Classification and Summarization of Medical Abstracts

Paolo Caggiano - Davide Giardini

Contents

Abstract

Biomedical and life sciences literature is one of the biggest fields in term of number of academic publications. Its exponentially increasing volume and interdisciplinary nature makes information retrieval tools essential in this field. In this study, we analyze multiple combinations of preprocessing, feature extraction and feature selection techniques to tackle the problem of Multi-Label Multi-Class (MLMC) classification of medical abstracts. Secondly, we use two different extractive text summarization techniques in order to create an overview of each document. In both cases, we search for the best approach to address the two tasks by evaluating the results through appropriate measures.

1 Introduction

The historical extent of healthcare and its constant evolution made medicine one of the field with the greatest extent of academic literature. This large amount of medical text represents an important challenge for those who seek to retrieve information from this type of data, such medical doctors and researchers. Often, the screening phase in these jobs requires a significant effort, where experts evaluate thousand of articles to pinpoint relevant information. Search engines specifically tailored for academic publications, like Google Scholar, have already proven to be excellent methods for navigating this large amount of text. Furthermore, search engines developed precisely for medical publications, like PubMed[1], have also already been developed and are widely used. Though, the fact that biomedical literature often refers to a large number of subdomains, still poses great challenges for automatic classification systems. Furthermore, since it is crucial for physicians and researchers in medicine and biology to have quick and efficient access to up-to-date information according to their interests and needs, methods must be found in order to enable users to quickly assimilate and determine the content of a document [1].

The rise of text classification and summarization tools has emerged as a possible solution to navigate through this massive amount of features. This paper delves into the integration of these two text mining methods within the framework of analyzing medical documents.

After a brief introduction to the dataset, in Section 3 we tackle the problem of Multi-Class Multi-Label (MCML) classification of medical abstracts related to 5 different classes of patient conditions. While doing this, we evaluate, with the use of suitable measure, which combination of Pre-Processing, Feature Extraction and Feature Selection techniques helps to achieve the best results. More precisely, we investigate:

[1]https://pubmed.ncbi.nlm.nih.gov/

- **Three different Pre-Processing techniques**.
  i.e., Basic Preprocessing, Stopwords removal and Lemmatization
- **Four different Feature Extraction techniques**.
  i.e., BoW, TF, tf-idf and Word Embeddings
- **Two different Feature Selection techniques**.
  i.e., a naive removal of rare words and Principal Component Analysis
- **Four different classificators**.
  i.e., Naive Bayes, Decision Trees, Random Forest and Support Vector Machines

Then, we tackle the problem of text summarization with two different techniques: one graph-based approach and one based on Latent Semantic Analysis. We evaluate the resulting summaries in two ways: comparing them to the title of the document, via the "Rouge" metrics, and by assessing their abilities in performing the previous task of text classification.

Finally, in section 5, we conclude our report.

2 Dataset

To perform our evaluation, we utilized the dataset coming from a study devoted in evaluating unsupervised text classification methods[2]. The dataset, available on Github[2], is composed of 9445 observations describing 5 different classes of patient conditions :

1. Neoplasms
2. Digestive system diseases
3. Nervous system diseases
4. Cardiovascular diseases
5. General pathological conditions

Each observation of this dataset is composed of the combination of the paper title and abstract. This will later prove to be a challenge for the summarization task, in which we will want to separate the two in order to perform the summarization on the abstract and the evaluation on the title.

The main peculiarity of this dataset is that each document can be part of one or many classes, making this a Multi-Class Multi-Label classification problem.

[2]https://github.com/sebischair/medical-abstracts-tc-corpus

3 Classification

Document Classification is a fundamental learning problem of many information management and retrieval tasks. The problem of MCML classification may be solved as n independent binary classification problems, where n is the number of total classes, in our case five.

In this section, we are initially going to introduce all Pre-Processing, Feature Extraction, Feature Selection, Classificators and Performance measures we used. Then, in the "Result" subsection, we are firstly going to investigate on the combinations of Pre-Processing and Feature Extraction, and then apply Feature Selection on the best one.

