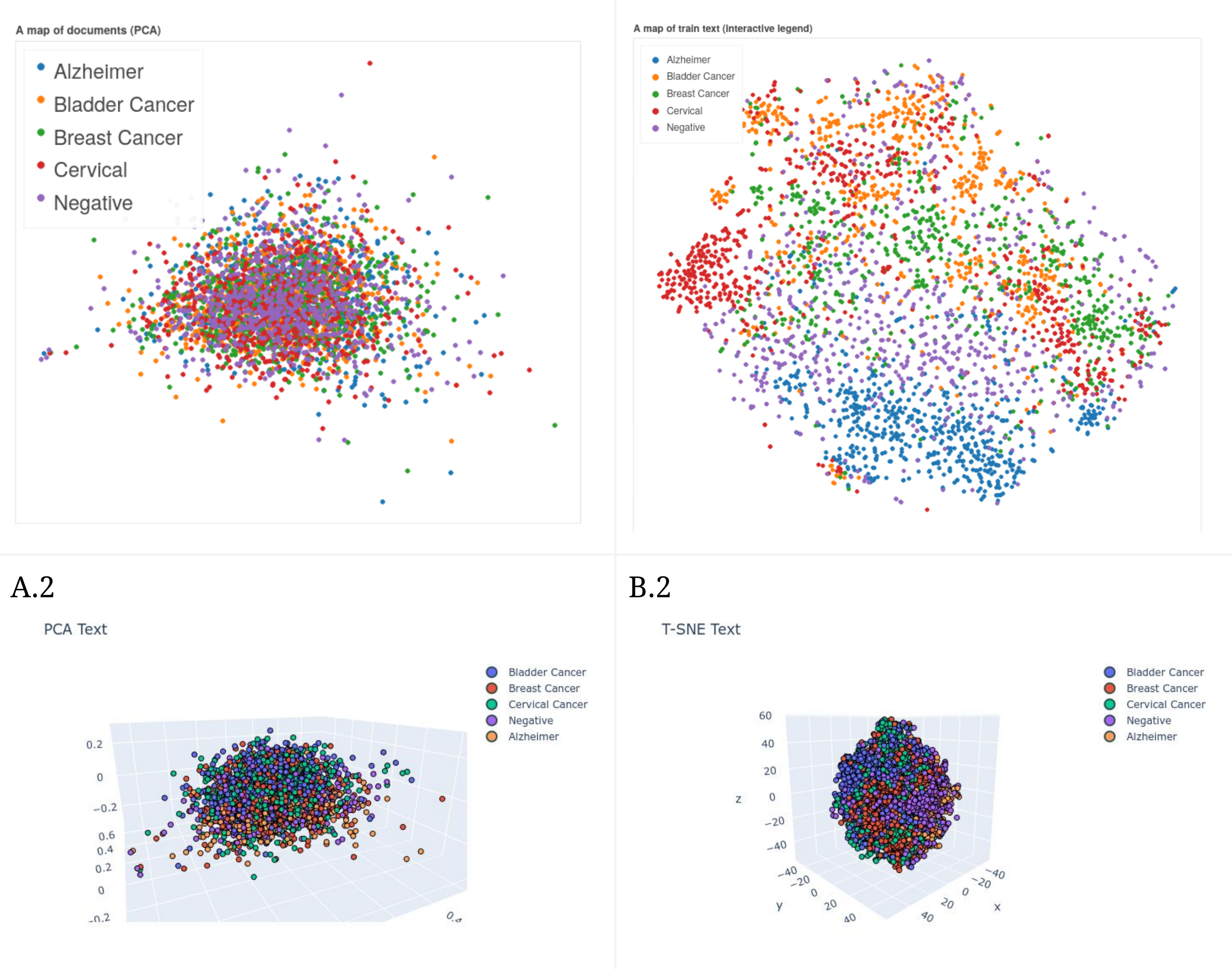
| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Alzheimer | 0.99 | 0.94 | 0.96 | 300 |
| Bladder Cancer | 0.98 | 0.92 | 0.95 | 300 |
| Breast Cancer | 0.92 | 0.89 | 0.91 | 300 |
| Cervical Cancer | 0.93 | 0.92 | 0.92 | 300 |
| Negative | 0.82 | 0.94 | 0.87 | 300 |
| **accuracy** | -- | -- | 0.92 | 1500 |
| **macro avg** | 0.93 | 0.92 | 0.92 | 1500 |
| **weighted avg** | 0.93 | 0.92 | 0.92 | 1500 |

**Table 7:** Parameters used (A) and metrics (B) of the best classification result with TF-IDF sing meta-features of polarity and subjectivity sentiments with TextBlob.

In Table 7 we can observe that the incorporation of the information of the feelings of polarity and subjectivity effectively helps the classification. These characteristics are meta-information of the texts, since they do not contain them explicitly. The model confusion matrix is available in the appendix. We can also check that there is no overfitting since the train accuracy is 92.2 % and test accuracy is 92.13%. To try to improve this result, the Adaboost ensemble method was tried. The result of the Adaboost ensemble method is also shown in Appendix A and in the Adaboost.ipynb notebook.

Furthermore, we can see how the spatial distribution in PCA changes when these parameters are applied (Figure 5). In 2D we do not observe a good classification, but we should never forget that 8502 dimensions are used for each text (2 additional to the 8500 of the vocabulary pertaining to subjectivity and polarity). Moreover, the variance explained by the first two components is only 2.81% and 1.47%. Thus, the PCA display is only shown to demonstrate that the inclusion of the meta-information changes the arrangement of the texts in space (comparison Figure 4 vs Figure 6).

| | |
|---|---|
| A.1 | B.1 |



A.2

B.2

**Figure 6:** PCA visualization (A.1, A.2) and T-SNE visualization (B.1, B.2) of the best classification result with TF-IDF sing meta-features of polarity and subjectivity sentiments with TextBlob.

These representations are according to how the most frequent words are distributed by each category (Figure 7). We see how the texts of "Alzheimer" are separated from the others because they present more frequent words different from the rest of the classes. On the contrary, the "Negative" class presents more common words distributed and shared by the rest of the classes. This makes classification difficult.

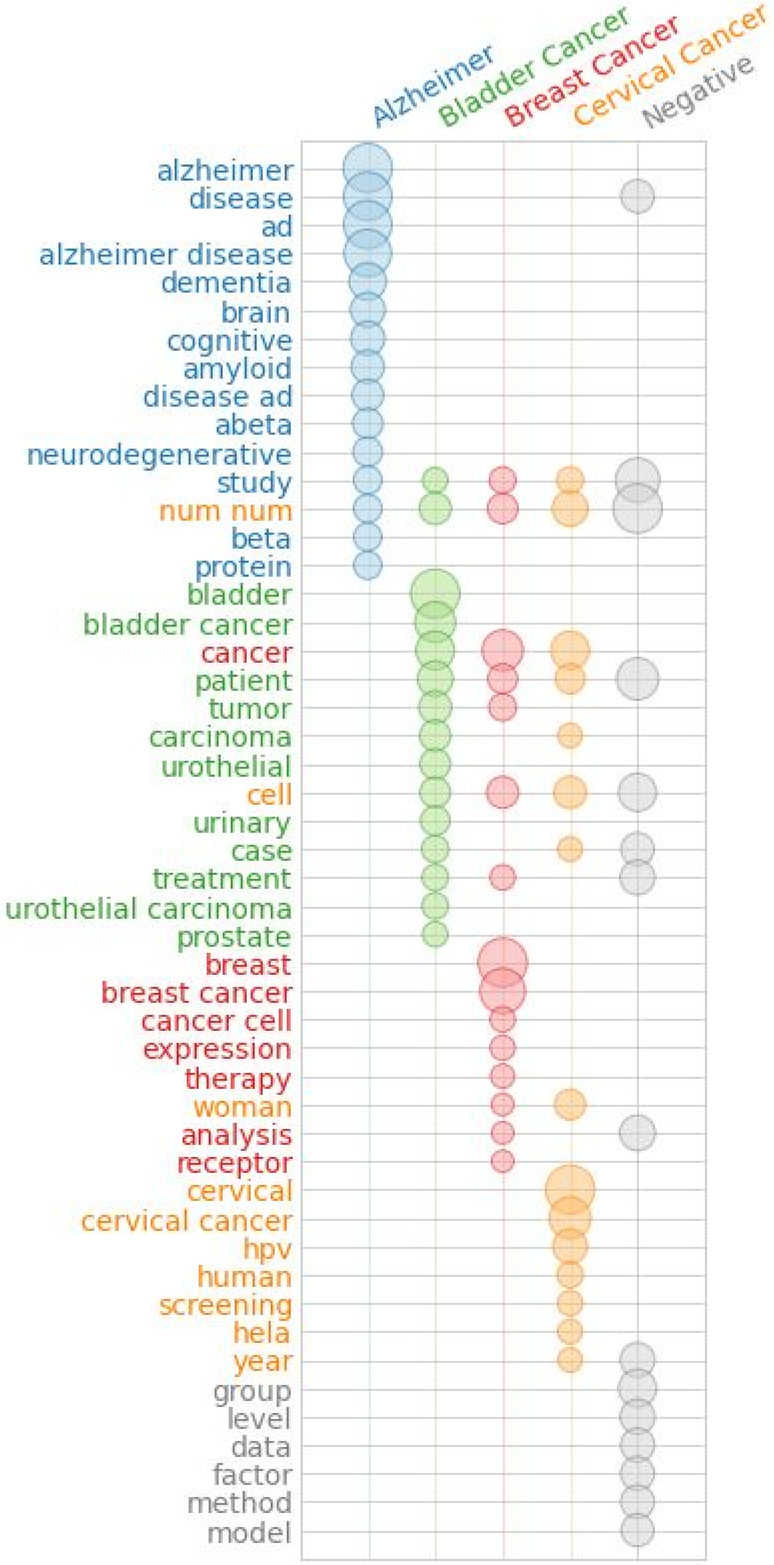


**Figure 7:** Representation of the 15 most common words for each type of text after TF-IDF vectorization. Made with textacy.

