uc3m | Universidad Carlos III de Madrid

Master Degree in Big Data Analytics

2020-2021

*Master Thesis*

# "Automatic classification of clinical reports with ICD-10 coding"

---

IGNACIO SISAMÓN SERRANO

Tutor:

Isabel Segura Bedmar

September 2021, Madrid

*"Siempre me dicen: "Deja la cabeza y hazle caso a tu corazón" Pero ¿Qué hacer cuando tu corazón está fatal de la cabeza?"*

Love of Lesbian

# SUMMARY

Nowadays, there is a huge amount of clinical reports that need to be classified in order to analyze health status and optimize resources in hospital and medical centers. To ease this task and perform a quicker and more efficient, Natural Language Processing methodology is used to classify automatically clinical case reports with ICD-10 codes. ICD-10 is the $10^{th}$ revision of the International Statistical Classification of Diseases, mainly divided into Diagnosis and Procedure, that allows a basis for a classification standard for worldwide research. To face this problem, a reduced number of clinical cases reports, both in Spanish and English, are given in a competition proposed by CodiEsp, where an imbalance code distribution is observed. In order to classify these reports, two different approaches are proposed. Firstly, a TF-IDF vectorization of clinical reports, followed by the use of some classical machine learning algorithms and finally boosted with an Optuna hyper-parameter optimization. Secondly, BERT that is the state-of-the-art model for text classification.

Results show that the reduced training set provided provokes a low accurate model built with BERT, while classical algorithms perform much better with a final result of *mean Average Precision* = 0.4088 and $micro\_f1 = 0.6028$ for Diagnosis. In addition to the competition proposed, and in order to check the models in a more realistic environment, a reduction of the problem is proposed with a more balance distribution of codes by using just the most frequent ones. In this case, BERT model improves its performance, but it is still far from traditional machine learning classifiers.



**Keywords:** CodiEsp - Clinical records - ICD-10 - BERT - Multi-label classifiers

# Dedication

To family and friends. "Pa eso están, están pa eso".

# Contents

**1 Introduction** **1**

1.1. Objectives of the master thesis . . . . . 3

1.2. Socio-economical impact . . . . . 3

1.3. Regulatory Framework . . . . . 4

**2 Related work** **5**

**3 Methodology** **7**

3.1. Data preprocessing . . . . . 7

3.2. Multi-label methodology . . . . . 8

3.2.1. Machine Learning classifiers and TF-IDF vectorization . . . . . 8

3.2.1.1 KNN . . . . . 10

3.2.1.2 Decision Trees . . . . . 11

3.2.1.3 Random Forest . . . . . 11

3.2.1.4 Bagging Classifier . . . . . 11

3.2.1.5 Gradient Boosting Classifier . . . . . 11

3.2.1.6 Linear SVC . . . . . 12

3.2.1.7 Hyperparameter Optimization: Optuna . . . . . 12

3.2.2. BERT . . . . . 12

3.2.2.1 Fine-tuning . . . . . 14

**4 Evaluation** **15**

4.1. Metrics . . . . . 15

4.2. Dataset . . . . . 18

4.3. Results . . . . . 24
4.3.1. All contained labels problem . . . . . 24
4.3.1.1 Diagnosis . . . . . 24
4.3.1.2 Procedure . . . . . 25
4.3.2. Most frequent labels problem . . . . . 26
4.3.2.1 Diagnosis . . . . . 27
4.3.2.2 Procedure . . . . . 27
4.3.3. Optuna optimization . . . . . 28
4.3.4. Discussion . . . . . 30

**5 Project Management** **33**
5.1. Project Infrastructure. . . . . 33
5.2. Planning. . . . . 33

**6 Conclusion** **34**
6.1. Future of the project . . . . . 35

**Bibliography** **37**

# List of Figures


1.1 Clinical text mining flow [2] . . . . . 2

3.1 Bert input representation [21] . . . . . 13

4.1 Distribution of ICD-10 codes in Procedure . . . . . 20

4.2 Distribution of ICD-10 codes in Diagnosis . . . . . 21

4.3 Frequency Distribution of Top 30 tokens for Diagnosis (English left, Spanish right) . . . . . 22

4.4 Frequency Distribution of Top 30 tokens for Procedure (English left, Spanish right) . . . . . 23

4.5 UMAP visualization for Diagnosis (English left, Spanish right) . . . 23

4.6 UMAP visualization for Procedure (English left, Spanish right) . . . 24

4.7 MAP score as a function of language and algorithm . . . . . 31

4.8 MAP score as a function of language and algorithm (most frequent labels) . . . . . 31

4.9 MAP score: Optuna comparison . . . . . 32

4.10 MAP score: Optuna comparison (most frequent labels) . . . . . 32

5.1 Project planning . . . . . 33

# List of Tables


3.1 Frequency of some words in documents example . . . . . 9

4.1 Example for precision and recall explanation . . . . . 16
4.2 Precision, Recall and $F_1$ score for example explanation . . . . . 16
4.3 Statistics of the datasets with clinical cases for Diagnosis (English) . 18
4.4 Corpora clinical cases statistics for Diagnosis (Spanish) . . . . . 18
4.5 Corpora clinical cases statistics for Procedure (English) . . . . . 19
4.6 Corpora clinical cases statistics for Procedure (Spanish) . . . . . 19
4.7 Results for all labels D English . . . . . 25
4.8 Results for all labels D Spanish . . . . . 25
4.9 Results for all labels P English . . . . . 26
4.10 Results for all labels P Spanish . . . . . 26
4.11 Results for 146 labels D English . . . . . 27
4.12 Results for 146 labels D Spanish . . . . . 27
4.13 Results for 45 labels P English . . . . . 28
4.14 Results for 45 labels P Spanish . . . . . 28
4.15 Optuna results for Diagnosis English (left) and Spanish (right) . . . 29
4.16 Optuna results for Procedure English (left) and Spanish (right) . . . 29
4.17 Optuna results for Diagnosis English (left) and Spanish (right) . . . 29
4.18 Optuna results for Procedure English (left) and Spanish (right) . . . 30

# Chapter 1

# Introduction

Nowadays, the pandemic caused by COVID-19 highlights the importance of an efficient system for searching, analysis and exploitation strategies for different medical contexts. Clinical case studies are detailed reports of the symptoms, signs, diagnosis or treatment of a patient, that are included in the Electronic Health Records (EHR) [1]. Nevertheless, the extraction of relevant information from *EHR* is a complex exercise due to several factors: high amount and growth of data worldwide, diversity of structures and formats (lack of standardization), complex vocabulary and terminology used and diversity of languages used around the world [2].

One solution to solve these issues is the classification of clinical reports with the ICD-10 coding, performed by experts that requires to classify each clinical case report with their corresponding codes [3].

The ICD-10 is the $10^{th}$ revision of the **I**nternational Statistical **C**lassification of **D**iseases and related Health Problems. Its main purpose is the identification of different health trends and the creation of global statistics. Besides, it is the base for an international diagnostic classification standard for clinical and research purposes.

It allows a hierarchical and easy storage of health information for decision-making; shared information between hospitals, regions and countries; data comparisons across different time periods, etc [4]. Furthermore, it is suitable for monitoring incidence and prevalence of diseases, causes of deaths or even recording of rare diseases.

Introducing some history of the ICD, the first international classification, known as "List of Causes of Death", was adopted in 1983 by the International Statistical Institute [4]. Since them, the ICD has been revised and republished until the $10^{th}$ edition, implementing the $11^{th}$ (ICD-11) on January, 2022. With the creation of the WHO (**W**orld **H**ealth **O**rganization) in 1948, the ICD jumped into the $6^{th}$ edition remaining under WHO's supervision. Finally, the ICD-11 represents a migration from a mere statistical framework to a clinical classification for statistical use.

In this project, the CIE-10 is the version used. It consists on three different volumes:

Volume 1 contains the disease classification; Volume 2 contains the fundamentals for the instructions manual; and Volume 3 encompasses an alphabetic index of diseases and diagnosis terms for a fast positioning [5].

Focusing on Volume 1, it enclose the **Tabular list**, which contains the minimum level to report to the WHO and the classification itself at the three (category) and four (subcategory) character levels [6]. A category is the key or code with 3 characters (composed by a letter in the first position A-Z, followed by two different numbers 00-99; for example: A00 -> cholera) corresponding to a disease or diagnosis term. A subcategory is a subdivision of the category, indicated by adding an extra character with a dot from .0 to .9 (for example: A00.0 Cholera due to Vibrio cholerae 01, biovar cholerae)

Nevertheless, the ICD-10 classification is a complex and high time consuming task that requires lots of resources that are not currently available. Therefore, the analysis of health scenarios is not fully complete and many times arrives late.

This is where Artificial Intelligence and Natural Language Processing (NLP) enter, developing methods to manage and extract information from these clinical cases. Having a strong model to perform a fast and correct classification allows hospitals and medical centers to optimize and reorganize resources.

One possible NLP approach is to detect medical entities such as treatments and diagnosis with Named Entity Recognition (NER), and link them with ICD-10 codes. The flow needed in this case would be the one shown in Figure 1.1. The main issue with NLP and the EHR is that most of the NLP tools are developed for English language, while EHR are in native language. Therefore, efforts are now focused on developing NLP tools for different languages than English.

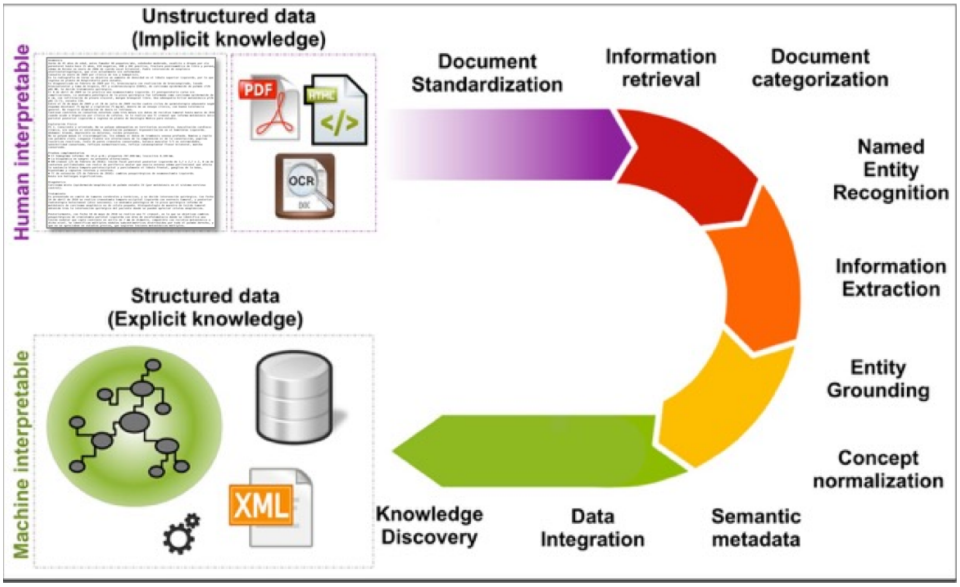


Figure 1.1: Clinical text mining flow [2]

However, in this project the approach taken is a multi-label problem, in which classification algorithms assign multiple labels to each clinical case report. To do so, two different methods can be used to solve a multi-label problem: a problem transformation in which multi-label problem is transformed into single-label problems or the use of ensemble methods.