3.1 Text Pre-Processing

We are going to evaluate three different Pre-Processing techniques:

1. **Basic Preprocessing**.
   Refers to removing punctuation and other non-alphanumeric characters, setting all the words to lower case and tokenizing them.
2. **Stop-Words removal**.
   Refers to removing words that occur very frequently and have little semantic content (a, the, to, ..), and that are therefore useless to address the problem of classification.
3. **Lemmatization**.
   Refers to the reduction of each token to its lemma, i.e., the dictionary form of a word.

3.2 Feature Extraction

Feature Extraction is an essential step in working with documents, since we have to transform the data into a format with which a computer can deal with. In our project we use four different methods: BoW, TF, tf-idf and Word embeddings.
"Bag of Words" (BoW) refers to simply assigning the value 1 to a word if it is present in the document, 0 otherwise.
The "Term-Frequency" (TF) weighting system is based, as the name suggests, on the number of times that a term t occurs in a document d.
The tf-idf weight of a term t in a document d, instead, is the product of its "TF" weight and its "idf" weight. The "Inverse Document Frequency" (idf) of a term t is computed as $log\left(\frac{N}{df_t}\right)$, where the "Document Frequency" (df_t) is the number of documents that contain

the term t. In other words, the tf-idf weight of a term t in a document d increases with the number of occurrences of the term within the document, and with the rarity of the term in the collection. For example, medical texts often contain common terms that appear frequently but may not carry much discriminative power (e.g., common medical terms like "patient" or "treatment"). The tf-idf procedure guarantees that such terms have a small weight, contrary to relatively rare but essential terms for distinguishing between classes.
Lastly, word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. In word embeddings, individual words are represented as real-valued vectors in a predefined vector space: each word is mapped to one vector and the vector values are learned in a way that resembles a neural network. In this paper, we are going to compute the vector representations of each word in two ways: training a word2vec model over our dataset and utilizing pre-trained word embeddings. The second approach, more precisely, refers to the utilization of embeddings trained in a previous work [3], and stored on the researcher's website[3]. These word vectors were induced from a combination of texts from PubMed and PMC[4]. The main idea is that the training of the vectors over a larger amount of documents of the same type will result in a better representation of the words, and therefore a better classification.

3.3 Feature Selection

Feature selection is one of the most frequent and important techniques in data preprocessing, and has become an indispensable component of the machine learning process. It refers to the process of detecting relevant features and removing irrelevant, redundant, or noisy data. This process speeds up data mining algorithms, improves predictive accuracy, and increases comprehensibility. In our project we use two techniques: a "naive" approach and Principal Component Analysis. In the first case, we simply discard the words that are rare, in order to reduce the number of features. We define as "rare" the words that occur only once in the entire collection of documents. Of our vocabulary of 26447 tokens (not including stopwords), 8228 of them is mentioned only once. This means that 8228 variables in the BoW, TF and tf-Idf matrices are composed only of zeros, except for only one element. In the second case, we use the Principal Component Analysis to reduce the dimensionality of the data. The goal of PCA is to find new, simpler variables that are not directly observed, but are linear functions of those in the original dataset. These new dimensions, called principal components (PCs), are identified so as to explain most of the variance-covariance structure of the observed variables, in an effort to retain the maximum amount of information of the original data. In other words, given m as the document dimension and k as number of components, we aim to select the k components, where $k < m$, that explain most of the variability in the data.
Feature Selection techniques are only going to be applied on BoW, TF, and tf-idf matrices. In fact, since representation matrices coming from word embeddings will already have a much lower dimensionality, we are not interested in applying these types of techniques on them. More precisely, as already mentioned, they are going to be applied only on the best combination of Pre-Processing and Feature Extraction.

3.4 Classification Algorithms

In our research we use four classification algorithms.

- **Decision Tree**
  A decision tree is a non-parametric supervised learning algorithm for classification and regression tasks. It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes.
- **Random Forest**
  Random forest consists of a large number of individual decision trees that operate together. Each individual tree in the random forest outputs a class prediction. The class with the most votes becomes our model's prediction.
- **Naive Bayes**
  Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
- **Support Vector Machine**
  SVM classifies data by finding the best hyperplane that separates all data points of

[3] bio.nlplab.org
[4] https://www.ncbi.nlm.nih.gov/pmc/

one class from those of the other class, where "best" is defined in terms of largest margin between the two classes.