### Word2vec

The second strategy uses Ginsem. The two parameters that are optimized are the dimensions of the embedding, the size of the context window and the elimination of infrequent words. The dimension of the embedding are important because set the dimension to represent each token word. The context window is also important because it establishes the maximum distance between a target word and its neighbors. In theory, a smaller window should give you terms that are more related [5].

Along with Word2vec embeddings, TF-IDF vectorizer IDF values are applied. The previously selected values for the parameter are used for this step.

We see that the best combination of parameters is to use 128 dimensions in the embeddings, word removal with a frequency of less than 10 documents and a context window of 10 words.

| Embedding size | Window size | Min count | Best Model | Accuracy |
|---|---|---|---|---|
| 32 | 5 | 10 | Log. Reg. L1 | 79.07 |
| 32 | 5 | 100 | Log. Reg. L1 | 80.4 |
| 32 | 10 | 10 | Log. Reg. L1 | 82.87 |
| 32 | 10 | 100 | Log. Reg. L1 | 81.53 |
| 64 | 5 | 10 | Log. Reg. L1 | 82.53 |
| 64 | 5 | 100 | Log. Reg. L1 | 82.2 |
| 64 | 10 | 10 | Log. Reg. L1 | 84.7 |
| 64 | 10 | 100 | Log. Reg. L1 | 83.67 |
| 128 | 5 | 10 | Log. Reg. L1 | 84.33 |
| 128 | 5 | 100 | Log. Reg. L1 | 84.6 |
| **128** | **10** | **10** | Log. Reg. L1 | **85.87** |
| 128 | 10 | 100 | Log. Reg. L1 | 85.0 |
| 200 | 5 | 10 | Log. Reg. L1 | 84.73 |
| 200 | 5 | 100 | Log. Reg. L1 | 85.0 |
| 200 | 10 | 10 | Log. Reg. L1 | 85.53 |
| 200 | 10 | 100 | Log. Reg. L1 | 84.87 |

**Table 8:** Exploring the best combination of parameters for word embedding with Gensim's Word2Vec

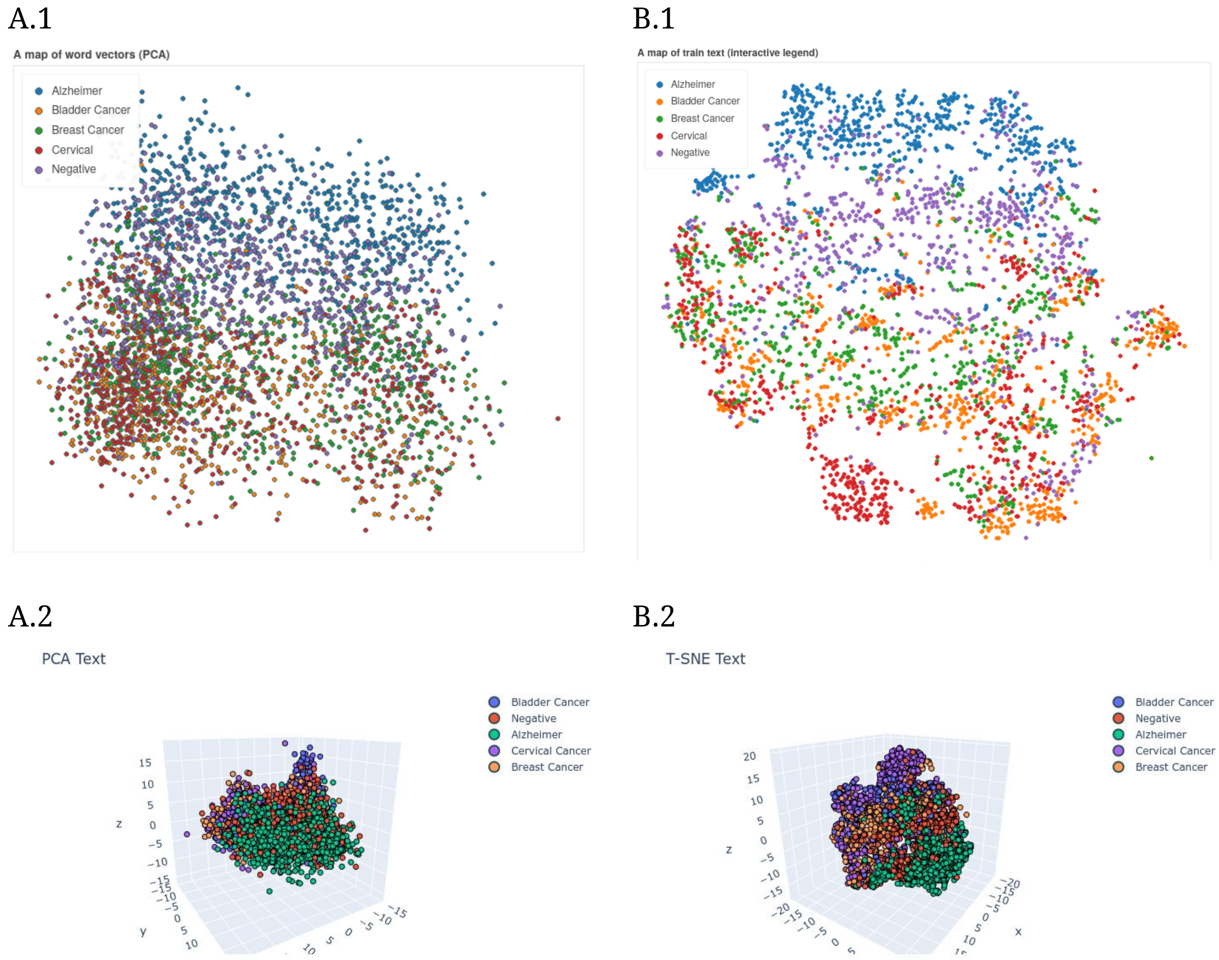


**Figure 8:** PCA visualization (A.1, A.2) and T-SNE visualization (B.1, B.2) of the best classification result with Word2Vec using IDF values, window = 10, min_count = 10, embedding size = 128.

In table 8 we can see how the maximum accuracy of the classification with Word2Vec is lower than that achieved with Logistic Regression L1 regularization (86 % < 92%). We can also see in the Figure 8 that the layout in document space is different in both approaches.

The table with the best Word2Vec model metrics is available in Appendix A. Again, we check that the problem lies in the classification of the "Negative" texts, since the the recall is 81% and precision is 75%. This means that only 81% texts truly belonging to “Negative” are correctly classified, and only 75% of the texts classified as “Negative” are truly “Negative”.

Depending on our interest we will focus more on recall or precision. Considering that the "Negative" class is a mix, if we are interested in classifying the texts so that there are no texts unrelated to that category and we do not mind losing some documents from other categories, we will be interested in improving the precision. On the other hand, if we do not want to lose any documents from the other classes and we don't mind including some "Negative" documents we will look at the recall.

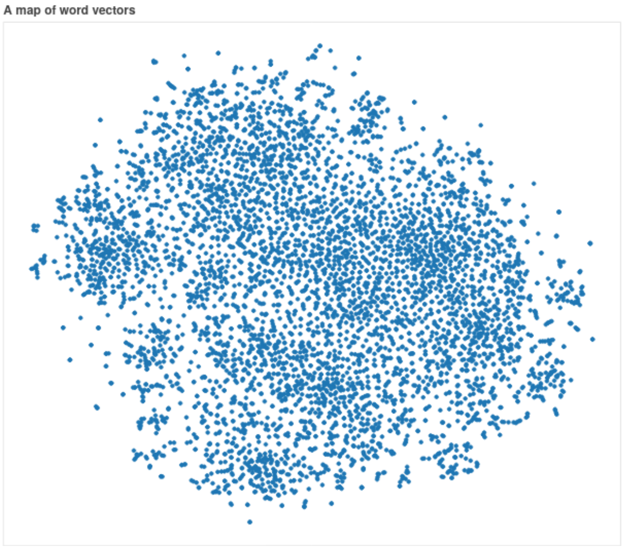


**Figure 9:** Visualization of the vocabulary from best classification result with Word2Vec using IDF values, window = 10, min_count = 10, embedding size = 128.

## Deep Learning: Recurrent Neural Networks with Keras

In this project, other classification alternatives have also been explored, such as neural networks with deep learning. Although the main steps and measures taken to build the working pipeline with deep learning will be discussed, it is recommended to see the notebooks:

1. Emb_propio.ipynb
2. Scispacy_bigrams.ipynb
3. Scispacy_emb.ipynb
4. Emb_BIOBERT.ipynb

The first notebook shows the procedure followed for the pre-processing of the text and the construction of a neural network that builds the embeddings from the texts. In order to help the RNN to achieve good embeddings the label "bigram" is added on the most frequent bigrams identified with Countvectorizer to the text. In the notebook, there are example that show how is done and the results, but in order to clarify here is an example. Imagine "breast cancer" is identified as one of the top bigrams. Now the sentence "... patients with **breast cancer** require 20 ml ..." is transformed into "... patients with **bigram breast cance**r require 20 ml ...".

In a the second notebook,  it shows the incorporation of labels into text by recognizing Genes and Gene Products and Diseases with ScispaCy. The models used from ScispaCy are "en_ner_bionlp13cg_md" (a spaCy NER model trained on the BIONLP13CG corpus.) and "en_ner_bc5cdr_md" (a spaCy NER model trained on the BC5CDR corpus). This approach is used for the same purpose as the one before, but adding more labels. As before, there are examples in the notebook, but here is an example. Imagine the sentence "Prohibitin promotes **androgen receptor** activation in ER-positive **breast cancer**". Also, CountVectorizer has identified "breast cancer" as a bigram and ScispaCy as a disease and "androgen receptor" as a Gene_Or_Gene_Product. The sentences will be transformed as follows: "prohibitin promotes **androgen receptor bioent** activation in er-positive **bigram breast cancer(disease)**".

In the third approach ScispaCy embedding from "en_core_sci_lg" is used. These embeddings come from a full spaCy pipeline for biomedical data with a larger vocabulary and 600k word vectors. The latest approach is adapted from the guide that is available at https://medium.com/@armandj.olivares/a-basic-nlp-tutorial-for-news-multiclass-categori zation-82afa6d46aa5 [7]. Different architectures and parameters are tested as shown below.