To boost the investigation on text mining focused on clinical texts, CodiEsp proposed a competition on classifying Spanish clinical reports contained in EHRs [2], which is faced in this project.

### 1.1. Objectives of the master thesis

The main goal of this master thesis is to look for the best approach for the Task 1: Multilingual Information Extraction at CodiEsp CLEF eHealth 2020 [2]. To do so, several approaches are tested facing two different problems, a complete one (using all codes) and a reduced one (using only most frequent codes).

To complete this main goal, four specific objects must be achieved:

1. Investigation on the current state-of-the-art methodologies for text mining (BERT and TF-IDF) and machine learning classification algorithms.
2. Implementation of classical machine learning algorithms to label the clinical cases by using first a TF-IDF vectorization.
3. Fine-tuning of a pre-trained BERT model for the CodiEsp task.
4. Comparison of results.
5. Conclusion and future studies proposals.

### 1.2. Socio-economical impact

The investigation on the automatic coding of clinical cases reports into the ICD-10 represents a huge impact on the society. Even though at first sight there is no direct repercussion, the easiness on having all world's clinical reports classified provides a really wide view of the planet health situation. In case of a pandemic, as the COVID-19 one, this would give information for all countries about the disease before even spreading in their own country.

Furthermore, on a daily basis, having all clinical cases coded would help hospitals and medical centers in the optimization of resources by foreseeing future needs.

On the other hand, to get to this situation, the investment needed in this system is really huge. Nowadays, there is no system capable of even help experts in the ICD-10 classification. Besides, most NLP tools that can be applied to address the task

are trained only for English, which leads either to invest in the training for other languages or create a stronger structure on the clinical cases reports worldwide with English as unique language. This effort should be from the whole planet, involving all countries and health systems.

### 1.3. Regulatory Framework

Recalling the economical investment needed for this project, the regulatory framework is another obstacle added. The personal protection of data creates a big wall when accessing this kind of data. In Spain, this is regulated with the *"Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales."* [7].

Clinical cases reports belong to the patient and not to health institutions. In fact, even hospital personnel as doctors or nurses cannot access freely to any clinical case study; they can only access to those reports of the patients that are currently under their supervision [8]. However, in the vast majority cases, hospital ethics committee authorizes researchers to perform studies with anonymous data.

# Chapter 2

# Related work

Automating medical coding has been aimed for several years; in fact, automatic classification of medical records is an active field of research in task competitions and Natural Language Processing [3]. Over the years, techniques to perform this task have evolved from Dictionaries [9], to statistical models [10], machine learning models and, recently, Deep Learning models [11].

Taking a deeper look to [12], the growth on the number of studies performed with it is exponential. Nevertheless, the improvement on the results has grown quite slow. In fact, one of the biggest challenges of this task is the shortage of training data in both clinical and NLP field. As NLP is wide with lots of different tasks, labelling training examples is not easy and fast-forward. This is where Deep Learning models outline: they benefit and improve from millions and billions of labelled data. Therefore, researchers used the amount of text available on the web to pre-train these models and then fine-tuned these models on smaller samples of data for specific tasks, improving performance.

Looking at the CodiEsp competition [2], different methodologies were employed. Before starting the explanation, it is highlighting to mention not only the wide options used by teams, but also the top performance methodology that differs from Diagnosis to Procedure (machine-learning methods versus non-machine-learning method respectively). The methodologies can be grouped in three different approaches:

- Multi-label text classification: a set of documents need to be classified according to ICD-10 codes.
- Named Entity Recognition (NER): automatic algorithms detect key clinical word or words to assign codes.
- Combination: mix the two previous methods.

Therefore, some approaches used by teams are:

1. *IXA-AAA*. This team used a combination of text classification and NER methodologies. On the classification phase, a binary XGBoost classifier was trained for each label. Then, using BERT representation, similarities between text fragments and ICD-10 code definitions were performed in order to estimate the probability for assigning codes. Besides, all texts were expanded using medical entities and phrases frequently associated to codes. This team achieved a maximum value of mean average precision of 0.698 for Diagnosis and 0.481 on Procedures [13].

2. *IAM*. IAM used a dictionary with a tree-data-structure composed with the corpus given in the competition and the ICD-10 terminology. Then, using distance matching between a new word and words included in the dictionary, codes are assigned achieving a mean average value of 0.52 for Diagnosis and 0.43 for Procedures [14].

3. *The Mental Strokers*. This team re-trained the BETO (Spanish BERT-model) model using the training set of the competition. After that, a linear classification layer was added and NER was performed by fine-tuning the model [15]. Finally, the team achieved a mean average precision of 0.517 for Diagnosis and 0.445 for Procedure.

4. *FLE*. This approach was based on two steps. First, using the pre-trained multilingual BERT, entities were identified. Then, these entities were matched with ICD-10 code definitions through a linking algorithm (Knowledge Graph) [16]. In addition, they increased the size of the training set by using a text augmentation algorithm (Fujitsu Augmentation), trained with the corpus and some examples from PubMed [17]. The team got a mean average value of 0.519 for Diagnosis and 0.443 for Procedure.

Summarizing the above information, the best approach in the competition is a combination of multi-label text classification with a NER mothodology used by *IXA-AAA*. In addition, these 4 teams results are around 0.5 for mean average precision.

# Chapter 3

# Methodology

In this section, all the information related to the dataset, data preprocessing and the different approaches used are described.

**The dataset**

The corpus of this project is based on two main datasets, as mentioned before, divided by language: Spanish and English. Each dataset follow the same structure, with two different sets of data for the diagnosis and the procedure, each one subdivided into train, development,test and background sets:

- Train set: composed by 500 clinical cases.
- Development set: composed by 250 clinical cases.
- Test set: composed by 250 clinical cases.
- Background set: additionally, there is a background set composed by 2.751 clinical cases (with no assigned code).

Each clinical case report is represented by its articleID and its corresponding ICD10-code.

## 3.1. Data preprocessing

The first step is to use all the information given relating each clinical case study with their corresponding codes for train, development and test sets. Besides, BERT and multi-label approaches requires different input structure of data in order to use their algorithms.

Once this is performed, a technique to reduce the vocabulary of text documents is to apply several transformations. These transformations are in order:

1. **Word Lemmatizer**: transform words into its main lemma (studying -> study // studied -> study) reducing the number of words and grouping them.
2. **Short words**: usually, short words (1 or 2 characters' length) are meaningless for text analysis, therefore they can be erased. This step can be combined with the following one.
3. **Stopwords**: this method contains words that are considered useless for text mining (differently to the previous case, they are independent on the length) and remove them for the analysis. Examples: "a", "you", "theirs", or in Spanish, "a", "en", "la".

## 3.2. Multi-label methodology

Two different approaches have been used for the same solution, both approaches facing the same standpoint, a multi-label problem to classify clinical reports.

The first approach is to use the classical machine learning algorithms used successfully for text classification as decision trees or close neighbours techniques. To do so, first a transformation of the texts to vectors is performed through the TF-IDF technique [18].

The second approach is to use the state-of-the-art BERT model and then compare the results obtained.

### 3.2.1. Machine Learning classifiers and TF-IDF vectorization

- **TF-IDF**

TF-IDF (Term Frequency – Inverse Document Frequency) is a numerical statistic method used to reflect how important a word is to a document in a corpus. Its main idea is to drop most common terms and extract only the most relevant ones from a corpus. This value increases proportionally to the number of times a word appear in any document of the corpus and it is offset by the number of documents on this corpus, adjusting the fact that some words appear more frequently in general [18].

In this methodology, the number of times a term occurs in a document is called *term frequency (TF)*. However, using the example of the word "the", the term frequency will tend to emphasize documents which use the word "the" more frequently incorrectly, without giving the needed weight to the real important words. As can be summarized by [19]: The weight of a term that occurs in a document is simply proportional to the term frequency:

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{3.1}$$

where $f_{t,d}$ is the number of times a term *(t)* appears on document *(d)*, and $\sum_{t' \in d} f_{t',d}$ is the number of terms that appears on document *(d)*.

This is the reason of the *Inverse Document Frequency (IDF)*, which is a factor that corrects this emphasis and increases the weight of terms that occur less often. In this case, the summary by [20] is: The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs:

$$\text{IDF}(t, C) = \log \frac{N}{|\{d \in C : t \in d\}|} \tag{3.2}$$

where $N$ is the total number of documents in corpus C and the denominator ($|\{c \in C : t \in c\}|$) is the number of documents where the term $t$ appears.

Therefore, the TF-IDF is the product of these two mentioned terms: *term frequency* and *Inverse Document Frequency*. Joining equations 3.1 and 3.2, the result is:

$$\text{TF-IDF}(t, d, C) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} * \log \frac{N}{|\{d \in C : t \in d\}|} \tag{3.3}$$

To ease the explanation, an example can be proposed. Setting a corpus (C) of two different documents with the following sentences:

- Document 1 (D1): "This is my example"
- Document 2 (D2): "This example is a good example as a example".

Choosing some words and making the counting on table 3.1:

| Document 1 | | Document 2 | |
|---|---|---|---|
| This | 1 | This | 1 |
| is | 1 | is | 1 |
| my | 1 | good | 1 |
| example | 1 | example | 3 |

Table 3.1. FREQUENCY OF SOME WORDS IN DOCUMENTS EXAMPLE

The *term frequency* for each document is:

$$TF(\text{"}example\text{"}, D1) = \frac{1}{4} = 0.25 \qquad TF(\text{"}example\text{"}, D2) = \frac{3}{9} = 0.33$$

Now, the *inverse document frequency* is 1 as the word "example" appears in the two documents of the corpus:

$$IDF(\text{"}example\text{"}, C) = \frac{2}{2} = 1$$

Therefore, the TF-IDF is:

$$TF - IDF(\text{"}example\text{"}, d1, C) = 0.25 * 1 = 0.25$$

$$TF - IDF(\text{"}example\text{"}, d2, C) = 0.33 * 1 = 0.33$$

- **Machine Learning algorithms**

The machine learning algorithms used in this project have been selected due its excellent results on text classification problems.

Before explaining the algorithms, it is important to recall that the problem faced is a multi-label problem, which means that for each clinical case study different codes can be assigned. Therefore, Bagging's algorithm, Gradient Boosting and Naive Bayes need a previous transformation in order to answer the multi-label problem. To do so, the OneVsRestClassifier from *sklearn*, also known as One-vs-All, is implemented. This algorithm consists on fitting one classifier for each different class. Then, each class is fitted against all other classes for each classifier, providing great computational efficiency.

#### 3.2.1.1 KNN

The KNN (k-nearest neighbours algorithm) is an algorithm developed by Evelyn Fix and Joseph Hodges [25] and later expanded by Thomas Cover. This algorithm is mainly used for classification and regression, where the output is some class and some value respectively [24].

After the transformation of clinical cases studies into numerical vectors, as mentioned in TF-IDF section 3.2.1, the training phase is based on simply storing the vectors together with their corresponding labels. Then, in the classification phase, an unlabelled vector (a test clinical case study) is classified by assigning most frequent labels among the $k$ training samples nearest to that query point. In this case, $k$ is a user-defined parameter that establish the number of neighbours to split the data in.

