MARIANA YUKARI NOGUTI

COMPARISON OF NATURAL LANGUAGE PROCESSING ALGORITHMS APPLIED TO SMALL SUPERVISED DATASETS IN THE LEGAL DOMAIN

(*pre-defense version, compiled at July 8, 2022*)

Dissertation Proposal presented to the M.Sc. Program in Informatics, Exact Sciences Sector, of the Federal University of Paraná, Brazil.

Field: *Computer Science*.

Advisor: Luiz Eduardo S. Oliveira.

Co-advisor: Eduardo Vellasques.

CURITIBA PR - BRAZIL

2022

RESUMO

O presente trabalho procura investigar a performance de técnicas de *transfer learning* em conjunto com técnicas de *data augmentation* e diferentes algoritmos de aprendizagem supervisionada e semi-supervisionada na classificação de textos da área legal em tópicos pré-definidos. A intenção é investigar as melhores técnicas capazes de otimizar a performance na aludida tarefa utilizando uma base de dados rotulados relativamente pequena e grandes quantidades de dados não rotulados. Mais especificamente, serão utilizados como teste dados de atendimentos ao público realizados pelo Ministério Público do Estado do Paraná (MPPR), com o objetivo de classificar as descrições dos atendimentos em um dos assuntos listados pela instituição e automatizar a tarefa no sistema de registros. Como os integrantes da instituição possuem diversas demandas, não é possível avaliar um grande volume de dados, de modo que a otimização de classificadores com utilização de poucos dados é uma tarefa relevante para o desenvolvimento do produto final. Além disso, considerando o vocabulário particular utilizado cotidianamente pelo MPPR, pretende-se avaliar o impacto da realização de *fine-tuning* em modelos de linguagem pré-existentes em português na performance do classificador.

Para a presente pesquisa foi obtida uma base rotulada contendo 6.500 observações com o objetivo de classificar textos curtos em 50 diferentes assuntos relacionados às áreas de atuação do MPPR. Também foram disponibilizados grandes volumes de observações não rotuladas para compor uma base semi-supervisionada, bem como uma base contendo mais de um milhão de registros internos, utilizada no treinamento de diferentes modelos de linguagem.

Os resultados da pesquisa demonstram que, no caso da aprendizagem supervisionada através de classificadores lineares como a Regressão Logística e o SVM e ensembles como o Gradient Boosting e Random Forest, a melhor performance é observada utilizando *embeddings* extraídos pela técnica word2vec quando comparado com o modelo BERT. Este último demonstra performance superior quando utiliza como vantagem a arquitetura do próprio modelo como classificador, tendo superado os modelos anteriores neste sentido. O melhor resultado obtido indica que o uso conjunto do modelo de linguagem BERT ajustado ao vocabulário jurídico, técnicas específicas de aprendizado semi-supervisionado e *data augmentation* obtém melhor performance quando comparado aos demais modelos, com obtenção de acurácia de 80,7% na predição de 50 classes.

Palavras-chave: Processamento de Linguagem Natural. Aprendizado Semi-Supervisionada. Aprendizado por Transferência. *Data Augmentation*.

ABSTRACT

This research seeks to investigate the performance of transfer learning techniques in conjunction with data augmentation and different supervised and semi-supervised learning algorithms in the classification of texts in the legal area on predefined topics. The intention is to investigate how the recent advances in Natural Language Processing (NLP) can contribute to tackle such type of problem (where amount of labelled data is low but there is a large volume of unlabelled/domain-specific data). More specifically, we will use the records of demands to the Public Prosecutor's Office of the State of Paraná in order to classify the descriptions in one of the subjects listed by the institution and automate the task in the records system. Considering that the members of the institution have several demands, it is not possible to evaluate a large volume of data, so that the optimization of classifiers in low regime data is a relevant task for the development of the final product. In addition, considering the specificity of the vocabulary used by the MPPR, it is intended to assess the impact of fine-tuning pre-existing Portuguese language models on the classifier's performance.

For this investigation, a labeled dataset was obtained containing 6,500 observations in order to classify texts on 50 different categories related to the areas of activity of the MPPR. Large volumes of unlabeled observations were also made available to compose a semi-supervised dataset, as well as a dataset containing more than one million internal records, used in the training of different language models.

Our results demonstrate that, in the case of supervised learning through linear classifiers such as Logistic Regression and SVM and boosted trees such as Gradient Boosting and Random Forest, better performance is observed using embeddings extracted by the word2vec technique when compared to the BERT model. The latter demonstrates superior performance when using the architecture of the model itself as a classifier to its advantage, having surpassed the previous models in this sense. The best result obtained indicates that the joint use of the BERT language model fine-tuned to the legal vocabulary, specific techniques of semi-supervised learning and data augmentation presents better performance when compared to all previous models, having obtained an accuracy of 80.7% in the prediction of 50 classes.