The latest notebook uses BIOBERT's embeddings. BIOBERT is a pre-trained biomedical language representation model for biomedical text mining. More information on the process followed can be found in the notebook Emb_BIOBERT.ipynb.

Deep learning models have achieved state-of-the-art results across many domains, including a wide variety of NLP applications. Deep learning for text and document classification includes three basic architectures of deep learning: Deep Neural Networks, Recurrent Neural Network (RNN) and Convolutional Neural Networks (CNN) [3].

The most important difference between CNN and RNN is that CNN is a feed-forward neural network, that means the information only flows in the forward direction. CNN are quite useful for recognize patterns across space (for example in images). On the other hand, RNN is a recurrent neural network, therefore information flows back and forth. This characteristic allows RNN to have "memory" because the neural network is trained to recognize patterns across time [7]. RNN assigns more weights to the previous data points of a sequence. Therefore, this technique is a powerful method for text, string, and sequential data classification [3]. For this reason we will use RNN in the project.

RNN mostly works by using Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) for text classification, which contains input layer (word embedding), hidden layers, and finally output layer. GRUs are a simplified variant of the LSTM architecture. However, a GRU differs from LSTM because it contains two gates, giving a more flexible in controlling the weights, and a GRU does not possess internal memory [3].

## Pre-processing data

The criteria used for this stage of pre-processing are:

- Introduction words for each section are deleted
- Convert all text to lower case
- Word substitution:
    a. P values higher than 0.05 are replaced by the term "lppv" (Low Priority P-value)
    b. P values lower or equal to 0.05 are replaced by the term "hppv" (High Priority P-value)
    c. Isolated numbers are replaced by the term "NUM"

- ** Adding labels (optional)
    - ** Addition of the label "bigram" to the top N bigrams identified in the corpus (in Emb_propio.ipynb, Scispacy_bigrams.ipynb)
    - ** Addition of the label “bioent” for "GENE_OR_GENE_PRODUCT"
    - entities recognized by ScispaCy (only in Scispacy_bigrams.ipynb)
    - ** Addition of the label “disease” for “DISEASE” entities recognized by ScispaCy (only in Scispacy_bigrams.ipynb)
- **Adding metadata (optional):
    - Addition of subjectivity and polarity features (only in Scispacy_emb.ipynb )

### Splitting in training, test and validation sets

The vocabulary (top_words) that will be used in the many to one Recurrent Neural Network (RNN) will be extracted only from the training data (train) to avoid over-fitting. Similarly, the cleaning of the texts will be done for training and test separately. In the first approach the bigrams are extracted using Countvectorizer (Bag of Words) only on the training. Similarly, in the second approach using ScispaCy recognized entities, only training is used.

The division that will be used to extract the training data set will be: 70% for training data, the remaining 30% will be divided equally in two parts, 15% validation and 15% test.

### Features extraction and adding labels

The labels identified in the pre-processing stage are added to the documents. Examples of this process are explained in the notebooks corresponding to each approach. Similarly, an example has been shown above.

In Figures 10, 11 and 12 we can see the characteristics extracted from the texts that have been used to facilitate the generation of the embeddings to the neural network. Figure 10 shows the 20 most abundant bigrams in the training corpus after filtering. The 100 most abundant bigrams in the corpus are labeled as bigram in the original text. Figure 11 and 12 show the biological entities identified by ScispaCy. These entities are also tagged to help generate the embeddings. It is interesting to note that in all cases the identified words are biologically relevant.

In notebook Emb_propio.ipynb only the bigrams are added, while in the second notebook Scispacy_bigrams.ipynb both the bigrams and the bio entities identified by

ScispaCy are added to facilitate the creation of the embeddings.

In the last notebook Scispacy_emb.ipynb, ScispaCy pre-trained embeddings are used as an alternative. To try to facilitate the classification, meta information is also added. As was done in the classification with the classic machine learning methods, the subjectivity and polarity identified by TextBlob is used.

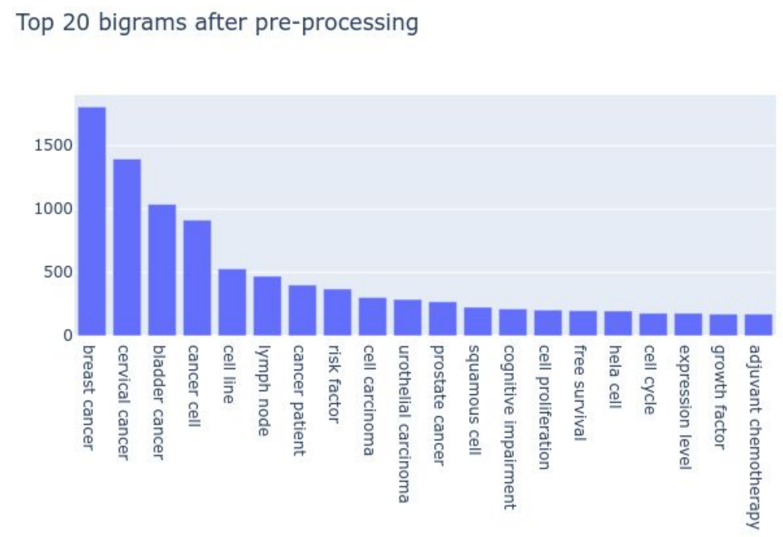


**Figure 10:** Top 20 bigrams identified in the training text These terms will be labeled "bigram" in the first and second neural network approaches: Emb_propio.ipynb, Scispacy_bigrams.ipynb

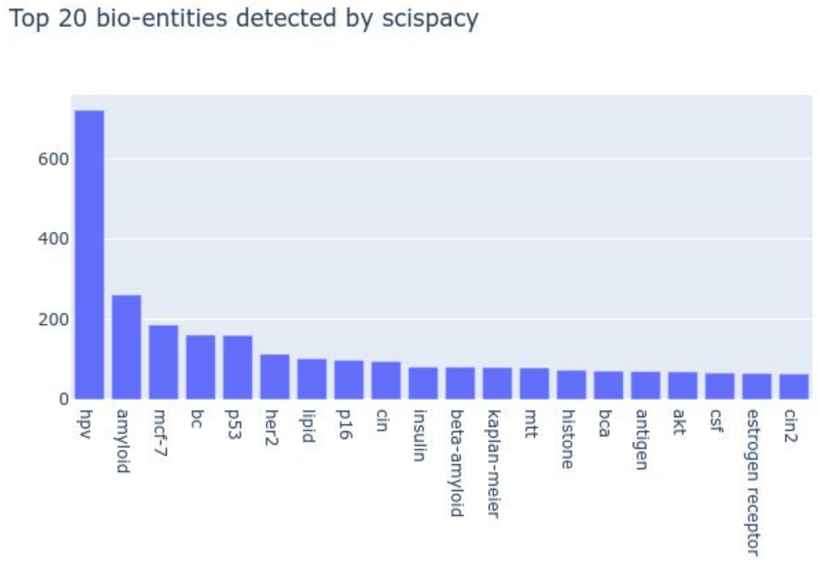


**Figure 11:** Top 20 Genes or Genes Products (bio entities) identified by ScispaCy in the training text These terms will be labeled "bioent" in the second neural network approach: Scispacy_bigrams.ipynb

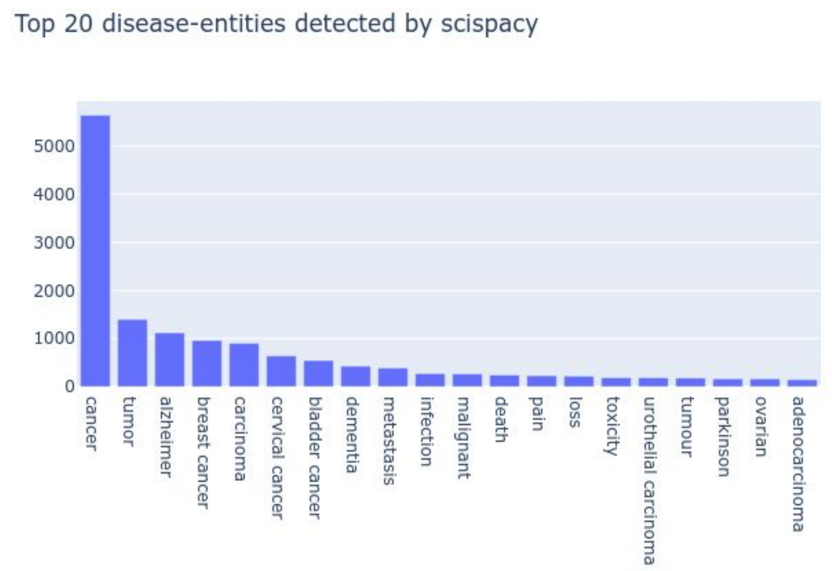


**Figure 12:** Top 20 diseases identified by ScispaCy in the training text These terms will be labeled "disease" in the second neural network approach: Scispacy_bigrams.ipynb

## RNN models

Again, in the notebooks you can find more detailed information about all the steps used and the results of each one of them. I recommend starting with notebook Emb_propio.ipynb, then notebook Scispacy_bigrams.ipynb and finally notebook Scispacy_emb.ipynb.

The previous steps employed evaluation using neural network are:

- Split of the data in train, test and validation (discussed above)
- Pre-processing of texts (discussed above)
- Transforming text into sequence: for this step, the Keras Tokenizer and its `text_to_sequences` function are used for ignoring non-vocabulary words such as "oov". The size of the vocabulary is modified to test the effect on classification.
- Padding of the sequences: the number 0 is reserved for pre-padding.

To build the pre-trained embeddings, ScispaCy vectors are extracted from the extracted vocabulary words. Therefore, the size of the vocabulary will be key for the classification.

The size of the embeddings is a maximum of 200 dimensions. All of them are used.

The elements used to optimize classification with neural networks are the number of neurons in the hidden layer, the type of hidden layer (LSTM or GRU), the number of hidden layers, the size of the vocabulary and the maximum size of the sequences (texts coded to indexes).