This algorithm relies in the distance to neighbours. Nevertheless, this distance can be computed in several ways. The most widely used, and indeed used by KNN algorithm, is the Euclidean distance, which computes the length of the line segment between two points.

The main drawback of this algorithm is the "mayority voting" rule, in which the most frequent class dominate the prediction in a skewed distribution [26].

#### 3.2.1.2 Decision Trees

Decision trees are a non-parametric supervised learning method developed for both classification and regression. The main goal of this algorithm is to create a tree model that predicts the value of a target variable by learning simple decision rules inferred from the data [27].

This algorithm has some advantages as its simple understanding and interpretation, requires few data preparation (it handles blank values). On the other hand, its main drawbacks are the possibility of over-fitting, instability due to fluctuations in data and the creation biased models [28].

#### 3.2.1.3 Random Forest

Random Forest is an ensemble learning method (use multiple learning algorithms to obtain better predictive performance) for classification, regression that operates by constructing a multitude of decision trees during training. The output given for the classification model is the class given by most tress, whereas the output for regression is given by the mean of all tress [29].

As mentioned before on Decision Trees, one disadvantage of them is the possible over-fitting. However, this drawback is overcome thanks to the use of several trees, reducing the variance of the model.

#### 3.2.1.4 Bagging Classifier

Bagging classifier is another ensemble learning method that fits base classifiers on different random substets of the original dataset. Then, it votes or averages by aggregating each classifier prediction to form a final prediction. Due to this randomness on the dataset splitting, the variance is reduced compared to other methods [30].

Despite these advantages, the creation of several classifiers can lead to a computationally expensive model. Besides, for high-bias datasets, bagging model drags this bias to the result. Finally, there is a loss of interpretability of the model [31].

#### 3.2.1.5 Gradient Boosting Classifier

Gradient Boosting is an additive model, as random forest, that allows the optimization of arbitrary differentiable loss functions by combining decision trees. Its implementation allows to choose for the loss function between the logistic regression or the exponential one [32].

As with Bagging Classifier, its main drawback is the loss of interpretability and the computational demand. This algorithm is one of the most widely used and it

normally outcomes Random Forest [33].

#### 3.2.1.6 Linear SVC

Support Vector Machines (SVM) are a set of learning methods used for classification and regression. On the training phase, the algorithm builds a model that assign new samples one category or the other by mapping training examples to points space and maximizing the width of the gap between those two categories [34].

The main advantages of support vector machines are its effectiveness in cases where number of dimensions is greater than number of samples, its versatility on the kernel functions that defines boundaries for decision (decision function). On the other hand, the main disadvantage is its long train training and the difficulties on the interpretability of the final model. Linear SVC (Support Vector Classification) uses SVM methodology with a linear kernel.

#### 3.2.1.7 Hyperparameter Optimization: Optuna

Optuna is an automatic hyperparameter optimization software framework, particularly designed for machine learning, which implements sampling algorithms to exploit correlation between parameters [35]. It is designed for easy implementation, flexibility and scalability, allowing a parallel and distributed performance [36].

Its implementation in Python is quite simple with just three steps:

1. Definition of search space: enumerate hyperparameters suggestions.
2. Implementation of the model: choose model (pytorch, keras, scikit-learn...)
3. Value to be optimized: select the metric to optimize.

Optuna optimization is used only on the top performing algorithms, to try to get better results. The optimization is focused on the mean average precision metric, which is the base score for the competition.

### 3.2.2. BERT

Pre-trained representations can be either contextual (directional or unidirectional) or context free. Examples of context free models are *word2vec*,[22] which uses a neural network to learn word associations from a large corpus of text, or *GloVe*,[23] whose training is performed on global word-to-word co-occurrence statistics from a corpus and final representations show linear substructures in the word vector space. This means that any word, for example "park, has the same representation in the

sentence "We can go to the park next Monday" and "Sir, park the car on the right". On the other hand, contextual models have a different representation based on the other words of the sentence. If the model is unidirectional, on the first sentence "park" is represented based on the first part of the sentence "We can go to the", or "Sir" in the second one, but not the rest of the sentence. This is where BERT enters, representing "park" with both the previous and next context (bidirectional).

BERT is the acronym for Bidirectional Encoder Representations from Transformers, a new language representation model based on Artificial Intelligence *(AI)* for NLP developed by Google in 2018 [21].

BERT is designed as an unsupervised language representation, whose main feature is to be the first deeply bidirectional model, pre-trained using only plain text corpus [12]. In addition, the pre-trained BERT model can be tuned additional layers to create outstanding models for a wide range of task.

BERT's way of working is quite simple:

**Masked LM (MLM)**

Before starting the analysis, BERT replace a 15% of the words by a [MASK] token. Then, the model attempts to predict this original value of the masked words based on the other words. Therefore, the prediction of the output needs:

1. Creation of a classification layer on the top of the encoder output.
2. Transformation of output vector with the embedding matrix into vocabulary dimension.
3. Calculation of the probability of each word with the softmax function.
4. Disregarding prediction for non-masked words.

**Next Sentence Prediction (NSP)**

Once the training starts, the model receives pairs of sentences, 50% of them subsequent and the other 50% non-subsequent, learning to predict if the second sentence is subsequent or not. This is performed by inserting a [CLS] token at the beginning of the first sentence and a [SEP] token at the end of all sentences. This scheme is shown in Figure 3.1:

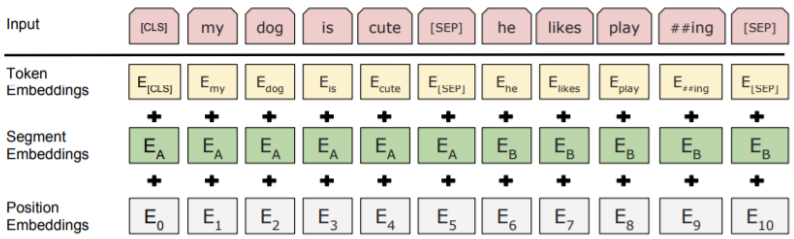


Figure 3.1: Bert input representation [21]

To predict if the second sentence is the subsequent one, it is required:

1. The input sequence goes through the Transformer model.
2. The output of the [CLS] token is transformed into a 2x1 vector using a classification layer.
3. Calculate the probability of being the subsequent sentence with the softmax function.

In order to minimize the combined loss function of the two strategies, the MLM and the NSP are trained together during training phase of BERT.

#### 3.2.2.1 Fine-tuning

The BERT model is composed of different terms in the fine-tuning part:

- Bert model pre-trained: used with two different models [37]:
  - *Bert-base-multilingual-uncased*: 12-layer, 768-hidden, 12-heads, 168M parameters. Trained on lower-cased text in the top 102 languages (used for Spanish corpus).
  - *Bert-base-uncased*: 12-layer, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased English text.
- Dropout: used for regularizing Deep Neural Networks. Normally, these kind of networks tend to overfit easily with few examples [38]. Therefore, the best technique to avoid this is to randomly drop out nodes during training, known as Dropout. In fact, the Bert model pre-trained has dropout regularizations in between its own encoders. This method has a hyper-parameter controlling the probability at which the output of the layer is dropout.
- Dense Layer: concatenation of 4 different dense layers reducing the dimensions progressively to achieve better results. A Dense Layer have the same formula as a linear layer (wx + b), but the final result is passed through an Activation function (non-linear). In this model , the activation function chosen is the sigmoid as the aim is to predict probabilities of one code belonging to any ICD-10 code.
- Loss function: the loss function used in this model is the *BinaryCrossEntropy*, that computes the cross-entropy loss between true labels and predicted labels. The main reason for choosing this loss function is the binary problem (1 assigns a code, 0 not assigned) faced.

# Chapter 4

# Evaluation

## 4.1. Metrics

For Task 1: Multilingual Information Extraction at CodiEsp CLEFeHealth 2020, the competition was focused on maximizing the algorithm in terms of the **mAP** (mean Average Precision, explained below). However, there are other metrics that can be taken into account in order to check the algorithm performance:

- **Precision**: in a binary problem, it denotes the proportion of predicted positive cases that are in fact real positive cases. That is, take all the predicted cases as positive and see how many are truly positive. Extending this idea for the multi-label problem, precision is calculated as the sum of true positives cases across all classes divided by the sum of true positives and false positives cases [39]. Equation 4.1 is given for a specific label:

$$Precision_i = \frac{M_{ii}}{\sum_j M_{ji}} \tag{4.1}$$

- **Recall**: contrary to precision, in a binary problem, the recall is the proportion of real positive cases that are correctly classified as positive. That is, take all the actual positives cases and see how many are predicted as positive. For multi-label case, recall is calculated as the sum of true positives across all classes divided by the sum of true positives and false negatives cases [39]. Giving the equation 4.2 for a specific label:

$$Recall_i = \frac{M_{ii}}{\sum_j M_{ij}} \tag{4.2}$$

To ease the explanation for precision and recall for multi-label problem, an example is proposed. Let us imagine there are three different ICD_10 codes (A00.1, B90.8 and D50.1) with the corresponding results on Table 4.1:

| | | TRUE/ACTUAL | | |
|---|---|---|---|---|
| | | A00.1 | B90.8 | D50.1 |
| PRE | A00.1 | 4 | 6 | 3 |
| DIC | B90.8 | 1 | 2 | 0 |
| TED | D50.1 | 1 | 2 | 6 |

Table 4.1. EXAMPLE FOR PRECISION AND RECALL EXPLANATION

For example, the precision for the A00.1 code, using equation 4.1, is the number of correctly predicted clinical cases studies ($M_{11} = 4$) out of all predicted clinical cases with this code ($\sum_j M_{1j} = 4+3+6 = 13$), which amounts to 4/13=0.308. This means that only one third of the clinical cases studies coded as A00.1 have actually this code.

On the other hand, using equation 4.2, the recall is the number of correctly predicted clinical cases studies with A00.1 ($M_{11} = 4$) code out of the number of actual clinical cases with this code ($\sum_j M_{i1} = 4+1+1 = 6$), which is 4/6=0.667. In this case, two thirds of the clinical cases with code A00.1 are classified with this code.

In order to combine both the precision and recall in only one measure, the **F-score** can be used. Its definition is given in equation 4.3:

$$F_\beta = \frac{\left(\beta^2 + 1\right) P * R}{\beta^2 * P + R} \quad (0 \leq \beta \leq +\infty) \tag{4.3}$$

where $\beta$ is a parameter that controls the balance between Precision (P) and Recall (R). For $\beta = 1$, the $F_1$ becomes the harmonic mean of P and R; for $\beta > 1$, F becomes more recall-oriented while for $\beta < 1$, it becomes more precision oriented.

As before, the example from table 4.1 is going to be used to ease the explanation. Calculating the precision, and recall for each code, the F1-score is also calculated using equation 4.3. Using again code A00.1, $F_1$ is , $F_1 = 2(0.308 * 0.667)/(0.308 + 0.667) = 0.421$. Summarising results in table 4.2:

| Class | Precision | Recall | F1-score |
|---|---|---|---|
| **A00.1** | 0.308 | 0.667 | 0.421 |
| **B90.8** | 0.667 | 0.200 | 0.308 |
| **D50.1** | 0.667 | 0.667 | 0.667 |

Table 4.2. PRECISION, RECALL AND $F_1$ SCORE FOR EXAMPLE EXPLANATION

Once the measurements of all classes are calculated, the classifier's overall performance in terms of the F1 score can be computed. However, there are different ways

to calculate this overall performance:

- *Macro F1-score*: the first option is to compute the average mean of the F1-scores, that is to give the same weight to all classes:

$$Macro - F1 = \frac{0.421 + 0.308 + 0.667}{3} = 0.465$$

However, when there is class imbalance (some classes appear much more than others), two other options are available.