Keywords: Natural Language Processing. Semi-Supervised Learning. Transfer Learning. Data Augmentation.

LIST OF FIGURES

2.1 CBOW and Skip-Gram tasks for word2vec 18
2.2 Encoder structure inside the Transformer architecture 20
2.3 Attention mechanism 21
2.4 BERT input example 23
2.5 BERT classification example 24
2.6 SVM structure in two dimensional space 26
2.7 Decision tree structure 27
2.8 TSA schedules 31
2.9 UDA training strategy 32
2.10 Example of confusion matrix 32

5.1 Datasets scheme 50
5.2 Hierarchical structure of subjects in labeled data 51
5.3 Confusion matrix produced by the UDA model 60
5.4 Dumbbell chart of comparative performance between humans and the classifier 63

LIST OF TABLES

5.1 Breakdown of manually verified examples in the demands dataset 52
5.2 Evaluation of BERT using different layers in the validation set 53
5.3 Evaluation of BERT using different learning rates in the validation set 54
5.4 Results of supervised classifiers in the test set 56
5.5 Results of BERT classifiers in the test set 57
5.6 Results of self-training and co-training in the test set 58
5.7 Results of UDA with generic BERT in the test set 59
5.8 Per-class metrics for the UDA model 61
5.9 Global metrics for the best classifiers 62

A.1 Best parameters obtained with grid search and logistic regression 72
A.2 Best parameters obtained with grid search and SVM 73
A.3 Best parameters obtained with grid search and random forest 73
A.4 Best parameters obtained with grid search and gradient boosting 74

LIST OF EQUATIONS

2.1 Term frequency calculation 16
2.2 Inverse document frequency calculation 16
2.3 Term frequency-inverse document frequency calculation 16
2.4 Language Model 19
2.5 Positional Encoding Equations 20
2.6 Attention Mechanism 21
2.7 Compatibility Score 21
2.8 Scaled Dot-Product Attention 22
2.9 Logistic Regression Equation 25
2.10 Linear Equation 25
2.11 Softmax Function 25
2.12 Precision per class 33
2.13 Recall per class 34
2.14 F1 Score per class 34
2.15 Accuracy formula 34
2.16 Flat Macro Precision 35
2.17 Flat Macro Recall 35
2.18 Flat Macro F1 Score 35
2.19 Hierarchical Precision 36
2.20 Hierarchical Recall 36
2.21 Hierarchical F1 Score 36

LIST OF ACRONYMS

BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag-of-Words
BPE	Byte Pair Encoding
CBOW	Continuous Bag-of-Words
CNJ	*Conselho Nacional de Justiça*
CNN	Convolutional Neural Network
EDA	Easy Data Augmentation
GB	Gradient Boosting
GPU	Graphics Processing Unit
LDA	Latent Dirichlet Allocation
LM	Language Model
LR	Logistic Regression
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
MLM	Masked Language Modeling
MPPR	*Ministério Público do Estado do Paraná*
NLP	Natural Language Processing
NSP	Next Sentence Prediction
OOV	Out-of-Vocabulary
RF	Random Forest
RNN	Recurrent Neural Network
SSL	Semi-Supervised Learning
SVM	Support Vector Machines
TF-IDF	Term Frequency–Inverse Document Frequency
TPU	Tensor Processing Unit
TSA	Training Signal Annealing
UDA	Unsupervised Data Augmentation

CONTENTS

1 Introduction **10**

1.1 Motivation 10
1.2 Challenges 11
1.3 Objectives 12
1.4 Contributions 12
1.5 Organization of the thesis 13

2 Theoretical Background **14**

2.1 Tokenization 14
2.2 Text normalization 15
2.3 Word Filtering 15
2.4 Text Representation 15
2.4.1 Bag-of-Words 15
2.4.2 TF-IDF 16
2.4.3 Embeddings 17
2.5 Transfer learning 18
2.6 Language models 19
2.6.1 Transformers 19
2.6.2 BERT 22
2.7 Logistic regression 25
2.8 Support vector machines 26
2.9 Decision trees 27
2.10 Ensemble learning 27
2.10.1 Bagging 28
2.10.2 Boosting 28
2.11 Semi-supervised learning 28
2.11.1 Self-training 29
2.11.2 Co-training 29
2.11.3 Unsupervised data augmentation 30
2.12 Evaluation metrics 32
2.12.1 Per-class Metrics 33
2.12.2 Global Metrics 34