To reduce the number of neural network architectures to be tested, we first compare the above 3 approaches with a single-layer neural network with LSTM. The approaches with the best accuracy will be the used to test new architectures. This results are shown in the Table 9. We can see that the approaches of adding the label "bigram" and using ScispaCy embeddings have the best results.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| Simple LSTM without preprocess text | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = true | 72.80% |
| Simple LSTM preprocessed text with bigrams | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 74.40% |
| Simple LSTM preprocessed text with SpispaCy labels | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 72.27% |
| Simple LSTM<br>Spacy Embeddings | ● LSTM<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 200<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = True | 76.93% |

**Table 9:** Accuracy of the best combination of parameters for each of the different strategies used in a first exploration with a RNN simple LSTM.

In Appendix B you will find the results of all the parameter combinations tested for each of the strategies. From these results we can see that:

- the presence of a dropout layer is useful.
- the addition of metadata does not help the neural network
- the addition of labels does not improve greatly the classification accuracy
- the presence of pretrained embeddings is an useful approach.

The effect of polarity and subjectivity on classification is not as expected. If we look at the degree of these feelings in each category (Figure 13) we see that there are few differences. This may be the reason why it does not help the neural network.

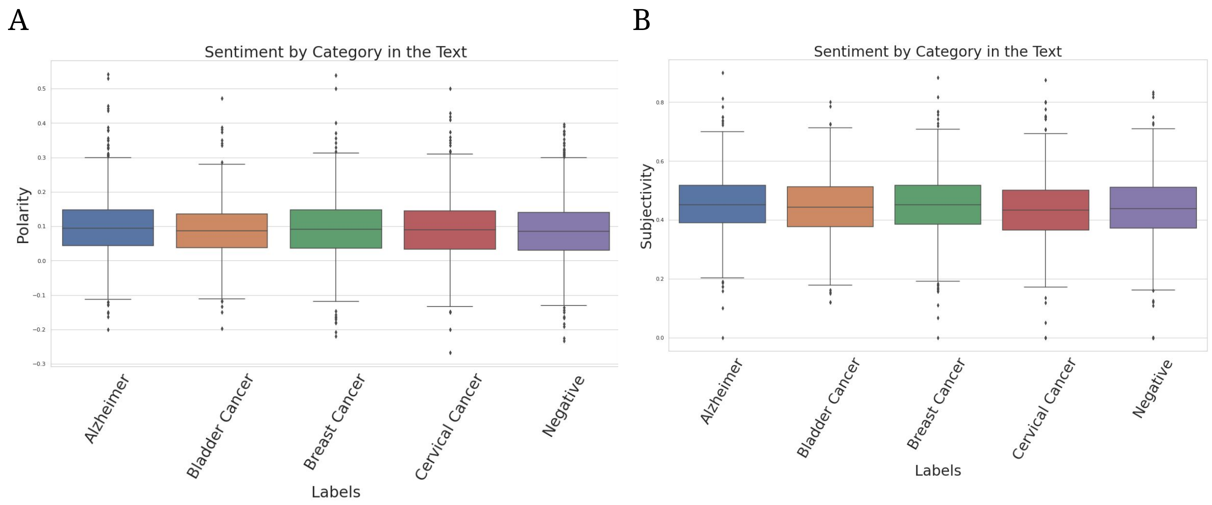


**Figure 13:** Boxplots showing how polarity (A) and subjectivity (B) are distributed in each class calculated with TextBlob.

Table 10 shows the results of new tested architectures in text preprocessing and bigram tag addition. We can see that none of the new architectures improve the classification in this strategy.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| Simple GRU preprocess text | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron GRU = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 48.80% |
| Double GRU preprocess text | ● GRU = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_GRU1 = 80<br>● neuron_GRU2 = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2 | 34.93% |

| | | |
|---|---|---|
| | ● Dropout layer = True | |
| Double LSTM preprocess text | ● LSTM = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_LSTM1 = 80<br>● neuron_LSTM2 = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 66.40% |

**Table 10:** Results of the exploring process for new architectures of the RNN preprocessing the text and adding the "bigram" label.

Table 11 shows the results of the classification with different architectures of the new network using ScipaCy embeddings. We can see how the models with GRU achieve a success rate above 90%.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| **Simple GRU Spacy Embeddings** | **● GRU**<br>**● input_length = 250**<br>**● embedding dimensions = 200**<br>**● neuron LSTM = 200**<br>**● vocabulary size = 8500**<br>**● Dropout = 0.2**<br>**● Dropout layer = False** | **91.47%** |
| Double GRU Spacy Embeddings | ● GRU = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_GRU1 = 128<br>● neuron_GRU2 = 128<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = True | 90.00% |
| Double LSTM Spacy Embeddings | ● LSTM = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_LSTM1 = 128<br>● neuron_LSTM2 = 128<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = True | 76.00% |

**Table 11:** Results of the exploring process for new architectures of the RNN using the pre-trained ScispaCy embeddings.

The 1-layer architecture with 200 neurons in the hidden GRU layer, with a vocabulary of 8500 words, 250 words for text and 200 dimensions in the ScispaCy word embedding shows the best result. New architectures were explored by resizing the vocabulary, adding normalization layers and increasing the dropout value to try to reduce overfitting and achieve better results (Table 12).

Dropout is a regularization technique for neural network models proposed by Srivastava, et al. in 2014. Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass [8]. The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data.

In the case of Batch Normalization, it is well known that normalization is a way of transforming the data in order to speed up the optimization process. Apply it to the hidden layers is tricky since the weights of the neurons change during the training. Ioffe and Szegady solved this problem by doing the normalization in batches (hence the name), such that during each batch the parameters remain fixed [9].

| **Neuronal Network** | **Parameters** | **Accuracy** |
|---|---|---|
| Simple GRU Spacy Embeddings | ● GRU = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = True | 91.47% |
| Simple GRU Spacy Embeddings | ● GRU = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 10000<br>● Dropout = 0.2<br>● Dropout layer = True | 91.60% |
| Simple GRU Spacy Embeddings | ● GRU = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 20000<br>● Dropout = 0.2<br>● Dropout layer = True | 91.47% |
| Simple GRU Spacy Embeddings | ● GRU = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 10000<br>● Dropout = 0.5<br>● Dropout layer = True | 89.33% |

| | | |
|---|---|---|
| Simple GRU Spacy Embeddings | ● GRU = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 10000<br>● Dropout = 0.5<br>● Dropout layer = True<br>● Batchnormalization = True | 88.80% |
| Simple GRU Spacy Embeddings | ● GRU = 1<br>● input_length = 400<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 15000<br>● Dropout = 0.2<br>● Dropout layer = True<br>● Batchnormalization = True | 91.20% |
| Double GRU Spacy Embeddings | ● GRU = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron GRU1 = 120<br>● neuron_GRU2 = 120<br>● vocabulary size = 10000<br>● Dropout = 0.2<br>● Dropout layer = True<br>● Batchnormalization = True | 91.33% |
| Double GRU Spacy Embeddings | ● GRU = 2<br>● input_length = 400<br>● embedding dimensions = 200<br>● neuron GRU1 = 120<br>● neuron_GRU2 = 120<br>● vocabulary size = 10000<br>● Dropout = 0.2<br>● Dropout layer = True<br>● Batchnormalization = True | 92.00% |
| Double GRU Spacy Embeddings | ● GRU = 2<br>● input_length = 500<br>● embedding dimensions = 200<br>● neuron GRU1 = 120<br>● neuron_GRU2 = 120<br>● vocabulary size = 15000<br>● Dropout = 0.2<br>● Dropout layer = True<br>● Batchnormalization = True | 92.00% |
| Double GRU Spacy Embeddings | ● **GRU = 2**<br>● **input_length = 500**<br>● **embedding dimensions = 200**<br>● **neuron GRU1 = 150**<br>● **neuron_GRU2 = 150**<br>● **vocabulary size = 15000**<br>● **Dropout = 0.2**<br>● **Dropout layer = True**<br>● **Batchnormalization = True** | **92.40%** |
| Double GRU Spacy Embeddings | ● GRU = 2<br>● input_length = 500<br>● embedding dimensions = 200<br>● neuron GRU1 = 200 | |

| | | |
|---|---|---|
| | ● neuron_GRU2 = 150<br>● vocabulary size = 20000<br>● Dropout = 0.3<br>● Dropout layer = True<br>● Batchnormalization = True | 90.80% |

**Table 12:** Results of the exploring process for new architectures of the best RNN using the pre-trained ScispaCy embeddings.

The table above shows the different combinations of parameters that were tested to obtain the best classification. Classification accuracy of 92.40% was achieved using a neural network with two GRU hidden layers of 150 neurons each with a dropout layer (p = 0.2), a vocabulary size of 150000 words and using 500 elements for each sequence (text). All dimensions of spacy embeddings (200) were used.