- *Weighted F1-score*: the number of appearances of each class is taken into account when computing the mean. In this case, there are 25 samples: 6 A00.1, 10 B90.8, and 9 D50.1. Therefore, the weighted F1-score is:

$$Weighted - F1 = \frac{0.421 * 6 + 0.308 * 10 + 0.667 * 9}{25} = 0.464$$

- *Micro F1-score*: in this case, the computation of the micro-precision and micro-recall for the classifier is needed before calculating the F1. To do so, the micro-precision and micro-recall are obtained by extending equation 4.1 and 4.2, respectively, for all classes:

$$Micro{-}Precision = \frac{\sum_i M_{ii}}{\sum_{\substack{i \\ i \neq j}} \sum_j M_{i,j}} = \frac{4 + 2 + 6}{(4 + 2 + 6) + (6 + 3 + 1 + 0 + 1 + 2)} = 0.480 \tag{4.4}$$

$$Micro{-}Recall = \frac{\sum_i M_{ii}}{\sum_{\substack{i \\ i \neq j}} \sum_j M_{i,j}} = \frac{4 + 2 + 6}{(4 + 2 + 6) + (6 + 3 + 1 + 0 + 1 + 2)} = 0.480 \tag{4.5}$$

In this case, the micro-precision and the micro-recall are the same, which means that taking the harmonic mean will result in the same result: *Micro-F1 = Micro-Precision = Micro-Recall*. Therefore:

$$Micro - F1 = 0.480$$

As it can be seen, with this example in which there is few class imbalance, results differ depending on the F1-score selected. On the results section, the chosen score is the *Micro-F1* due to class imbalance.

- **Mean Average Precision** (mAP): is the mean of the Average Precision score, which represents the weighted mean of all precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight. Looking at its equation 4.6:

$$\text{AP} = \sum_{n} (R_n - R_{n-1})\, P_n \tag{4.6}$$

where n represents the $n^{th}$ threshold. Therefore, for the mean Average Precision, the expression is seen in equation 4.7 :

$$\text{mAP} = \frac{1}{k} \sum_{k} AP_k \tag{4.7}$$

where in this case k represents each of the different classes (ICD-10 codes).

## 4.2. Dataset

As mentioned on Section 3, the dataset is divided into three different subsets. Taking a deeper look on them, some statistics regarding clinical cases studies can be obtained and summarized on Tables 4.3 (English) and 4.4 (Spanish) for Diagnosis; and Tables 4.5 and 4.6 for Procedure:

| Subset / Statistics | **Train** | **Development** | **Test** |
|---|---|---|---|
| **Number of clinical cases** | 500 | 250 | 250 |
| **Number of words** | 141,228 | 71.837 | 71,250 |
| **Average of words** | $282,46 \pm 131,92$ | $287.35 \pm 135.77$ | $285.0 \pm 131.69$ |

Table 4.3. STATISTICS OF THE DATASETS WITH CLINICAL CASES FOR DIAGNOSIS (ENGLISH)

| Subset / Statistics | **Train** | **Development** | **Test** |
|---|---|---|---|
| **Number of clinical cases** | 500 | 250 | 250 |
| **Number of words** | 172,528 | 86,913 | 87,149 |
| **Average of words** | $345.06 \pm 162.53$ | $347.65 \pm 165.34$ | $348.6 \pm 160.83$ |

Table 4.4. CORPORA CLINICAL CASES STATISTICS FOR DIAGNOSIS (SPANISH)

| Statistics \ Subset | **Train** | **Development** | **Test** |
|---|---|---|---|
| **Number of clinical cases** | 500 | 250 | 250 |
| **Number of words** | 126,888 | 66,297 | 65,401 |
| **Average of words** | 292.37 ± 131.2 | 301.35 ± 131.51 | 294.6 ± 125.89 |

Table 4.5. CORPORA CLINICAL CASES STATISTICS FOR PROCEDURE (ENGLISH)

| Statistics \ Subset | **Train** | **Development** | **Test** |
|---|---|---|---|
| **Number of clinical cases** | 500 | 250 | 250 |
| **Number of words** | 154,797 | 80,250 | 80,077 |
| **Average of words** | 356, 68 ± 160.73 | 364.77 ± 162.26 | 360.71 ± 153.74 |

Table 4.6. CORPORA CLINICAL CASES STATISTICS FOR PROCEDURE (SPANISH)

As it can be seen, the average length of the different clinical cases studies of each subset inside Diagnosis or Procedure are quite similar. Nevertheless, focusing on the length of clinical cases inside subsets, there is a bigger variation on the average words of each case. Comparing Diagnosis with Procedure, Diagnosis clinical cases studies are normally longer than Procedures ones.

On the other hand, differences among languages can be observed. The corpus given in both languages is the same, the original in Spanish, and then translated to English with a machine translation system adapted with the medical domain [40]. Assuming the translation system works pretty accurate, a difference in the number of words required to explain a clinical case diagnosis or procedure can be seen. English requires less words than Spanish to explain the same clinical report, which could influence the TF-IDF method.

Furthermore, a different analysis can be carried out. For the performance of the algorithm, it is important to check different variables, as it is the distribution in terms of ICD-10 codes number of appearances on the training set.

Therefore, on Figure 4.1, the distribution for the number of appearances of each code can be seen. As it can be predicted from the big amount of codes (563) and the reduced number of clinical codes studies (500), more than 50% (66.43%) of codes appear just one time on the train set. In fact, 78.33% of codes appear one or two times and 96.09% appears less than ten times.

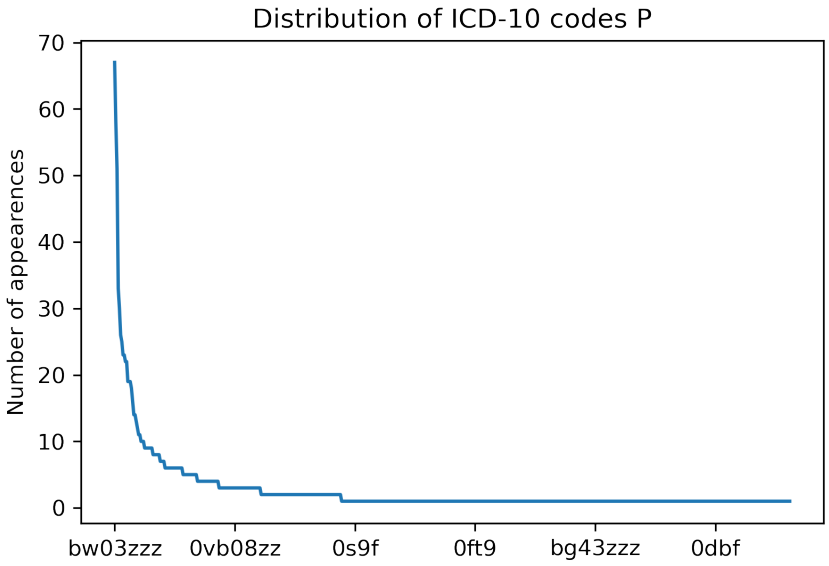


Figure 4.1: Distribution of ICD-10 codes in Procedure

The clinical codes that are appear more frequently are:

- *bw03zzz*: Plain Radiography of Chest // Radiografía simple de Tórax // 67 appearances.
- *bw40zzz*: Ultrasonography of Abdomen // Ecografía simple de Abdomen // 58 appearances.
- *bw20*: Computerized Tomography (CT Scan) of Abdomen // Tomografía computarizada de Abdomen // 51 appearances.
- *bw24*: Computerized Tomography (CT Scan) of Chest and Abdomen // Tomografía computarizada de Tórax y Abdomen // 33 appearances.

Expanding this analysis to the development and test set, the results are the same. Some codes appear frequently, while the vast majority appear few times. The top codes of the train set are also the top ones on the development and train set, so there is no difference on the distribution of the codes among the subsets.

The same analysis can be performed to the Diagnosis part, with a ICD-10 codes distribucion shown on Figure 4.2. In this case, the 56% of the codes appear just one time, 73% one or two times and 94% less than 10 times, with 1767 different codes and 500 clinical cases studies.

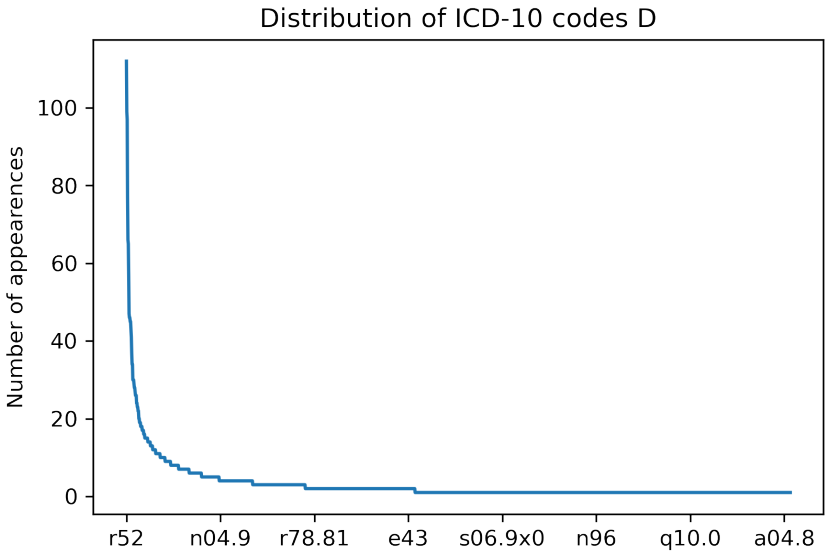


Figure 4.2: Distribution of ICD-10 codes in Diagnosis

In this case, the most common codes are:

- *r52*: Pain, unspecified // Dolor, no especificado // 163 appearances.
- *r69*: Illness, unspecified // Enfermedad NEOM (No Especificado de Otra Manera) // 150 appearances.
- *r50.9*: Fever, unspecified // Fiebre, no especificada // 142 appearances.
- *i10*: Essential (primary) hypertension // Hipertensión esencial (primaria) // 116 appearances.

As before, this situation is really similar in the development and test subsets. Therefore, there is no difference among subsets.

This analysis is quite important, as algorithms will be tested then against a reduced number of codes, corresponding to the most frequent ones, to check the real power of them.

Another analysis performed is to check the words that appear most frequent on the clinical cases studies, after the preprocessing (analysis is performed on the words root). Dividing the analysis in Diagnosis and Procedure:

**Diagnosis**. The most common word as it could be expected is *"patient"*, that appears 547 in the 500 different clinical cases studies of the train set. Nevertheless, it is quite possible that this word does not give any information for any code, so it could be even discarded before entering the algorithm. Now, taking a look at possible words that could have an important weight: renal (263 times), tumor (211), kidney (103), mg/dl (89); or in Spanish, renal (226), tumoración (135), riñón (90), mg/dl (109).