Even though most of the studies on the matter found that the SVM algorithm outperforms many other algorithms, followed by NB, RF, DT, RB, BN, and kNN[4], Support Vector Machines can be slow due to their quadratic time complexity in relation to the number of data points, making training time-intensive, especially for large datasets. For this reason, we are not going to evaluate the performances of SVMs on all the combinations of Preprocessing and Feature Extraction, but only on word embeddings and matrices on which Feature Selection was applied.

3.5 Performance measures

To evaluate the performances of the four algorithms, we take into account different tools and measures.

In general, in classification problems, a very useful tool for performance evaluation is the confusion matrix, since most of the metrics can be obtained analytically from it. This matrix is made up of rows and columns where the real and expected classes are indicated; in this way it is possible to evaluate the four combinations deriving from the classification, i.e: True Negatives, False Positives, False Negatives and True Positives.

From the Confusion Matrix, we can compute three measures: Recall, Precision and F-score.

Recall is equal to:

$$Recall = \frac{TP}{P}, \tag{1}$$

In our case it refers to the percentage of the documents of each class that are correctly identified
Precision is equal to:

$$Precision = \frac{TP}{TP + FP}; \tag{2}$$

it represents the fraction of documents which are classified in one class and that result to be effectively of that class.

From these two metrics, we can compute a third metrics: the F-score:

$$F - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}; \tag{3}$$

in other words, it is an harmonic mean between Precision and Recall. Since we are in a MCML classification problem, we can compute the overall F-score in two ways: with micro and macro averaging[4]. In micro averaging, we compute the F-score utilizing micro-averaged precision and recall. Micro-averaged precision and recall are computed using as TP, FP, and FN the sum of the TP, FP and FN obtained for each classification. Macro averaging, instead, consists in simply computing the arithmetic mean of all the F-1 scores for the different classes.

Though, since micro-averaging is more effective when, as in our case, the datasets vary in size[5], we are going to rely more strongly on this measure.

3.6 K Fold Cross Validation

In order to get a more effective estimate of the classifier's performances we will use for all of the analysis the K Fold cross validation. This method consists in dividing the data into k subsets, and then repeating k times the holdout method (i.e., splitting the data into a training set and a test set), such that each time one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get total effectiveness of our model. In our case, we decide to use the five folds cross validation (k=5).

3.7 Classification Results

We firstly applied the 5-Fold Cross Validation to the 6 combination of PreProcessing (StopWords Removal and Lemmatization) and Feature Extraction (BoW, TF and tf-idf). Each of them with the 3 classifiers. The results are shown in Figure 1.

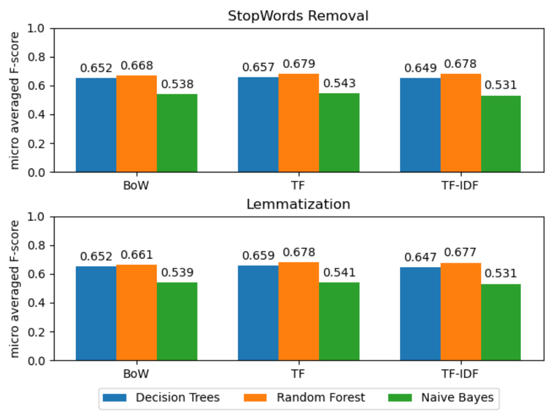

Figure 1: Classification Results

As we can clearly see from the image, Random Forest classifiers achieve the best results in all of the combinations, followed by Decision Trees and, with a relevant gap, by Naive Bayes. This result is in contrast to what researchers have found in [4], where Naive Bayes is regarded as the second best classifier after SVM. For what concerns the Pre Processing techniques, instead, the gap is not wide at all, but StopWords removal without lemmatization seems to achieve better results in more combinations. Lastly, TF seems to be the best Feature Extraction technique among the three. Once again, the gap is not large, but the classifications done with TF are those that achieve the best results.

For this reason, we decide to take as the "best" combination the one with no Lemmatization and computed via the TF matrix, as it is formed by the best overall Pre Processing and Feature Extraction techniques, and it contains the best classification: RF with 67.9%. We are now going to apply Feature Selection on top of this combination. The lower number of features will also enable us to test SVMs. The performance of naive Feature Selection, both in term of classification and execution time, are compared to the performance of the best combination without Feature Selection in Figure 2.