3 Related Work **37**
3.1 Text classification . . . 37
3.2 NLP in the legal domain . . . 39

4 Methodology **42**
4.1 Preprocessing . . . 42
4.1.1 Word2vec preprocessing . . . 42
4.1.2 BERT preprocessing . . . 42
4.2 Feature extraction . . . 43
4.2.1 Generic word2vec . . . 43
4.2.2 Specialized word2vec . . . 43
4.2.3 Generic BERT . . . 43
4.2.4 Specialized BERT . . . 44
4.3 Classifiers . . . 44
4.3.1 Supervised learning . . . 44
4.3.2 Semi-supervised learning . . . 45
4.4 Evaluation metrics . . . 47
4.4.1 Per-class Metrics . . . 47
4.4.2 Global Metrics . . . 47

5 Experiments **49**
5.1 Datasets . . . 49
5.1.1 Labeled data . . . 49
5.1.2 Unlabeled data . . . 51
5.2 Feature extraction . . . 53
5.3 Supervised learning . . . 55
5.4 Semi-supervised learning . . . 57
5.4.1 Self-training and co-training . . . 57
5.4.2 UDA . . . 59
5.5 Detailed Analysis: UDA with Back Translation and log TSA . . . 60

6 Conclusions **64**

REFERENCES **66**

APPENDIX A – HYPERPARAMETER INVESTIGATION **72**
A.1 Logistic regression . . . 72
A.2 SVM . . . 72
A.3 Random forest . . . 73
A.4 Gradient boosting . . . 74

1 INTRODUCTION

Text classification is an important and widely used task in Natural Language Processing (NLP) that consists of predicting labels associated with textual content. Among its most popular applications, we can mention sentiment analysis, tagging, spam detection, and topic classification. To obtain models capable of performing the aforementioned tasks, the commonly used methodology is divided into three main stages, namely the preprocessing of texts, the application of feature extraction, and the construction of classifiers.

In general, due to the great variability of the vocabulary found in the corpora of classification tasks, a significant amount of labeled examples is necessary to extract a representative sample of each class and distinguish them. This requirement for labeled data has a propensity to grow exponentially with the increase in the number of classes to be predicted, making a project difficult or even unfeasible due to the costs involved in the construction of an adequate database.

Some suggestions to work around the problem can be found in recently developed techniques in NLP, such as transfer learning, semi-supervised learning, and data augmentation. The first technique has been shown to be consistent in improving the performance of several NLP tasks while decreasing the need for a large amount of labeled data for learning. As for the second technique mentioned, the semi-supervised learning models enable the optimization of supervised models with unlabeled databases as a form of complementary information. Lastly, data augmentation is a technique mostly used in images, but latest developments demonstrate that it can also be applied to NLP. It consists of synthesizing examples to expand the database when supervised data is scarce. These strategies have shown promising results applied to NLP and are especially interesting for the problem proposed in this study, considering the existence of a large number of unlabeled data and a small number of labeled examples.

In this research, we intend to use several Portuguese language models to represent our texts and classify them into 50 categories using supervised and semi-supervised algorithms. The idea is to investigate and compare several techniques to optimize text classification under limited supervision as well as develop a classification tool usable in a Brazilian public agency.

1.1 MOTIVATION

The Public Prosecutor's Office of the State of Paraná (*Ministério Público do Estado do Paraná* - MPPR), Brazil, is an institution that represents the interests of society and operates in several areas such as education, environmental, public health, and human rights. One of its main duties is to receive the demands of the population and later forward them to the sector most suited to its resolution. Upon receipt of a demand, the person responsible for recording the information must fill in a brief description of what was reported by the citizen as a short text and associate it to a thematic field of multiple choice, to enable an adequate classification and forwarding of the

demands to the competent sectors of the Public Prosecutor's Office. Currently, MPPR has about 470 units throughout the state of Paraná, which register, on average, more than ten thousand demands per month.

A recurrent problem found due to the large volume of records and the absence of standardized protocols is the lack of reliability in the information registered. This fact was validated through a study involving the demands dataset, indicating whether the label of the theme assigned in the system was correct concerning the registered text describing a demand. A team specialized in the public service area analyzed the institution's database in order to validate 100 observations in each of the 50 selected topics, resulting in the database used in this study. During the experiment, 9,387 demands were analyzed for the creation of a dataset with 5,000 examples, having discarded 4,387 incorrectly classified records in the system, that translates to an (human) error rate of 46.73%.

Intending to improve the records in the system, we propose to create a model for classifying the theme of a demand based on its description. The automatic classification has the advantage of standardizing records and reducing inconsistencies, as well as decreasing the time taken to fill demands in the records system. Given the problems pointed out about the reliability of the records and given the high cost of employing legal experts to manually review current records, we intend to investigate techniques that increase the performance of text classification with few labeled examples.

1.2 CHALLENGES

Some challenges in the use of textual data have to do with standardization of written form and typos. The Public Prosecutor's Office is an institution composed of a large number of heterogeneous units, with the demand management system being accessed by interns, social workers, auditors, and prosecutors, among other contributors. Thereby, there is no common understanding of how demands should be reported, generating different forms of writing that are later reflected in the classifiers.