As it can be seen, there is some difference between Spanish and English that could influence the results. Besides, development and test subsets follow the same pattern that train set.

**Procedure**. In the case of Procedure, the results of the analysis are quite similar. *"Patient"* is again the word that appears mostly. Comparing to Diagnosis words, some of them are the same and some other ones are new, as they are mote related to clinical procedures: treatment (703), surgic (256), biopsi (205); or tratamiento (1038), cirugía (244), biopsia (250). Besides, the frequency of words that, a priori, could be important for the algorithms, is higher than in Diagnosis. Finally, as before some differences are encountered probably influencing the results and the same pattern is found in development and test subsets.

To visualize this analysis, the *Token Frequency Distribution* from *Yellowbrick* is used [41]. As its name says, this method is used to plot the frequency distribution of words across any corpora, which can be shown in Figure 4.3 for Diagnosis and 4.4 for Procedure (English left, Spanish right). The values for the frequency are normalized with the number of documents available.

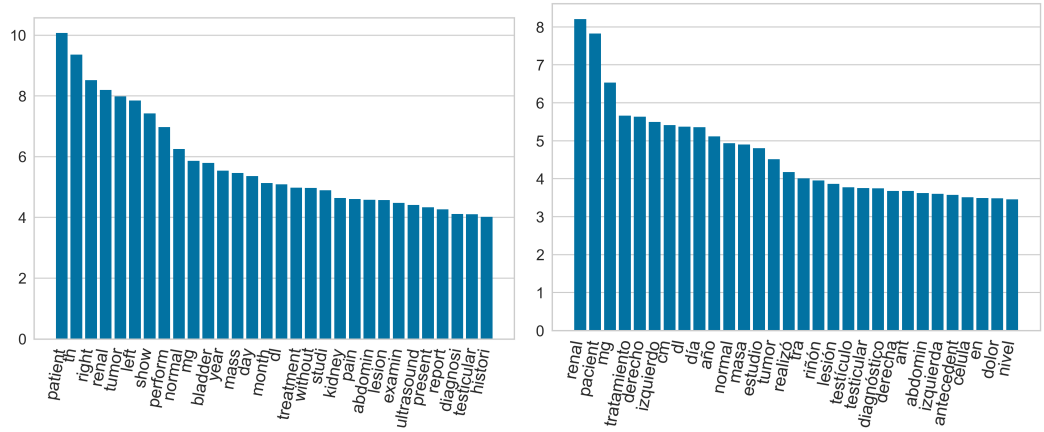


Figure 4.3: Frequency Distribution of Top 30 tokens for Diagnosis (English left, Spanish right)

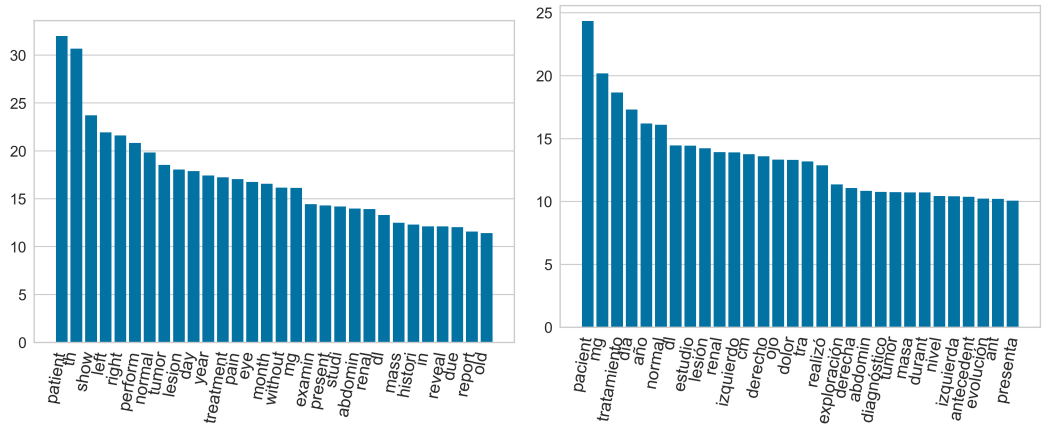


Figure 4.4: Frequency Distribution of Top 30 tokens for Procedure (English left, Spanish right)

Recalling the analysis from the clinical cases studies, and using the information extracted now, it is highlighting how although the Diagnosis clinical cases studies are longer, the frequency of some key words is much lower in Diagnosis than in Procedures.

Finally, UMAP (Uniform Manifold Approximation and Projection) [42] is used to reduce the dimensionality in two dimensions and visualize data in a scatter plot.

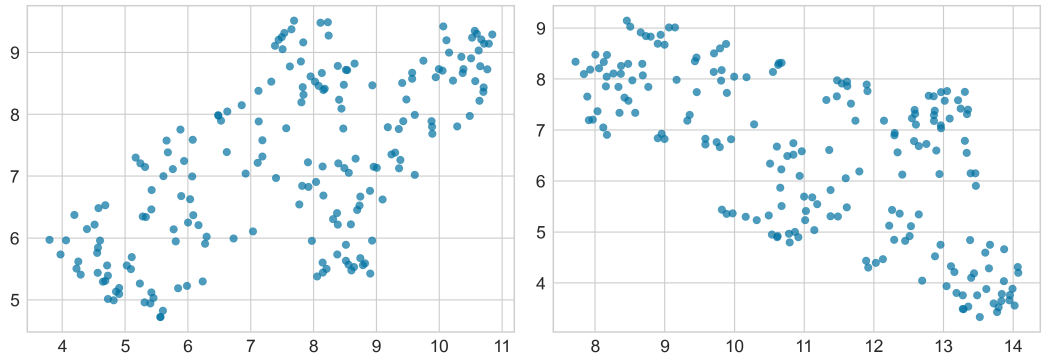


Figure 4.5: UMAP visualization for Diagnosis (English left, Spanish right)

On Figure 4.5, a quite strange situation can be seen. While there is a positive correlation on the English corpus, there is an opposite behaviour with a negative correlation on the Spanish one. Nevertheless, looking at the Procedure UMAP analysis on Figure 4.6, the same behaviour is obtained for both languages. Two main clusters can be obtained, with no correlation.

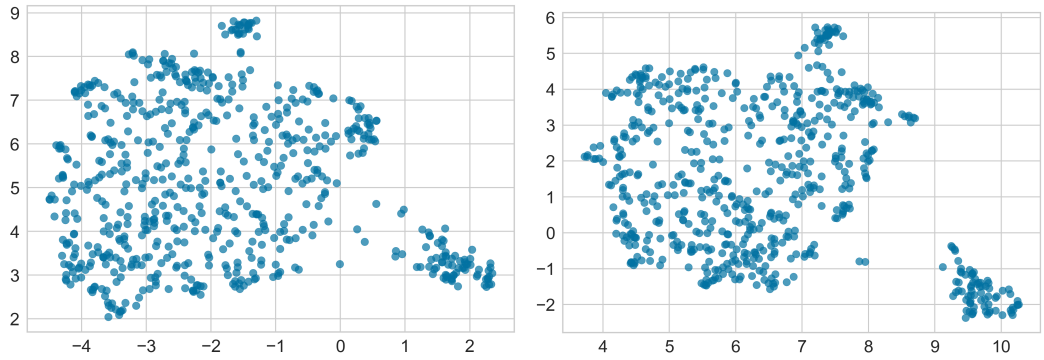


Figure 4.6: UMAP visualization for Procedure (English left, Spanish right)

## 4.3. Results

In this section, the results are going to be presented firstly, followed by a discussion of them. It is important to make a distinction between Diagnosis and Procedure results, whose corpus is different. Besides, as mentioned before, the main issue of this project is the few samples given compared to the enormous number of codes available. Therefore, two different problems have been faced: the complete problem (all labels present on dataset) and a reduced one (most frequent labels).

### 4.3.1. All contained labels problem

In the complete problem, all labels contained in train, development and test subsets have been taking into account for the analysis. This means that any code that is not in the three of them is discarded, to ease the performance of the algorithm; in fact, there is a classification on the competition task with these all contained labels. Therefore, for this problem, algorithms are facing 2558 different ICD-10 codes for Diagnosis and 871 for Procedure.

#### 4.3.1.1 Diagnosis

Starting with English results for Diagnosis, they are shown on Table 4.7. ***Baggings*** is the **top performance algorithm** for all metrics with a mAP of 0.3059 and a micro F1 of 0.5082. This ensemble learning method is able to overcome all models with a high difference. On second place, *Gradient boosting* model is considered to perform accurate comparing with the rest of the algorithms, which are not able to reach even half of its accuracy. In fact, *BERT* does show really low values for both F1 (close to 0) and mAP (0.0044).

| Algor. / Metrics | KNN | D. TREE | R. FOREST | BAGG. | G. BOOST. | L. SVC | BERT |
|---|---|---|---|---|---|---|---|
| **F1_micro** | 0.1283 | 0.1278 | 0.0359 | **0.5082** | 0.4520 | 0.0905 | - |
| **F1_macro** | 0.0123 | 0.0109 | 0.0008 | **0.0668** | 0.0881 | 0.0047 | - |
| **mAP** | 0.0391 | 0.0205 | 0.0205 | **0.3059** | 0.2095 | 0.0437 | 0.0044 |

Table 4.7. RESULTS FOR ALL LABELS D ENGLISH

Now, it is the turn of Spanish labels for Diagnosis results, shown on Table 4.8. As on the English case, the best algorithm is the ensemble method ***Baggings*** with metrics of $mAP = 0.3999$ and $F1\_micro = 0.5983$, followed by *Gradient Boosting* and the rest with really low metrics. Nevertheless, using the score macro F1, *Gradient Boosting* overcomes *Baggings* with a better performance. As before, *BERT* model is not able to perform good with really low values.

| Algor. / Metrics | KNN | D. TREE | R. FOREST | BAGG. | G. BOOST. | L. SVC | BERT |
|---|---|---|---|---|---|---|---|
| **F1_micro** | 0.1363 | 0.1396 | 0.0346 | **0.5983** | 0.5517 | 0.0800 | - |
| **F1_macro** | 0.0134 | 0.0115 | 0.0007 | 0.0873 | **0.1133** | 0.0039 | - |
| **mAP** | 0.0420 | 0.0237 | 0.0213 | **0.3999** | 0.3080 | 0.0417 | 0.0044 |

Table 4.8. RESULTS FOR ALL LABELS D SPANISH

These results from Spanish corpus can be compared with the ones obtained by teams on the competition [2]. Surprisingly, this simple method of TF-IDF plus an ensemble method have better scores than teams as "TeamX" [43], that used a mixed methodology based on machine learning encoders, code-filtering strategies and a final weighted binary cross-entropy model. Nevertheless, a bit far from the top performance (IXA-AAA with $mAP = 0.698$ or IAM with $F1\_micro = 0.748$).