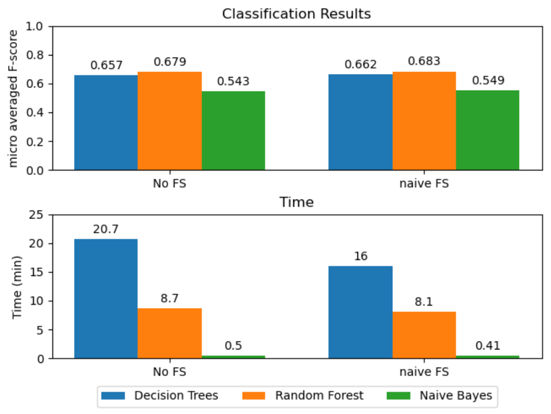

Figure 2: Feature Selection Results

It is important to notice that, since we approach the problem of MCML classification as 5 different binary classifications, the execution time reported is a sum of the execution time of each classifier over the 5 binary classifications. As it can be seen, not only this simple Feature Selection reduces the time required for execution, but it also improves the performances of all the classifiers. Here the Random Forest classifier achieves the best result yet (68.3%).

SVMs, instead, achieved on the dataset with Feature Selection a micro-averaged F-score of 64.7%. This, combined with an execution time of 2 hours and 10 minutes, makes this classifier hard to recommend, at least with this type of Feature Extraction.

Finally, for what regards Principal Component Analysis, we achieved terrible results. Furthermore, the computation of Principal Components, even if advantaged us in the classification process, was quite time consuming. Both these factors make PCA an hard technique to recommend in this type or work.

Let's now focus on Word Embeddings.
In Figure 3 we show the classification results of Classification done with word embeddings as features, where the word rempresentations have been computed with a word2vec model over the dataset's document. More specifically, the model was trained with a 100 vector size and a window of 7.

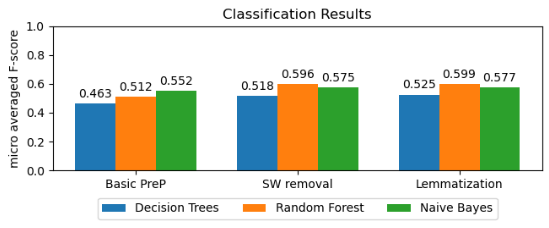

Figure 3: W2V Results

The results obtained fall very short compared to the previous ones. With some tweaks to the parameters (vector size and window size), though, we were able to achieve better results. In Table 1 we compare the micro-averaged F1-score for Random Forest over 9 different combinations of vector and window size:

		Windowsize		
		5	**7**	**10**
Vectorsize	**50**	58.6	59.6	59.8
	100	59.1	59.8	60.6
	200	59.6	60.3	61.4

Table 1: W2V results with different parameters

As we can see, an increase in vector size and window size helps to achieve better results: the best result achieved by the RF classifier is with a window size of 10 and a vector size of 200 (61.4%). A further increase in classification performances is obtained when applying, to this exact model, SVMs: 63.3%. Still, these results remain way lower than the one achieved with a TF matrix with Feature Selection (68.3%).

A very different outcome is instead achieved when applying pretrained word embeddings. In figure 4 we compare the result achieved with this techniques with those achieved by the best combination so far (Stopwords removal, TF feature extraction and naive Feature Selection).

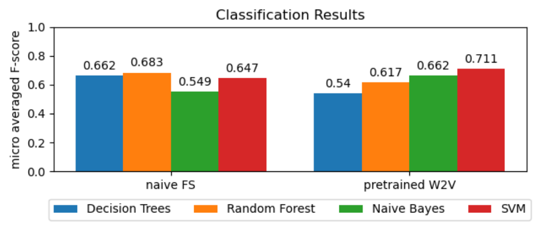

Figure 4: pretrained W2V results

While Decision Trees and Random Forest classifiers drop in performances, the opposite is true for Naive Bayes and SVMs. With SVMs, in fact, we now achieve the best classification result in term of micro averaged F-score: 71.1%. Furthermore, the fact that the pretrained embeddings are of size 200, means that the classifiers have to deal with way less features. In other words, not only pretrained embeddings helped us to achieve the best classification result, but are also way less computationally expensive: all of the classifiers required less then a minute to run, including SVMs.