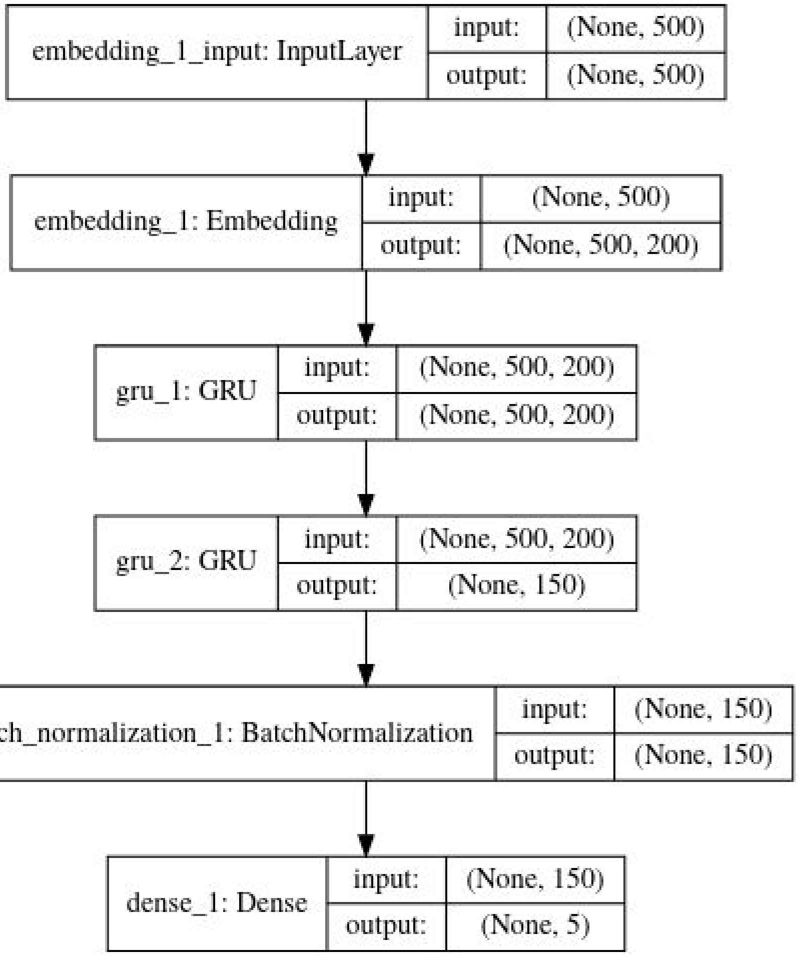


**Figure 14:** Image of architecture with better results (92.40%). Architecture: two GRU hidden layers of 150 neurons each with a dropout layer (p = 0.2), a vocabulary size of 150000 words and using 500 elements as input length. Normalization and Dropout layer are included.

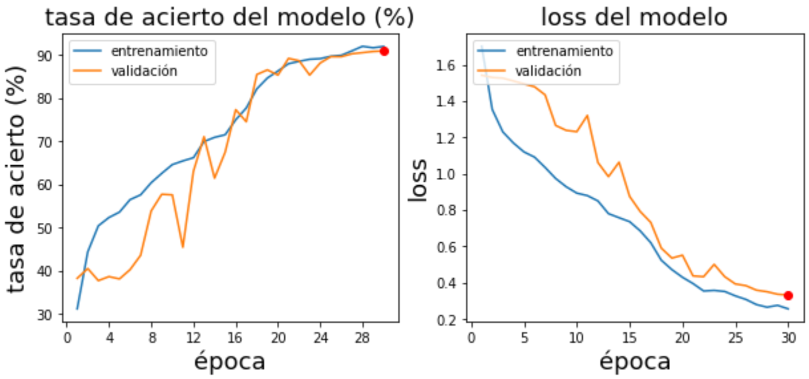
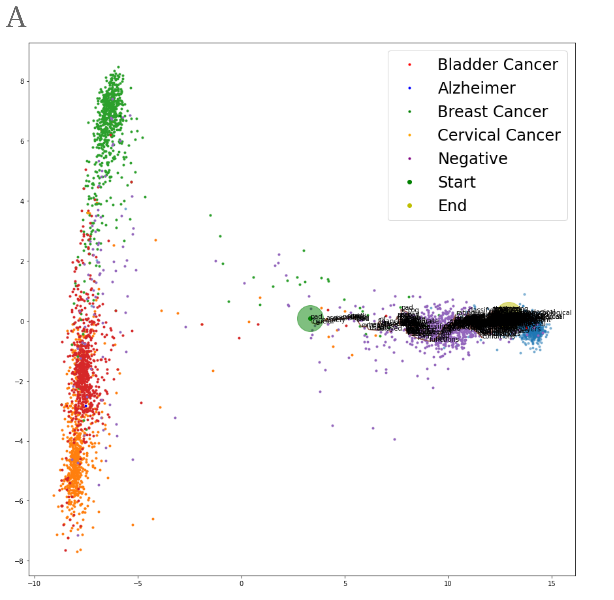


**Figure 15:** Learning curves of the model Architecture: two GRU hidden layers of 150 neurons each with a dropout layer (p = 0.2), a vocabulary size of 150000 words and using 500 elements as input length. Normalization and Dropout layer are included.

The network built is also studied by analyzing the decision-making during the classification process. Figure 16 shows the classification process of a text belonging to the class Alzheimer and Breast Cancer. During this exploration, the terms "out of vocabulary" (oov) do not help the classification but the terms considered end up moving the decision towards the correct class.



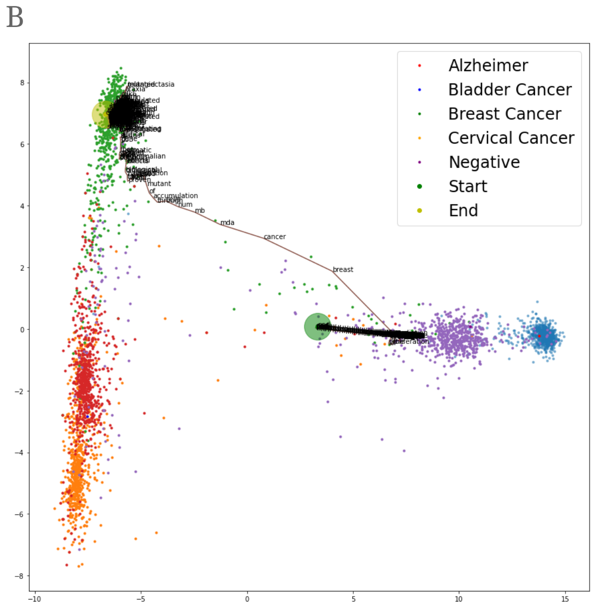


**Figure 16 A:** Visualization of decision making for the classification of a text from the Alzheimer's class.

**Figure 16 B:** Visualization of decision making for the classification of a text from the Breast Cancer class.

The reason why GRU works better than LSTM may be due to the data available. GRUs are simpler and thus easier to modify and some people have report that GRUs train faster and perform better than LSTMs on less training data [9]. Although LSTMs are more sophisticated that implies more complexity, which can lead to a bad classification when few data are available. It cannot be concluded which is better, but both must be tested. In our case GRUs take less time to train and are more efficient.

## Conclusions

To conclude the project, the best results of the different approaches are collected.

| Model | Parameters | Accuracy |
|---|---|---|
| TF-IDF<br>Logistic Regression Lasso Regularization (L1) | ● TF-IDF<br>● Polaridad = True<br>● Subjectivity = True<br>● Vocabulary size = 8500<br>● Max_df = 0.7<br>● Min_df = 0<br>● n_grams = 1, 2<br>● Lemmatizer: NLTK<br>● Stop Words: Long | 92.13% |
| Word2Vec<br>Logistic Regression Lasso Regularization (L1) | ● Word2Vec<br>● Embedding size = 128<br>● Word size = 10<br>● Min count = 10 | 85.87% |
| Double GRU Spacy Embeddings | ● GRU = 2<br>● input_length = 500<br>● embedding dimensions = 200<br>● neuron GRU1 = 150<br>● neuron_GRU2 = 150<br>● vocabulary size = 15000<br>● Dropout = 0.2<br>● Dropout layer = True<br>● Batchnormalization = True | 92.40% |

**Table 13:** Summary of the 3 approaches to the classification of biomedical texts: TF-IDF, Word2Vec and RNN.

The neural network, closely followed by TF-IDF (Term frequency - Inverse document frequency) achieves the best result. We can see the potential of the neural networks. We have studied the difficulty of training the networks due to the large number of parameters available for optimization. Moreover, we must not forget that data largely determine the learning of the models. In this case the different categories presented the same proportion, which facilitates the use of traditional Machine Learning models. However, neural networks are a tool with great potential to deal with different problems, and they can become more complex.

To sum up, we have managed to generate classification models with over 85% accuracy in all the strategies followed, achieving a maximum of 92.40% with recurrent neural networks.

## Bibliography

1. Li, S. (2018). Multi-Class Text Classification with Scikit-Learn. Retrieved 9 May 2020, from https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f
2. Fernández Zafra, M. (2019). Text Classification in Python. Retrieved 9 May 2020, from https://towardsdatascience.com/text-classification-in-python-dd95d264c802
3. Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, & Brown. (2019). Text Classification Algorithms: A Survey. *Information*, *10*(4), 150. doi: 10.3390/info10040150
4. Netrapalli, P. (2019). Stochastic Gradient Descent and Its Variants in Machine Learning. *Journal Of The Indian Institute Of Science*, *99*(2), 201-213. doi: 10.1007/s41745-019-0098-4
5. Ganesan, K. (2018). Gensim Word2Vec Tutorial - Full Working Example | Kavita Ganesan. Retrieved 11 May 2020, from https://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/
6. Olivares, A. (2019). ArmandDS/news_category. Retrieved 11 May 2020, from https://github.com/ArmandDS/news_category/blob/master/News_Analysis_AO.ipynb

7. (2020) Quora “How are recurrent neural networks different from convolutional neural networks?”. Retrieved 11 May 2020, from https://www.quora.com/How-are-recurrent-neural-networks-different-from-convolutional-neural-networks

8. Brownlee, J. (2019). Dropout Regularization in Deep Learning Models With Keras.

Retrieved 12 May 2020, from https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

9. LSTM?, W., Sonawane, S., Jaiswal, A., Varsha, H., & Khetan, V. (2020). When to use GRU over LSTM?. Retrieved 12 May 2020, from https://datascience.stackexchange.com/questions/14581/when-to-use-gru-over-lstm

# Appendix A

Additional results of the Machine Learning approach with Sklearn.