#### 4.3.1.2 Procedure

On the case of the procedure, the same behaviour can be observed. First, English corpora results show that ***Baggings*** is the best algorithm ($mAP = 0.1614$ and $F1\_micro = 0.3382$), followed closely by *Gradient Boosting* specially in the micro F1 score; in fact, for the macro F1 it gets better results (0.0469 vs 0.0292). Again, the other methods are still far from these results. Contrary to Diagnosis, and with lower average results than it, *BERT* model starts to perform somehow better but still far from the other algorithms.

| Algor. / Metrics | KNN | D. TREE | R. FOREST | BAGG. | G. BOOST. | L. SVC | BERT |
|---|---|---|---|---|---|---|---|
| **F1_micro** | 0.1978 | 0.1750 | 0.0745 | **0.3382** | 0.3109 | 0.0944 | 0.0291 |
| **F1_macro** | 0.0137 | 0.0137 | 0.0026 | 0.0292 | **0.0469** | 0.0037 | 0.0006 |
| **mAP** | 0.0681 | 0.0346 | 0.0430 | **0.1614** | 0.1025 | 0.0458 | 0.0192 |

Table 4.9. RESULTS FOR ALL LABELS P ENGLISH

Finally, for the Procedure, the results are available on Table 4.10. The same structure as before is kept, ***Baggings*** highlights as the best algorithm with $mAP = 0.1845$ and $F1\_micro = 0.1845$, while *Gradient Boosting* is the second with the best result on macro F1 score. *BERT* model is again far from the others.

| Algor. / Metrics | KNN | D. TREE | R. FOREST | BAGG. | G. BOOST. | L. SVC | BERT |
|---|---|---|---|---|---|---|---|
| **F1_micro** | 0.1837 | 0.2146 | 0.0615 | **0.3719** | 0.3596 | 0.0946 | 0.0140 |
| **F1_macro** | 0.0120 | 0.0186 | 0.0024 | 0.0339 | **0.0538** | 0.0038 | 0.0011 |
| **mAP** | 0.0621 | 0.0505 | 0.0331 | **0.1845** | 0.1355 | 0.0475 | 0.0045 |

Table 4.10. RESULTS FOR ALL LABELS P SPANISH

Comparing these results with the competition, again they are better than the scores of some teams; however, they are far from the top teams (IAM with $mAP = 0.569$ or FLE $F1\_micro = 0.590$) [2].

### 4.3.2. Most frequent labels problem

Looking at the results and taking into account the amount of classes compared to the train clinical cases studies, a reduction of the problem is performed. To do so, the idea is to reduce the magnitude of the problem by selecting only those codes that appear 10 times or more. Therefore, the performance of the algorithm can be obtained in the presence of a more balance problem. After performing this operation, the problem is reduced to:

1. 500 train clinical studies, 250 for validation and 250 to test for Diagnosis.
2. 438 train clinical reports, 219 for validation and 224 to test for Procedure. In this case, the reduction of the problem lead to the elimination of some reports.
3. 146 clinical codes for Diagnosis and just 45 for Procedure.

#### 4.3.2.1 Diagnosis

Table 4.11 shows the results for the reduced problem in the case of English corpus for Diagnosis. As happened on the complete problem, the best algorithm in terms of performance is ***Baggings*** with $mAP = 0.5208$ and $F1\_micro = 0.7050$, followed closely by *Gradient Boosting*. *BERT* model starts to perform with some accuracy and it overtakes for the first time a classical algorithm (Linear SVC); nevertheless, it is still far from top performance algorithms.

As predicted, the reduction of the problem implies a great improvement on the prediction of the algorithms.

| Algor. / Metrics | KNN | D. TREE | R. FOREST | BAGG. | G. BOOST. | L. SVC | BERT |
|---|---|---|---|---|---|---|---|
| **F1_micro** | 0.2135 | 0.2302 | 0.0187 | **0.7050** | 0.6983 | 0.1756 | 0.1934 |
| **F1_macro** | **0.1363** | 0.1091 | 0.0035 | 0.0547 | 0.0569 | 0.0703 | 0.0439 |
| **MAP** | 0.0934 | 0.0799 | 0.0799 | **0.5208** | 0.5046 | 0.1117 | 0.1264 |

Table 4.11. RESULTS FOR 146 LABELS D ENGLISH

On the Spanish Diagnosis case (Table 4.12), ***Gradient Boosting*** is the best algorithm in terms of the micro F1 ($F1\_micro = 0.8203$) and *Baggings* in terms of mAP ($mAP = 0.6836$); however both algorithms perform equally. *BERT* approach shows better results, but far from *Baggings* or *Gradient Bossting*. The results for this case are quite high and quite interesting, considering that two algorithms are performing really accurate.

| Algor. / Metrics | KNN | D. TREE | R. FOREST | BAGG. | G. BOOST. | L. SVC | BERT |
|---|---|---|---|---|---|---|---|
| **F1_micro** | 0.2221 | 0.2892 | 0.0329 | 0.8192 | **0.8203** | 0.1555 | 0.2034 |
| **F1_macro** | 0.1376 | 0.1359 | 0.0067 | 0.7023 | **0.7301** | 0.0562 | 0.0470 |
| **MAP** | 0.0966 | 0.1109 | 0.0503 | **0.6836** | 0.6816 | 0.1073 | 0.1160 |

Table 4.12. RESULTS FOR 146 LABELS D SPANISH

In this case, no comparison with the competition can be performed as this reduction problem is an author's decision to check the performance when having better conditions for training.

#### 4.3.2.2 Procedure

For the reduced problem related to English corpus in Procedure, results are shown in Table 4.13. ***Gradient Boosting*** is the best algorithm in the case of micro F1, with

$F1\_micro$ = 0.5386; and *Baggings* in terms of *mAP*, with $mAP$ = 0.3250. The other methods are quite far from these two algorithms, incuding *BERT*.

| Algor. / Metrics | KNN | D. TREE | R. FOREST | BAGG. | G. BOOST. | L. SVC | BERT |
|---|---|---|---|---|---|---|---|
| **F1_micro** | 0.3365 | 0.3308 | 0.0939 | 0.5268 | **0.5386** | 0.1917 | 0.3242 |
| **F1_macro** | 0.2011 | 0.1961 | 0.0311 | 0.3476 | **0.4061** | 0.0707 | 0.0912 |
| **MAP** | 0.1653 | 0.1367 | 0.0876 | **0.3250** | 0.3219 | 0.1259 | 0.1939 |

Table 4.13. RESULTS FOR 45 LABELS P ENGLISH

Finally, for the Spanish corpus in the reduced problem for Procedure, results can be seen on Table 4.14. As before, ***Gradient Boosting*** wins in terms of the micro F1 ($F1\_micro$ = 0.6278); while *Baggings* is the first in terms of *mAP* ($mAP$ = 0.4232). In this case, *BERT* overtakes another position by performing better than *KNN*. These results are lower than for Diagnosis, but improve significantly compared with the reduced problem.

| Algor. / Metrics | KNN | D. TREE | R. FOREST | BAGG. | G. BOOST. | L. SVC | BERT |
|---|---|---|---|---|---|---|---|
| **F1_micro** | 0.3182 | 0.4324 | 0.0849 | 0.6209 | **0.6278** | 0.1930 | 0.3289 |
| **F1_macro** | 0.1682 | 0.3056 | 0.0294 | 0.4244 | **0.4709** | 0.0738 | 0.1179 |
| **MAP** | 0.1541 | 0.2122 | 0.0828 | **0.4232** | 0.4220 | 0.1313 | 0.1889 |

Table 4.14. RESULTS FOR 45 LABELS P SPANISH

### 4.3.3. Optuna optimization

For the optuna optimization, simple suggestions are proposed due to the complex problem and the expensive running time. This hyper-parameter optimization is only proposed for algorithms with the best results:

- **Bagging**:
  - *Number of estimators*: number of base estimators in the ensemble. Suggestions: [2-50]
  - *Bootstrap*: whether samples are drawn with replacement. Suggestions: *True* (replacement) or *False* (no replacement).
- **Gradient Boosting**:
  - *Number of estimators*: The number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance. Suggestions: [2-50]

- *Maximum depth*: The maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. Suggestions: [2-40]
- *Loss*: Loss function to be optimized. Suggestions: *Exponential* or *Deviance* (logistic regression).
- *Criterion*: The function to measure the quality of a split. Suggestions: $Friedman\_mse$ (mean squared error with improvement score by Friedman, *mse* (mean squared error) or *mae* (mean absolute error).

Although these simple suggestions are proposed for Optuna, results are improved for both algorithms in all cases, shown in Tables 4.15, 4.16, 4.17 and 4.18 . It is important to mention that F1 macro is not included as the most important ones are the micro and mAP.

- **All contained labels**

| Algor. / Metrics | **BAGG.** | **G. BOOST.** |
|---|---|---|
| **F1_micro** | 0.5175 | 0.4895 |
| **MAP** | 0.3156 | 0.2240 |

| Algor. / Metrics | **BAGG.** | **G. BOOST.** |
|---|---|---|
| **F1_micro** | 0.6028 | 0.5793 |
| **MAP** | 0.4088 | 0.3214 |

Table 4.15. OPTUNA RESULTS FOR DIAGNOSIS ENGLISH (LEFT) AND SPANISH (RIGHT)

| Algor. / Metrics | **BAGG.** | **G. BOOST.** |
|---|---|---|
| **F1_micro** | 0.3696 | 0.3394 |
| **MAP** | 0.1719 | 0.1182 |

| Algor. / Metrics | **BAGG.** | **G. BOOST.** |
|---|---|---|
| **F1_micro** | 0.3972 | 0.3815 |
| **MAP** | 0.1987 | 0.1480 |

Table 4.16. OPTUNA RESULTS FOR PROCEDURE ENGLISH (LEFT) AND SPANISH (RIGHT)

- **Most-frequent labels**

| Algor. / Metrics | **BAGG.** | **G. BOOST.** |
|---|---|---|
| **F1_micro** | 0.7309 | 0.7213 |
| **MAP** | 0.5386 | 0.5202 |

| Algor. / Metrics | **BAGG.** | **G. BOOST.** |
|---|---|---|
| **F1_micro** | 0.8325 | 0.8451 |
| **MAP** | 0.7006 | 0.6994 |

Table 4.17. OPTUNA RESULTS FOR DIAGNOSIS ENGLISH (LEFT) AND SPANISH (RIGHT)

| Algor. / Metrics | BAGG. | G. BOOST. |
|---|---|---|
| **F1_micro** | 0.5780 | 0.5673 |
| **MAP** | 0.3763 | 0.3617 |

| Algor. / Metrics | BAGG. | G. BOOST. |
|---|---|---|
| **F1_micro** | 0.6391 | 0.6422 |
| **MAP** | 0.4365 | 0.4349 |

Table 4.18. OPTUNA RESULTS FOR PROCEDURE ENGLISH (LEFT) AND SPANISH (RIGHT)

### 4.3.4. Discussion

Looking at the results as overall, two algorithms stand out over the others: Baggings and Gradient Boosting. They have results three and even four times more accurate than the rest. In fact, contrary to what it was expected, they even perform better than the BERT-model.

Focusing on the complete problem, a big difference between metrics for Diagnosis and Procedure can be observed, getting better results for the first one: diagnosis cases are easier to distinguish than procedure ones. In fact, it is curious how Diagnosis with 2,558 different codes have better results than Procedure with 871. One explanation of this fact is the UMAP analysis, that showed a correlation for Diagnosis corpus, while no correlation for Procedure corpus.