4 Text Summarization

Summarization helps to shorten the time needed for reading, speeds up the search for information and helps to get the most amount of information on one topic. The central object of computerized text summarization is decreasing the reference text into a smaller version maintaining its knowledge alongside with its meaning [6].

There are two type of summarization: extractive and abstractive. In extractive summarization the summary is created from important phrases or sentences selected from the input text. Abstractive summarization, instead, expresses the ideas in the source documents using different words. In our project we focus on extractive techniques. More precisely, we are going to use two approaches: Graph-Based Method(Indicator Representation Approach) and Latent Semantic Analysis (Topic Representation Approach).

4.1 Graph Based Method

Graph Based methods represent the document as a connected graph where vertices are the sentences, and edges reflect their similarity. With this representation, we are able to figure out the important sentences to be included in the summary based on their connection with the other vertices of the graph.

After computing the graph representation utilizing the similarity matrix between sentences, we are going to apply the "Page Rank" algorithm to retrieve the scores of each sentences. PageRank is an algorithm utilized by Google that attaches a score to Web pages on the basis of the Web connectivity[7]. PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites (in our case sentences) are likely to receive more links from other websites.

4.2 LSA

Latent Semantic Analysis is an algebraic-statistical method that extracts hidden semantic structures of words and sentences. It is an unsupervised approach that does not need any training or external knowledge. LSA is based on the distributional hypothesis: the semantics of two words will be similar if they tend to occur in similar contexts. LSA computes how frequently words occur in the documents and the whole corpus, and assumes that similar documents will contain approximately the same distribution of word frequencies for certain words. In this way, though, syntactic and semantic information is ignored and each document is treated as a bag of words. Singular Value Decomposition, an algebraic method, is used to decompose the document-term matrix into the product of three matrices U, Σ and V^T. In this way, LSA will consider each singular value of the decomposition as a potential topic found in the documents.
Even though LSA performs topic-modelling, the resulting topics can be used for text summarization. If we apply SVD to a matrix A in which each entry a_{ij} corresponds to the weight of word i in sentence j (were weights are calculated with tf-idf), then we can compute the product between Σ and V^T to compute the matrix D, that combines the topic weights and the sentence representation to indicate to what extent the sentence conveys the topic, with d_{ij}

indicating the weight for topic i in sentence j. The assumption is that sentences covering various topics are ideal contenders for summaries. The weight of the sentence s_j is computed as:

$$g(s_j) = \sqrt{\sum_{j=1}^{m} di^2 j} \qquad (4)$$

4.3 Evaluation

For what regards the results evaluation, we focus our attention on the ROUGE metric. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. ROUGE is a measure to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans [8]. However, our dataset does not provide the latter type of reference. To overcome this issue, we use document's title as reference. In other words, we firstly separate, for each observation in our data, the title from the abstract. We do this by using PubMed's API: we search for a publication whose title contains the first words of our observation, we retrieve its title, remove it from the observation and store it in a separate variable. Then, we compute the summarization only on the abstract part of the observation, and compare it to the title via the ROUGE metric.

There are different types of ROUGE:

- **Rouge-N**
  It's computed as the number of common n-grams between the candidate and reference summaries (denoted as p), and the total number of n -grams present in the reference summary (denoted as q). Mathematically:

$$Rouge - n = \frac{p}{q}$$

- **Rouge-L**
  It refers to the longest common subsequence between two texts. All n-grams must be consecutive.

As a second method for evaluating the summaries, we are going to determine how they perform in the classification task. More precisely, we are going to use the pretrained word embeddings to perform feature extraction over the two summaries (Graph based and LSA). Then, we are going to compare their classification abilities with the same classification done with the original text, and with a random summary. The main idea behind this type of measure is that, if a summary is done right, it will retain most of the important and discriminative information of the original text. If this is true, than a good summary should perform better in a classification task like the one we described before, at least compared to a random summary.

4.4 Summarization Results

From the resulting metrics displayed in Figure 5 we can clearly see that the Graph-based summarization performs overall better than the LSA-based summarization.