## Lemmatizer and stop words selection for TF-IDF approach

| Stop words | Lemmatizer | N-grams | Model | Accuracy |
|---|---|---|---|---|
| Long | NLTK | 1 | Naive Bayes | 73.8 |
| | | | Log. Reg | 80.33 |
| | | | **Log. Reg. L1** | **82.46** |
| | | | Log. Reg. L2 | 80.13 |
| | | | KNN | 67.47 |
| | | | Random Forest | 80.73 |
| | | | SVC | 78.7 |
| | | | SGD | 77.27 |
| NLTK | NLTK | 1 | Naive Bayes | 73.13 |
| | | | Log. Reg | 80.53 |
| | | | **Log. Reg. L1** | **82.47** |
| | | | Log. Reg. L2 | 80.2 |
| | | | KNN | 63.13 |
| | | | Random Forest | 81.93 |
| | | | SVC | 78.93 |
| | | | SGD | 76.93 |
| Clinical | NLTK | 1 | Naive Bayes | 74.0 |
| | | | Log. Reg | 80.4 |
| | | | **Log. Reg. L1** | **82.4** |
| | | | Log. Reg. L2 | 80.73 |
| | | | KNN | 67.73 |
| | | | Random Forest | 80.93 |
| | | | SVC | 79.2 |
| | | | SGD | 76.93 |
| Spacy | NLTK | 1 | Naive Bayes | 73.13 |
| | | | Log. Reg | 80.53 |
| | | | **Log. Reg. L1** | **82.46** |
| | | | Log. Reg. L2 | 80.2 |
| | | | KNN | 68.13 |

| | | | | |
|---|---|---|---|---|
| | | | Random Forest | 81.93 |
| | | | SVC | 78.93 |
| | | | SGD | 76.93 |
| Spacy | Spacy | 1 | Naive Bayes | 73.07 |
| | | | Log. Reg | 80.33 |
| | | | **Log. Reg. L1** | **83.27** |
| | | | Log. Reg. L2 | 80.27 |
| | | | KNN | 67.8 |
| | | | Random Forest | 81.13 |
| | | | SVC | 78.0 |
| | | | SGD | 76.33 |

| Stop words | Lemmatizer | N-grams | Model | Accuracy |
|---|---|---|---|---|
| Long | NLTK | 1-2 | Naive Bayes | 78.86 |
| | | | Log. Reg | 84.8 |
| | | | Log. Reg. L1 | 87.33 |
| | | | Log. Reg. L2 | 84.13 |
| | | | KNN | 74.07 |
| | | | **Random Forest** | **87.8** |
| | | | SVC | 84.2 |
| | | | SGD | 82.93 |
| NLTK | NLTK | 1-2 | Naive Bayes | 78.67 |
| | | | Log. Reg | 84.33 |
| | | | **Log. Reg. L1** | **87.07** |
| | | | Log. Reg. L2 | 84.6 |
| | | | KNN | 74.47 |
| | | | Random Forest | 86.47 |
| | | | SVC | 83.8 |
| | | | SGD | 82.8 |
| Clinical | NLTK | 1-2 | Naive Bayes | 78.53 |
| | | | Log. Reg | 84.93 |
| | | | Log. Reg. L1 | 87.13 |
| | | | Log. Reg. L2 | 85.13 |
| | | | KNN | 73.8 |
| | | | **Random Forest** | **87.4** |
| | | | SVC | 84.27 |

| | | | | |
|---|---|---|---|---|
| | | | SGD | 83.13 |
| Spacy | NLTK | 1-2 | Naive Bayes | 78.67 |
| | | | Log. Reg | 84.33 |
| | | | **Log. Reg. L1** | **87.07** |
| | | | Log. Reg. L2 | 84.67 |
| | | | KNN | 74.47 |
| | | | Random Forest | 84.47 |
| | | | SVC | 83.8 |
| | | | SGD | 82.87 |
| Spacy | Spacy | 1-2 | Naive Bayes | 78.8 |
| | | | Log. Reg | 84.73 |
| | | | **Log. Reg. L1** | **87.6** |
| | | | Log. Reg. L2 | 84.27 |
| | | | KNN | 74.47 |
| | | | Random Forest | 87.53 |
| | | | SVC | 84.2 |
| | | | SGD | 83.47 |

| Stop words | Lemmatizer | N-grams | Model | Accuracy |
|---|---|---|---|---|
| Long | NLTK | 1-3 | Naive Bayes | 79.0 |
| | | | Log. Reg | 85.0 |
| | | | **Log. Reg. L1** | **87.2** |
| | | | Log. Reg. L2 | 84.47 |
| | | | KNN | 74.53 |
| | | | Random Forest | 86.8 |
| | | | SVC | 83.63 |
| | | | SGD | 82.77 |
| NLTK | NLTK | 1-3 | Naive Bayes | 78.47 |
| | | | Log. Reg | 84.8 |
| | | | Log. Reg. L1 | 87.13 |
| | | | Log. Reg. L2 | 84.73 |
| | | | KNN | 74.53 |
| | | | **Random Forest** | **87.27** |
| | | | SVC | 83.8 |
| | | | SGD | 83.2 |
| Clinical | NLTK | 1-3 | Naive Bayes | 79.07 |

| | | | | |
|---|---|---|---|---|
| | | | Log. Reg | 85.07 |
| | | | **Log. Reg. L1** | **87.27** |
| | | | Log. Reg. L2 | 85.33 |
| | | | KNN | 73.6 |
| | | | Random Forest | 87.0 |
| | | | SVC | 84.33 |
| | | | SGD | 83.53 |
| Spacy | NLTK | 1-3 | Naive Bayes | 78.47 |
| | | | Log. Reg | 84.8 |
| | | | Log. Reg. L1 | 87.13 |
| | | | Log. Reg. L2 | 84.73 |
| | | | KNN | 74.53 |
| | | | **Random Forest** | **87.27** |
| | | | SVC | 83.8 |
| | | | SGD | 82.47 |
| Spacy | Spacy | 1-3 | Naive Bayes | 78.87 |
| | | | Log. Reg | 84.93 |
| | | | **Log. Reg. L1** | **87.4** |
| | | | Log. Reg. L2 | 85.2 |
| | | | KNN | 74.73 |
| | | | Random Forest | 87.27 |
| | | | SVC | 84.13 |
| | | | SGD | 83.0 |

**Tabla A.1:** Classification results in exploring the optimal stop words list and lemmatizer for the pre-processing step TF-IDF

## Vocabulary size TF-IDF selection

| Vocabulary size | Model | Accuracy |
|---|---|---|
| 10 | Naive Bayes | 65.4 |
| | Log. Reg | 73.0 |
| | Log. Reg. L1 | 73.0 |
| | Log. Reg. L2 | 72.8 |
| | KNN | 70.53 |
| | **Random Forest** | **73.07** |
| | SVC | 72.67 |
| | SGD | 70.53 |
| 500 | Naive Bayes | 77.07 |
| | Log. Reg | 83.8 |
| | Log. Reg. L1 | 86.47 |
| | Log. Reg. L2 | 83.47 |
| | KNN | 73.47 |
| | **Random Forest** | **86.93** |
| | SVC | 83.27 |
| | SGD | 84.2 |
| 1000 | Naive Bayes | 78.6 |
| | Log. Reg | 83.87 |
| | **Log. Reg. L1** | **87.33** |
| | Log. Reg. L2 | 84.2 |
| | KNN | 74.67 |
| | Random Forest | 80.53 |
| | SVC | 83.53 |
| | SGD | 84.2 |
| 5000 | Naive Bayes | 78.86 |
| | Log. Reg | 84.8 |
| | Log. Reg. L1 | 87.33 |
| | Log. Reg. L2 | 84.13 |
| | KNN | 74.07 |
| | **Random Forest** | **87.8** |
| | SVC | 84.2 |
| | SGD | 82.93 |
| 10000 | Naive Bayes | 78.33 |

| | | |
|---|---|---|
| | Log. Reg | 84.73 |
| | Log. Reg. L1 | 86.27 |
| | Log. Reg. L2 | 84.4 |
| | KNN | 73.73 |
| | **Random Forest** | **87.27** |
| | SVC | 84.2 |
| | SGD | 83.07 |
| 50,000 | Naive Bayes | 77.47 |
| | Log. Reg | 84.53 |
| | Log. Reg. L1 | 86.2 |
| | Log. Reg. L2 | 84.2 |
| | KNN | 78.3 |
| | **Random Forest** | **86.8** |
| | SVC | 84.73 |
| | SGD | 83.4 |