Making an extra zoom, there is a difference between English and Spanish results for both Diagnosis and Procedure. Spanish corpus classification has better results than English. This could have two different explanations: firstly, and as mentioned before, Spanish corpus has more words per document than English, providing probably more information to the TF-IDF; secondly, the English dataset is a machined translation of the Spanish one, possibly influencing the expression in clinical reports and therefore lowering algorithms performance. This behaviour can be better observed in Figure 4.7, together with the differences in the algorithms performance.

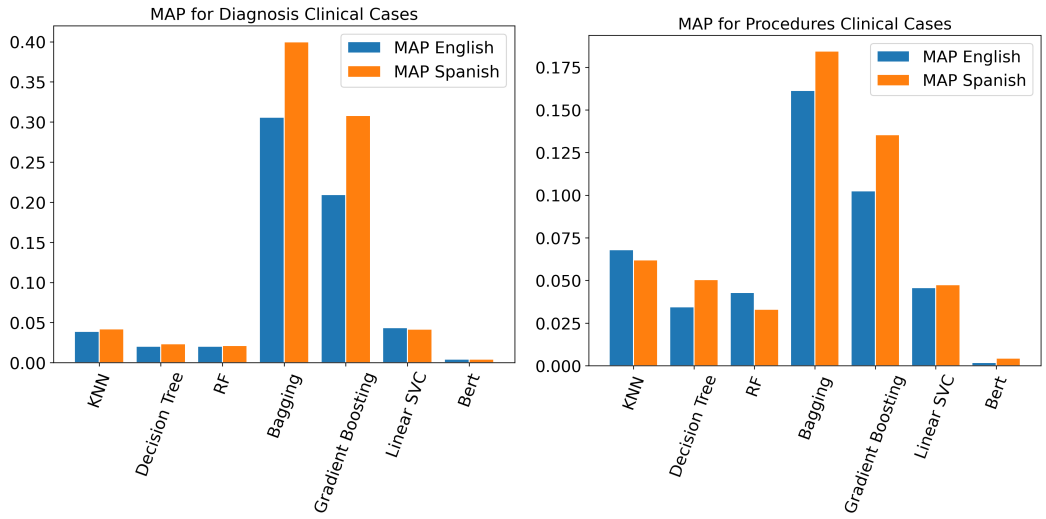


Figure 4.7: MAP score as a function of language and algorithm

This same behaviour can be observed in the reduced problem (most frequent labels problem), on Figure 4.8. Diagnosis results are more accurate than Procedure (only for Gradient Boosting and Bagging model but not for the others), and Spanish ones are also more accurate than English ones. In this case, the results obtained for the reduced problem show that Baggings and Gradient Boosting are two algorithm that performs quite well with TF-IDF as long as the training set is balanced and have enough information for them to learn the basis.

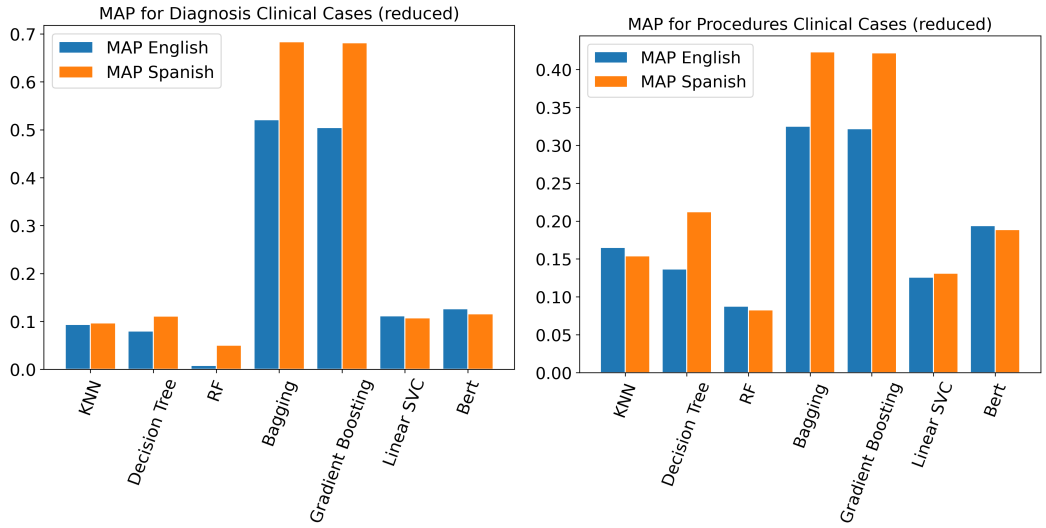


Figure 4.8: MAP score as a function of language and algorithm (most frequent labels)

Making a comparison between classical algorithms and BERT model, BERT methodology perform way worst in this specific case, contrary to the state-of-the-art [12]. Nevertheless, this could be explained easily by the reduced number of clinical cases

reports available for training and the high imbalance problem faced. In fact, balancing the problem with the reduction of labels boost the performance of BERT compared to classical algorithms.

Finally, using Optuna, values are improved in all cases for both Baggings and Gradient Boosting algorithm. This improvement can be seen in Figures 4.9 and 4.10, where the difference Optuna gives is greater for Gradient Boosting in English Procedure, than for others as Baggings in English Diagnosis.

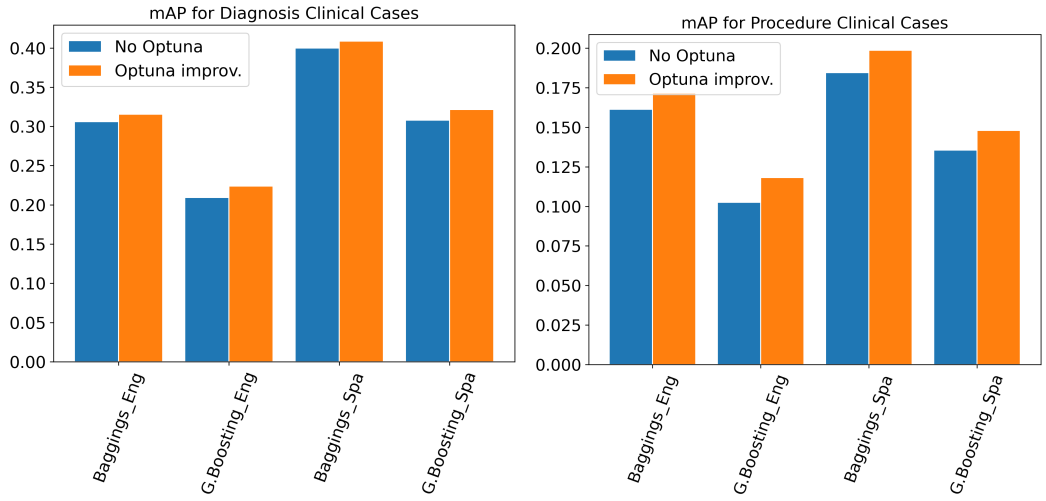


Figure 4.9: MAP score: Optuna comparison

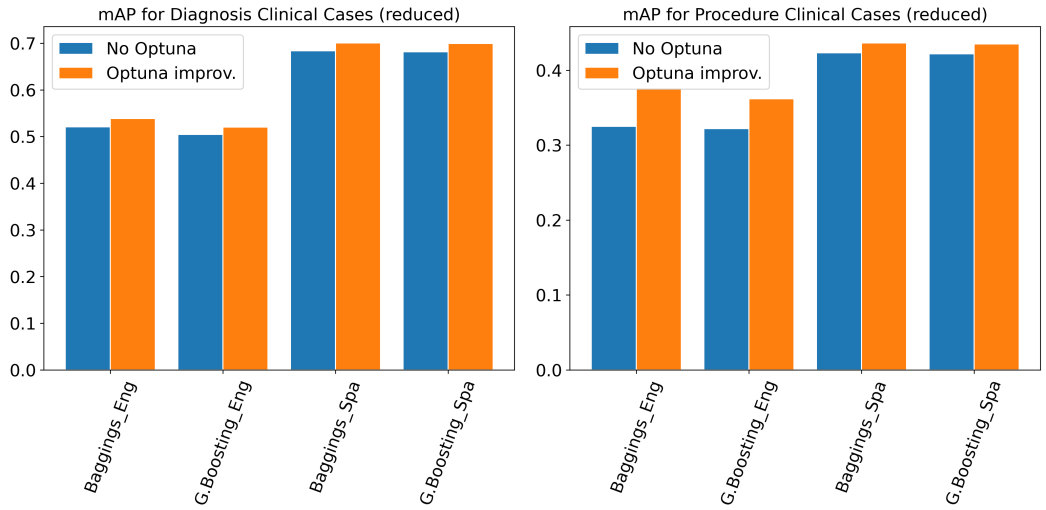


Figure 4.10: MAP score: Optuna comparison (most frequent labels)

Therefore, final values are slightly better by just adding a hyper-parameter tuning of classical algorithms.

# Chapter 5

# Project Management

## 5.1. Project Infrastructure

Python has been the selected language to run all experiments. In fact, to ease the debugging and explanations, notebooks have been used. For the classic algorithms, Jupyter is used; whereas for BERT, Google Colab is the best option thanks to its cloud running and storage, and its free GPU and TPU usage. All developed code can be found at GitHub repository: https://github.com/ignasiser97/ICD-10_Classification.

## 5.2. Planning

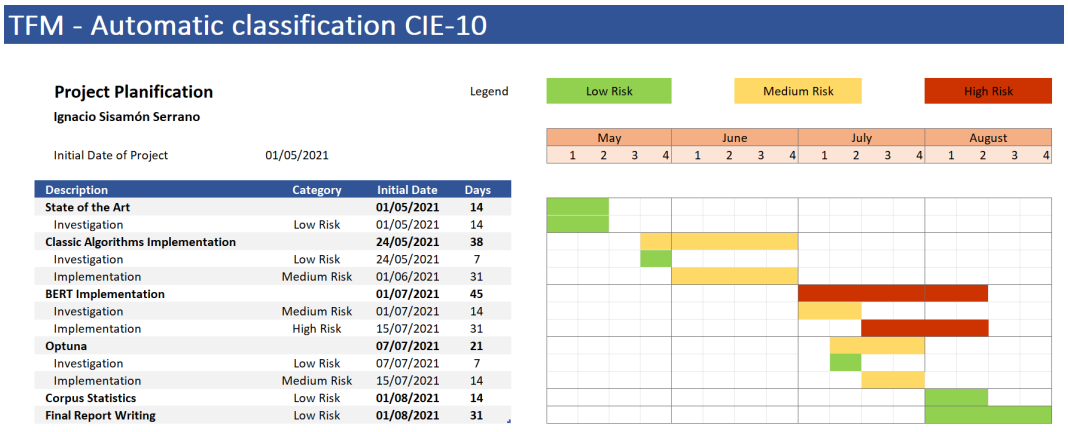


Figure 5.1: Project planning

# Chapter 6

# Conclusion

After analyzing the results, several conclusions are obtained:

- NLP methodology is a really powerful tool to classify clinical case studies with the ICD-10 codes, without the required training an expert needs. Nevertheless, this methodology needs still some improvements to perform with better results.
- The analysis of the corpus of this project reveals interesting information, as Procedure and Spanish uses more words than Diagnosis and English respectively. Besides, the distribution of the ICD-10 codes in the dataset shows an imbalanced problem with most of codes appearing just once or twice, influencing negatively the algorithms performance. Furthermore, reducing the dimensionality of the data into 2D, an opposite behaviour between Spanish (negative correlation) and English (positive corelation) Diagnosis is observed, while no correlation and clusters are observed equally on both languages for Procedure.
- Taking a look at the results, the methodology based on TF-IDF for the vectorization of words is, a priori, the best solution for this specific project. The combination of this methodology, together with classical algorithms, has obtained the best accuracy results, compared with BERT model. This behaviour has been contrary to expected results, possibly due to the imbalanced problem faced and the lack of enough training clinical reports.
- Focusing on the classical machine learning algorithms, Bagging and Gradient Boosting classifiers are the top performance ones, with really good results and far from the rest. These two algorithms are ensemble learning methods that use the combination of different decision trees and boosting them with a final hyper-parameter tuning using *Optuna*, results are improved even more.