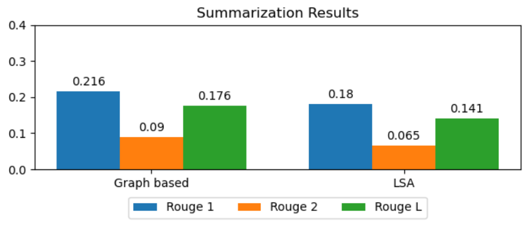

Figure 5: Summarization Results

This is also reflected in their classification ability. As we have seen in the classification results, the micro averaged F-score for the whole document was 71.1% (with SVM), while the same metric is of 63.3% for the random summary. If we apply SVM to the Features extracted from the Graph-based, instead, the micro averaged F-score increses to 64.0%. This demonstrates an higher ability of these types of summaries to retain information useful for discriminating between the five classes. The same cannot be said for the LSA-based summaries, that achieve a score of 62.0%, lower than the random summaries.

5 Conclusions and Future Developments

In this report, we analyzed different techniques to perform Multi Class Multi Label classification and summarization on medical abstracts.
In the first part, we focused on classification. We evaluated three different Pre-Processing techniques, four different Feature Extraction techniques, Two Feature Selection techniques and four classifiers. Our results showed that, when working with Feature Extraction techniques different from Word embeddings, stopwords removal without Lemmatization, a TF

based representation matrix and a Random Forest classifier tend to achieve the best result. The combination of these three techniques, resulted in a 67.9% micro averaged F-score. Furthermore, removing "rare" words (i.e., those that appear only once in the whole collection) results in a boost both in classification performances (68.3%) and in time required for the computation. PCA, instead, did not perform as expected.
When dealing with Word Embeddings, we demonstrated that, if we would like to train them from scratch, then lemmatizing the documents and increasing both the window and vector size of the model will lead to better results. Though, the most crucial factor with word embeddings seems to be the raw amount of data they are trained with. Using the pre-trained word embeddings, in fact, resulted in an overall increase in performances, and resulted in the best classification performances: 71.1% with SVM.
In general, for what regards the classifier, we can say that Random Forest are the way-to-go if we use a feature extraction technique that is different from word embeddings, like BoW, TF and tf-idf. In fact, in this case, SVMs are both worse classifiers and way more time consuming. With word embeddings, instead, SVMs give the best result, and their performance inefficiency is solved by the lower number of dimensions that the embeddings require.

In the second part of the report, we focused instead on text summarization. We computed the summaries with two different techniques: a graph-based model and an LSA-based model. We then evaluated the two techniques with the help of the rouge metrics, and by their performances in the classification task. From our result, the graph-based summarization showed to be an overall better technique. In fact, it achieved higher rouge scores, and demonstrated an higher ability to retain information useful for discriminating between the five classes.

In this paper we wanted to focus on traditional classification and summarization techniques, i.e., not relying on Large Language Models. Though, in future works, it would be interesting to evaluate the performances of classification based on contextualized word embeddings and abstractive summarization. In particular, BERT models that have been fine-tuned over health records, like Med-BERT[9], would probably help to achieve great results.

References

[1] Stergos Afantenos, Vangelis Karkaletsis, and Panagiotis Stamatopoulos. "Summarization from medical documents: a survey". In: *Artificial intelligence in medicine* 33.2 (2005), pp. 157–177.

[2] Tim Schopf, Daniel Braun, and Florian Matthes. "Evaluating unsupervised text classification: zero-shot and similarity-based approaches". In: *arXiv preprint arXiv:2211.16285* (2022).

[3] SPFGH Moen and Tapio Salakoski2 Sophia Ananiadou. "Distributional semantics resources for biomedical text processing". In: *Proceedings of LBM* (2013), pp. 39–44.

[4] Ghulam Mujtaba et al. "Clinical text classification research trends: systematic literature review and open issues". In: *Expert systems with applications* 116 (2019), pp. 494–520.

[5] Marina Sokolova and Guy Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information processing & management* 45.4 (2009), pp. 427–437.

[6] Laith Abualigah et al. "Text summarization: a brief review". In: *Recent Advances in NLP: the case of Arabic language* (2020), pp. 1–15.

[7] Monica Bianchini, Marco Gori, and Franco Scarselli. "Inside pagerank". In: *ACM Transactions on Internet Technology (TOIT)* 5.1 (2005), pp. 92–128.

[8] Chin-Yew Lin. "Rouge: A package for automatic evaluation of summaries". In: *Text summarization branches out*. 2004, pp. 74–81.

[9] Laila Rasmy et al. "Med-BERT: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction". In: *NPJ digital medicine* 4.1 (2021), p. 86.