**Tabla A.2:** Classification results in exploring the optimal vocabulary size for TF-IDF

## Confusion matrix best TF-IDF model

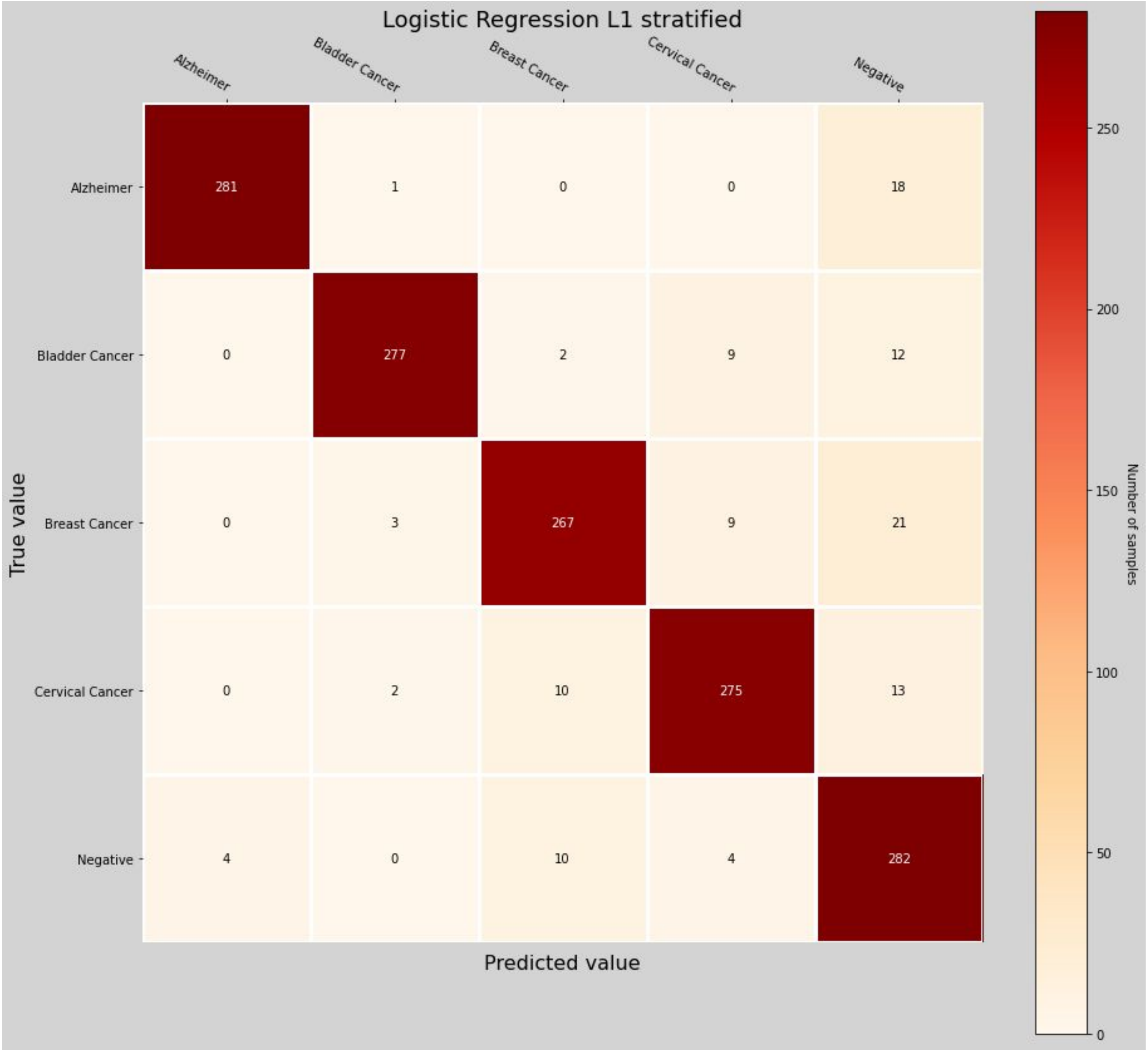


**Figure A.1:** Confusion matrix of the best TF-IDF procedure using NLTK lemmatizer, Long Stopwords, unigrams and bigrams, 8500 features, max_df = 0.7 and min_df = 0, and polarity and subjectivity . Best model: Logistic Regression with L1 regularization,

## Adaboost in TF-IDF approach

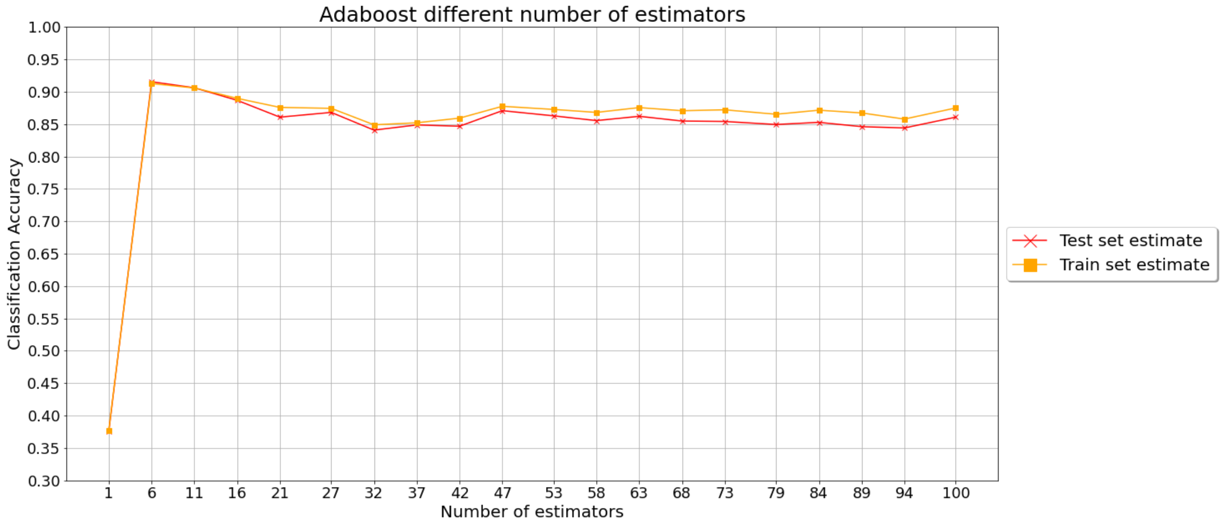


**Figure A.2:** Adaboost training process optimizing the number of classifiers used with the best combination of TF-IDF parameters. The best training value is obtained with 6 classifiers. The training error for this value is 91.25 % and in test 91.53 %.

## Metrics for the best Word2Vec model

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| Alzheimer | 0.94 | 0.92 | 0.93 | 304 |
| Bladder Cancer | 0.88 | 0.87 | 0.88 | 323 |
| Breast Cancer | 0.83 | 0.85 | 0.84 | 278 |
| Cervical Cancer | 0.89 | 0.83 | 0.86 | 295 |
| Negative | 0.75 | 0.81 | 0.78 | 300 |
| **accuracy** | -- | -- | 0.86 | 1500 |
| **macro avg** | 0.86 | 0.85 | 0.86 | 1500 |
| **weighted avg** | 0.86 | 0.86 | 0.86 | 1500 |

**Table A.2:** Metrics for the best model of Word2Vec using  IDF values, window = 10, min_count = 10, embedding size = 128.

# Appendix B

Additional results of the RNNs.

## Pre-processing effect on the bigram detection

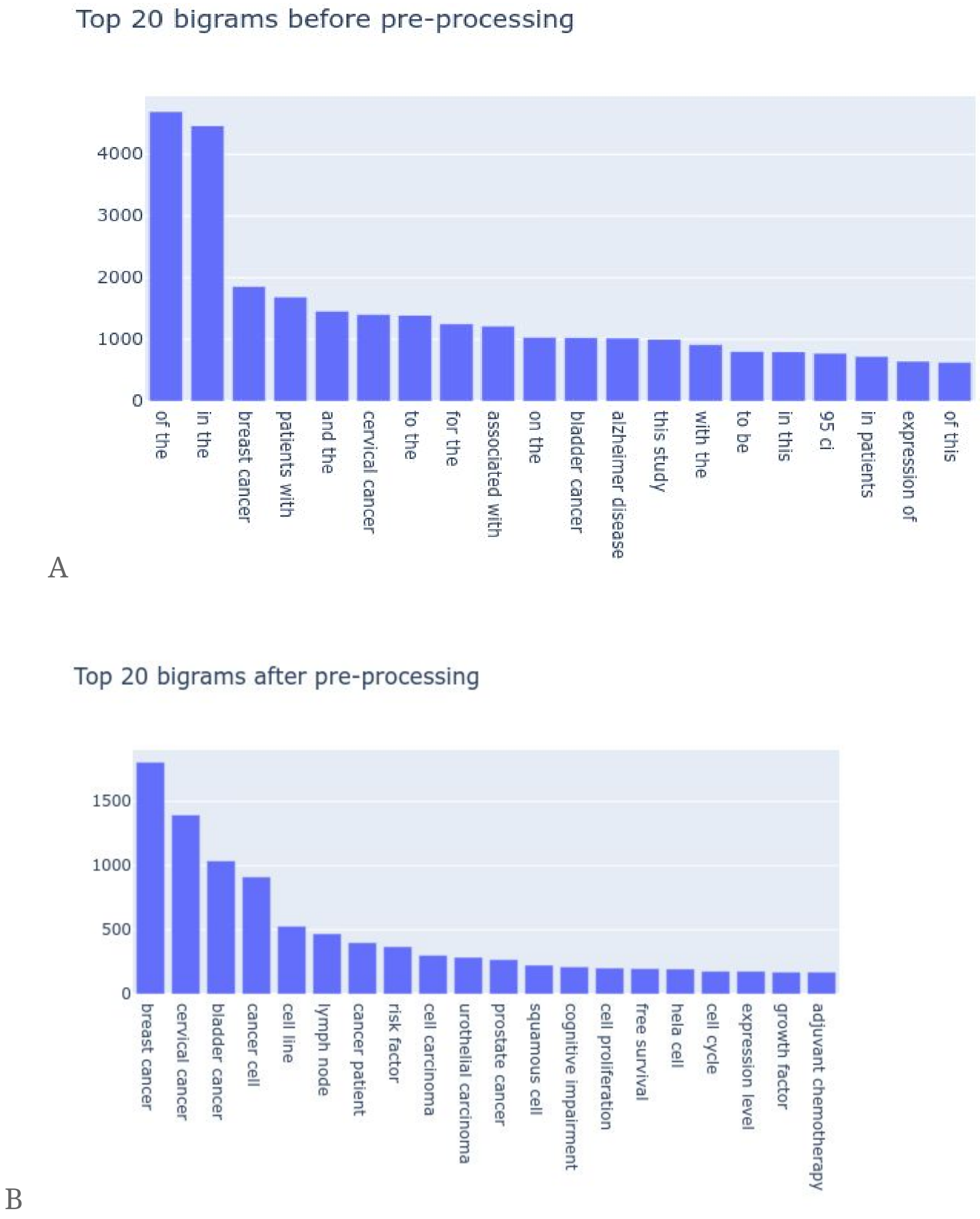


**Figure B.1:** Bar plot showing how the top 20 identified bigrams in the training corpus change before and after pre-processing for the addition of the bigramn tag in the neural networks

## Results of the exploring process of the RNN using raw data.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| Simple LSTM without preprocess text | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● dropout layer = False | 68.00% |
| Simple LSTM without preprocess text | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● dropout layer = True | 70.67% |
| Simple LSTM without preprocess text | ● LSTM =1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● dropout layer = True | 70.67% |
| Simple LSTM without preprocess text | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 80<br>● vocabulary size = 6000<br>● Deleting stop words = False<br>● Dropout = 0.2<br>● Dropout layer = true | 72.80% |

**Table B.1:** Results of the exploring process of the RNN using raw data.

## Results of the exploring process of the RNN preprocessing text and adding “bigram” labels.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| Simple LSTM preprocessed text | ● LSTM<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 6000<br>● Deleting stop words = False<br>● Dropout = 0.2<br>● dropout layer = False | 69.60% |
| Simple LSTM  preprocessed text | ● LSTM<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● dropout layer = True | 69.20% |
| Simple LSTM  preprocessed text | ● LSTM<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● dropout layer = True | 68.80% |
| Simple LSTM preprocessed text | ● LSTM<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 80<br>● vocabulary size = 6000<br>● Deleting stop words = False<br>● Dropout = 0.2<br>● Dropout layer = true | 74.40% |

**Table B.2:** Results of the exploring process of the RNN preprocessing text and adding “bigram” labels.

## Results of the exploring process of the RNN preprocessing text and adding labels from spaCy entity recognition.

| Neural Network | Parameters | Accuracy |
|---|---|---|
| Simple LSTM preprocess text with SpispaCy labels | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● No dropout layer | 69.20% |
| Simple LSTM preprocess text with SpispaCy labels | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 72.27% |
| Simple LSTM preprocess text with SpispaCy labels | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = False | 70.40% |
| Simple LSTM preprocess text with SpispaCy labels | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 71.07% |
| Simple LSTM preprocess text with SpispaCy labels | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Spatial dropout layer = True | 71.87% |
| Simple LSTM preprocess text with SpispaCy labels | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Spatial dropout layer = True | 71.47% |
| Simple LSTM preprocess text with SpispaCy labels | ● LSTM = 1<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 40<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = True | 70.93% |

| | | |
|---|---|---|
| Simple LSTM preprocess text with SpispaCy labels | • LSTM = 1<br>• input_length = 250<br>• embedding dimensions = 200<br>• neuron LSTM = 80<br>• vocabulary size = 8500<br>• Dropout = 0.2<br>• dropout layer = False | 67.20% |

**Table B.3:** Results of the exploring process of the RNN preprocessing text and adding labels from spaCy entity recognition.