- Therefore, the best result obtained (focused on the mAP as the competition dictates) is $mAP = 0.3146$ for Diagnosis in English, and $mAP = 0.4088$ in Spanish. On the other hand, for Procedure, results are slightly slower $mAP = 0.1719$ for English and $mAP = 0.1987$ for Spanish. The main possible reason on language differences is that Spanish uses more words than English, influencing the TF-IDF vectorization due to the reduced number of clinical studies and therefore varying the results. This is where BERT was expected to overtake this problem with its pre-trained model, however it did not accomplish it. On the case of Diagnosis and Procedure differences, the Diagnosis correlation observed with the UMAP reduction versus the no-correlation in the Procedure may explain these differences.

- Results obtained in this project compared to the competition ones show the power of TF-IDF vectorization and classical predictive algorithms, overtaking teams that used more complex structures as a combination of machine learning encoders, code-filtering strategies and a final weighted binary cross-entropy model concatenated.

- Leaving aside the competition and to explore the real power of TF-IDF with classical algorithms, a reduction of the problem is perform by removing labels appearing few times. This overtakes the imbalanced problem simulating, at the same time, a bigger train dataset with multiple appearances of every label. In this case, results are much higher compared to the complete problem: $mAP = 0.5386$ for English Diagnosis, $mAP = 0.7006$ for Spanish; and $mAP = 0.3763$ English Procedure and $mAP = 0.4365$ for Spanish.

### 6.1. Future of the project

The future of the project is quite interesting, taking into account the good performance of classical machine learning algorithms. Focusing on the vectorization of words, which is key in this project, two different approaches could be taken.

Firstly, and following the state-of-the-art, the BERT pre-trained model has proved its well way of working in other projects; however for this specific project, the following training and classification does not perform pretty accurate. Therefore, a good solution would be to substitute the TF-IDF with the BERT pre-trained model, and then tune it with the classical algorithms. This combination, followed by an optuna hyper-parameter optimization, could improve the results obtained in this project.

Secondly, a good idea could be to substitute again the TF-IDF by specific clinical word embedding already pre-trained with clinical words. Therefore, the final results would be again a pre-trained model for the vectorization of words, classical algorithms and optuna optimization.

With these two solutions, results could be probably improved.

# Bibliography

[1] Centers for Disease Control and Prevention. (2012, May 18). Principles of epidemiology. Centers for Disease Control and Prevention. Retrieved 4$^{th}$ September 2021, https://www.cdc.gov/csels/dsepd/ss1978/lesson1/section5.html.

[2] Miranda-Escalada, A., Gonzalez-Agirre, A., Armengol-Estape, J., Krallinger, M. (s/f). Overview of automatic clinical coding: annotations, guidelines, and solutions for non-English clinical cases at CodiEsp track of CLEF eHealth 2020.

[3] Stanfill, M. H., Williams, M., Fenton, S. H., Jenders, R. A., Hersh, W. R. (2010). A systematic literature review of automated clinical coding and classification systems. Journal of the American Medical Informatics Association: JAMIA, 17(6), 646–651.

[4] Classification of Diseases (ICD). (s/f). Who.int. Retrieved 2$^{nd}$ August 2021, https://www.who.int/standards/classifications/classification-of-diseases.

[5] World Health Organization. (2015). The international statistical classification of diseases and related health problems, ICD-10 V1 (10a ed.). World Health Organization.

[6] ICD-10 Version:2019. (s/f). Who.int. Retrieved 3$^{rd}$ August 2021, https://icd.who.int/browse10/2019/en.

[7] BOE-A-2018-16673. (s/f). Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. Retrieved 3$^{rd}$ August 2021, https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673.

[8] La nueva LOPD regula la protección de datos en investigación y ensayos clínicos. (s/f). Medicosypacientes.com. Retrieved 3$^{rd}$ August 2021, http://www.medicosypacientes.com/articulo/la-nueva-lopd-regula-la-proteccion-de-datos-en-investigacion-y-ensayos-cli

[9] Bounaama, Rabia Mohammed El Amine, Abderrahim. (2018). Tlemcen University at CELF eHealth 2018 Team techno: Multilingual Information Extraction - ICD10 coding ; CLEF 2018.

[10] Pérez, J., Pérez, A., Casillas, A., Gojenola, K. (2018). Cardiology record multi-label classification using latent Dirichlet allocation. Computer Methods and Programs in Biomedicine, 164, 111–119.

[11] Amin, Saadullah Neumann, Günter Dunfield, Katherine Vechkaeva, Anna Chapman, Kathryn Wixted, Morgan. (2019). MLT-DFKI at CLEF eHealth 2019: Multi-label Classification of ICD-10 Codes with BERT.

[12] Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing. (s/f). Googleblog.com. Retrieved 5$^{th}$ August 2021, https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html.

[13] Han, J., Kamber, M., Pei, J. (2012). Getting to know your data. En Data Mining (pp. 39–82). Elsevier.

[14] Cossin, S., Jouhet, V. (s/f). IAM at CLEF EHEALTH 2020: CONCEPT ANNOTATION in SPANISH ELECTRONIC HEALTH RECORDS. Ceur-ws.org. Retrieved 5$^{th}$ August 2021, http://ceur-ws.org/Vol-2696/paper_198.pdf.

[15] Costa, J., Lopes, I., Carreiro, A., Ribeiro, D., Soares, C. (2020). Fraunhofer AICOS at CLEF eHealth 2020 Task 1: Clinical Code Extraction From Textual Data Using Fine-Tuned BERT Models. CLEF.

[16] IBM Cloud Education. (n.d.). What is a knowledge graph? IBM. Retrieved 6$^{th}$ August, 2021, https://www.ibm.com/cloud/learn/knowledge-graph.

[17] García-Santa, N., Cetina, K. (2020). FLE at CLEF eHealth 2020: Text Mining and Semantic Knowledge for Automated Clinical Encoding. CLEF.

[18] P. Bafna, D. Pramod and A. Vaidya, "Document clustering: TF-IDF approach," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016, pp. 61-66, doi: 10.1109/ICEEOT.2016.7754750.

[19] H. P. Luhn, "A Statistical Approach to Mechanized Encoding and Searching of Literary Information," in IBM Journal of Research and Development, vol. 1, no. 4, pp. 309-317, Oct. 1957, doi: 10.1147/rd.14.0309.

[20] Sparck Jones, K. (1972), "A statistical interpretation of term specificity and its application in retrieval", Journal of Documentation, Vol. 28 No. 1, pp. 11-21. https://doi.org/10.1108/eb026526.

[21] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of deep bidirectional Transformers for language understanding. En arXiv [cs.CL]. http://arxiv.org/abs/1810.04805.

[22] word2vec. (s/f). code.google.com. Retrieved 10th August 2021, https://code.google.com/archive/p/word2vec/.

[23] Pennington, J., Socher, R., Manning, C. (2014). Glove: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[24] Zhang, M.-L., Zhou, Z.-H. (2007). ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition, 40(7), 2038–2048.

[25] Fix, Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties (PDF) (Report). USAF School of Aviation Medicine, Randolph Field, Texas.

[26] sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.24.2 documentation. (s/f). Scikit-learn.org. Retrieved 12th August 2021, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[27] 1.10. Decision Trees — scikit-learn 0.24.2 documentation. (s/f). Scikit-learn.org. Retrieved 12th August 2021, https://scikit-learn.org/stable/modules/tree.html.

[28] Machine learning decision tree classification algorithm - javatpoint. www.javatpoint.com. (n.d.). Retrieved 12th August, 2021, https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm.

[29] Ho, T. K. (1995). Random decision forests. Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1, 278.

[30] sklearn.ensemble.BaggingClassifier — scikit-learn 0.24.2 documentation. (s/f). Scikit-learn.org. Retrieved 12th August 2021, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html.

[31] Bagging (bootstrap aggregation). (2020, April 28). Corporatefinanceinstitute.Com. https://corporatefinanceinstitute.com/resources/knowledge/other/bagging-bootstrap-aggregation/.

[32] sklearn.ensemble.GradientBoostingClassifier — scikit-learn 0.24.2 documentation. (s/f). Scikit-learn.org. Retrieved 12th August 2021,

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html.

[33] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., Steinberg, D. (2008). Top 10 algorithms in data mining. Knowledge and Information Systems, 14(1), 1–37.

[34] 1.4. Support Vector Machines — scikit-learn 0.24.2 documentation. (s/f). Scikit-learn.org. Retrieved 12$^{th}$ August 2021, https://scikit-learn.org/stable/modules/svm.html.

[35] Optuna: A hyperparameter optimization framework — Optuna 2.9.1 documentation. (s/f). Readthedocs.io. Retrieved 12$^{th}$ August 2021, https://optuna.readthedocs.io/en/stable/.

[36] López, F. (2021, febrero 7). OPTUNA: A Flexible, Efficient and Scalable Hyperparameter Optimization Framework. Towards Data Science. Retrieved 12$^{th}$ August 2021, https://towardsdatascience.com/optuna-a-flexible-efficient-and-scalable-hyperparameter-optimization-framew

[37] Pretrained models. (s/f). Huggingface.co. Retrieved 14$^{th}$ August 2021, https://huggingface.co/transformers/pretrained_models.html.

[38] Brownlee, J. (2018, December 2). A gentle introduction to dropout for regularizing deep neural networks. Machinelearningmastery.Com. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/.

[39] Shmueli, B. (2019, julio 2). Multi-class metrics made simple, part I: Precision and recall. Towards Data Science. Retrieved 15$^{th}$ August 2021 https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2.

[40] Soares, F., Krallinger, M. (2019). BSC participation in the WMT translation of biomedical abstracts. Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2), 175–178.

[41] Yellowbrick: Machine Learning Visualization — Yellowbrick v1.3.post1 documentation. (s/f). Scikit-yb.org. Retrieved 16$^{th}$ August 2021, https://www.scikit-yb.org/en/latest/index.html.

[42] UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction — umap 0.5 documentation. (s/f). Readthedocs.io. Retrieved 16$^{th}$ August 2021, https://umap-learn.readthedocs.io/en/latest/index.html.

[43] Tagawa, Y., Nakano, N., Ozaki, R., Taniguchi, T., Ohkuma, T. (n.d.). TeamX at CLEF eHealth 2020: ICD coding with N-gram encoder and code-filtering strategy. Ceur-Ws.Org. Retrieved 16$^{th}$ August 2021, http://ceur-ws.org/Vol-2696/paper_115.pdf.