## Results of the exploring process of the RNN using pre-trained ScispaCy embeddings.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| Simple LSTM<br>Spacy Embeddings | • LSTM<br>• input_length = 60<br>• embedding dimensions = 200<br>• neuron LSTM = 200<br>• vocabulary size = 6000<br>• Dropout = 0.2<br>• Dropout layer = False | 73.07% |
| Simple LSTM<br>Spacy Embeddings | • LSTM<br>• input_length = 60<br>• embedding dimensions = 200<br>• neuron LSTM = 200<br>• vocabulary size = 6000<br>• Dropout = 0.2<br>• Dropout layer = True | 73.87% |
| Simple LSTM<br>Spacy Embeddings | • LSTM<br>• input_length = 250<br>• embedding dimensions = 200<br>• neuron LSTM = 200<br>• vocabulary size = 8500<br>• Dropout = 0.2 | 73.47% |
| Simple LSTM<br>Spacy Embeddings | • LSTM<br>• input_length = 250<br>• embedding dimensions = 200<br>• neuron LSTM = 400<br>• vocabulary size = 8500<br>• Dropout = 0.2<br>• Dropout layer = True | 68.53% |
| Simple LSTM<br>Spacy Embeddings | • LSTM<br>• input_length = 250<br>• embedding dimensions = 200<br>• neuron LSTM = 200<br>• vocabulary size = 8500<br>• Dropout = 0.2<br>• Dropout layer = True | 76.93% |
| Simple LSTM<br>Spacy Embeddings | • LSTM<br>• input_length = 250<br>• embedding dimensions = 200<br>• neuron LSTM = 200<br>• vocabulary size = 8500<br>• Dropout = 0.2<br>• Dropout layer = False | 73.47% |

**Table B.4:** Results of the exploring process of the RNN using pre-trained ScispaCy embeddings.

## Results of the exploring process of the RNN using pre-trained ScispaCy embeddings and adding meta-information about sentiment and polarity with TextBlob.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| Simple LSTM Spacy Embeddings TextBlob Sentiment | ● LSTM = 1<br>● input_length = 60<br>● neuron LSTM = 200<br>● vocabulary size = 6000 | 64.93% |
| Simple LSTM Spacy Embeddings TextBlob Sentiment | ● LSTM = 1<br>● input_length = 60<br>● neuron LSTM = 200<br>● vocabulary size = 85000 | 64.13% |
| Simple LSTM Spacy Embeddings TextBlob Sentiment | ● LSTM<br>● input_length = 60<br>● neuron LSTM = 200<br>● vocabulary size = 6000<br>● input_length = 60 | 41.47% |
| Simple LSTM Spacy Embeddings TextBlob Sentiment | ● LSTM<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 400<br>● vocabulary size = 8500<br>● Dropout = 0.2 | 72.27% |
| Simple LSTM Spacy Embeddings TextBlob Sentiment | ● LSTM<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 200<br>● vocabulary size = 8500<br>● Deleting stop words = False<br>● Dropout = 0.2 | 72.40% |
| Simple GRU Spacy Embeddings TextBlob Sentiment | ● GRU<br>● input_length = 60<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 6000<br>● Deleting stop words = True<br>● Dropout = 0.2 | 36.53% |
| Simple GRU Spacy Embeddings TextBlob Sentiment | ● GRU<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 400<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● dropout layer = False | 38.67% |
| Simple GRU Spacy Embeddings TextBlob Sentiment | ● GRU<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 200<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● dropout layer = False | 37.20% |

**Table B.5:** Results of the exploring process of the RNN using pre-trained ScispaCy embeddings and adding meta-information about sentiment and polarity with TextBlob.

## Results of the exploring process of the RNN with GRU hidden layer using pre-trained ScispaCy embeddings

| Neuronal Network | Parameters | |
|---|---|---|
| Simple GRU<br>Spacy Embeddings | ● GRU<br>● input_length = 60<br>● embedding dimensions = 200<br>● neuron LSTM = 200<br>● vocabulary size = 6000<br>● Deleting stop words = True<br>● Dropout = 0.2<br>● Dropout layer = True | 74.00% |
| Simple GRU<br>Spacy Embeddings | ● GRU<br>● input_length = 60<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Trainable = False<br>● Dropout layer = False | 63.07% |
| Simple GRU<br>Spacy Embeddings | ● GRU<br>● input_length = 60<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Trainable = False<br>● Dropout layer = False | 73.73% |
| Simple GRU<br>Spacy Embeddings | ● GRU<br>● input_length = 60<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Trainable = False<br>● Dropout layer = True | 74.27% |
| Simple GRU<br>Spacy Embeddings | ● GRU<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 200<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = False | 91.47% |
| Simple GRU<br>Spacy Embeddings | ● GRU<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 400<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = False | 91.33% |
| Simple GRU<br>Spacy Embeddings | ● GRU<br>● input_length = 250 | 91.20% |

| | | |
|---|---|---|
| | ● embedding dimensions = 200<br>● neuron LSTM = 400<br>● vocabulary size = 8500<br>● Deleting stop words = False<br>● Dropout = 0.2<br>● Dropout layer = 0.2 | |
| Simple GRU<br>Spacy Embeddings | ● GRU<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron LSTM = 400<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = True | 74.53% |

**Table B.6:** Results of the exploring process of the RNN with GRU hidden layer using pre-trained ScispaCy embeddings

## Results of the exploring process of new architectures of the RNN preprocessing the text and adding "bigram" labels.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| Double GRU preprocess text | ● GRU = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_GRU1 = 40<br>● neuron_GRU2 = 40<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 30.53% |
| Double GRU preprocess text | ● GRU = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_GRU1 = 80<br>● neuron_GRU2 = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 34.93% |
| Double LSTM preprocess text | ● LSTM = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_LSTM1 = 40<br>● neuron_LSTM2 = 40<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 42.27% |
| Double LSTM preprocess text | ● LSTM = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_LSTM1 = 80<br>● neuron_LSTM2 = 80<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 66.40% |
| Double LSTM preprocess text | ● LSTM = 2<br>● input_length = 250 | 54.00% |

| | | |
|---|---|---|
| | ● embedding dimensions = 200<br>● neuron_LSTM1 = 80<br>● neuron_LSTM2 = 80<br>● vocabulary size = 10000<br>● Dropout = 0.2<br>● Dropout layer = True | |

**Table B.7:** Results of the exploring process of new architectures of the RNN preprocessing the text and adding "bigram" labels.

### Results of the exploring process of new architectures of the RNN using ScispaCy embeddings.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| Double GRU Spacy Embeddings | ● GRU = 2<br>● input_length = 60<br>● embedding dimensions = 200<br>● neuron_GRU1 = 128<br>● neuron_GRU2 = 128<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 73.07% |
| Double GRU Spacy Embeddings | ● GRU = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_GRU1 = 128<br>● neuron_GRU2 = 128<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = True | 90.00% |
| Double LSTM Spacy Embeddings | ● LSTM = 2<br>● input_length = 60<br>● embedding dimensions = 200<br>● neuron_LSTM1 = 128<br>● neuron_LSTM2 = 128<br>● vocabulary size = 6000<br>● Dropout = 0.2<br>● Dropout layer = True | 72.93% |
| Double LSTM  Spacy Embeddings | ● LSTM = 2<br>● input_length = 250<br>● embedding dimensions = 200<br>● neuron_LSTM1 = 128<br>● neuron_LSTM2 = 128<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = True | 76.00% |

**Table B.8:** Results of the exploring process of new architectures of the RNN using ScispaCy embeddings.

## Results of the exploring process of new architectures of the RNN using BIOBERT embeddings.

| Neuronal Network | Parameters | Accuracy |
|---|---|---|
| Simple LSTM BIOBERT embedding | ● LSTM = 1<br>● input_length = 500<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 15000<br>● Dropout = 0.2<br>● Dropout layer = True | 56.93% |
| Simple GRU BIOBERT Embeddings | ● GRU = 1<br>● input_length = 200<br>● embedding dimensions = 300<br>● neuron GRU = 200<br>● vocabulary size = 15000<br>● Dropout = 0.2<br>● Dropout layer = True | 75.73% |
| Bi GRU BIOBERT Embeddings | ● GRU = 2<br>● input_length = 200<br>● embedding dimensions = 300<br>● neuron_GRU1 = 200<br>● neuron_GRU2 = 150<br>● vocabulary size = 15000<br>● Dropout = 0.2<br>● Dropout layer = True | 77.73% |
| Simple GRU BIOBERT Embeddings | ● GRU = 1<br>● input_length = 200<br>● embedding dimensions = 200<br>● neuron GRU = 200<br>● vocabulary size = 8500<br>● Dropout = 0.2<br>● Dropout layer = True | 86.53% |

**Table B.9:** Results of the exploring process of new architectures of the RNN using BIOBERT embeddings.