In addition, some themes present in the system overlap according to the demand resulting in a description that can be classified in more than one subject at the same time. However, considering that the system only allows the inclusion of one category per demand, the current guidance is that the dominant subject must be determined and the labeling of this theme is carried out. Again, given the heterogeneity of employees, this understanding is not common to all, reason why the data used in this research has been validated by an internal team of legal experts with the same orientation about how demands should be categorized so that the labels represent the standardized taxonomy of the institution as much as possible. Optimally, we would like experts to do the labeling work on a large scale, but in practice, the cost is very high. Due to the need for validation of the records prior to their use, the amount of available examples for training the model is relatively small when considering the number of categories for classification.

Finally, we point out that some of the evaluated models have a prohibitively expensive computational cost, being necessary to work with sub-optimal parameters due to the lack of resources for the complete implementation of the algorithms presented in this research.

1.3 OBJECTIVES

The main objective of this research is the evaluation of NLP techniques on a corpus of demands to the Public Prosecution's Office decreasing the need of labelled data during training. In the stage related to feature extraction, this optimization will be done by comparing transfer learning techniques and classic vectorization techniques, more specifically word2vec, as well as the impact of fine-tuning on language models. With regards to training the classifiers, linear classifiers, ensembles, and neural networks are evaluated using supervised and semi-supervised algorithms, comparing their performance on relatively small labeled datasets.

More specifically, we demonstrate that, by carefully combining pre-trained language models (on an unlabelled legal corpus comprising demands made by the general public) and data augmentation techniques, it is possible to automate the mentioned task and reduce inconsistencies and time spent manually filling out such information. We emphasize that the legal corpus deals with very specific matters and, for this reason, is really time-consuming and expensive to annotate.

1.4 CONTRIBUTIONS

The main contributions of this research can be understood as the investigation of Natural Language Processing techniques that optimize the quality of supervised tasks in low data regimes. Here is a summary of these contributions:

- We compare different dense feature representations, from fixed (context-agnostic) representations such as word2vec to context-dependent (Transformer-based) representations such as BERT.
- We compare the impact of using specialized corpora in training the aforementioned language models, providing two language models specialized in the legal area for the Portuguese language, and increasing the resources available to the community.
- We investigate several techniques with the potential to improve classification in situations where little data is available. Regarding the topic, we compare supervised, semi-supervised, data augmentation, and transfer learning techniques, seeking to identify the one that most contributes to the optimization of available resources.
- In a more practical approach, our contribution consists in the creation of a model to automate public service tasks, seeking to standardize the legal understanding of these classifications and making the service more agile and reliable.

1.5 ORGANIZATION OF THE THESIS

This dissertation is organized as follows: initially, we present the theoretical concepts that underlies our research in Chapter 2, then we give a brief overview of previous studies that addressed issues related to our work in Chapter 3. In Chapter 4, we present the research methodology, followed by Chapter 5, in which we present the datasets that we used for our experiments as well as the experiments carried out and their results. Lastly, the conclusions of our work are presented in Chapter 6.

2 THEORETICAL BACKGROUND

This chapter seeks to address the theoretical foundations used in this research. We will approach three main aspects of NLP, related to text preprocessing, feature extraction, and classification models. The theoretical elements associated with each of these stages are detailed down below, as well as intermediate concepts and relevant metrics of evaluation.

2.1 TOKENIZATION

Tokenization is a preprocessing technique that consists of converting text sequences found in a corpus (a collection of texts accessible by a computer) into smaller parts, called tokens. It can be performed at several encoding levels, from whole words to single characters. The most common form of tokenization is the word tokenization, a process that segments texts based on delimiters such as space and punctuation, generating a vocabulary composed of entire words. From the token mapping, we obtain a reference vocabulary based on the analyzed corpus. Any new token found after this process (ie.: during inference) is considered as out-of-vocabulary (OOV). A special ("UNK") token is used for that purpose.

The tokenization process performed at the character level consists of defining a vocabulary based on single characters, like letters and punctuation. In this type of tokenization, we do not have OOV tokens, since any text fragment can be subdivided and represented sequentially in a character level previously mapped in the vocabulary. On the other side, since we have a limited amount of representation for a complex corpus, we generally need more intricate models to join the character representations in meaningful ways.

A tokenization strategy with increased interest in the NLP field is the subword tokenization, which emerged with the intent of mapping a potentially infinite vocabulary using a finite number of tokens. In this tokenization process, each word in a given corpus can be represented from a character level to a whole word level, in a way that reduces the number of OOV considering that most words out of the vocabulary can be subdivided into smaller parts until they match a corresponding representation. The subword tokenization usually divides the texts using iterative algorithms that consider the generated token frequency and its probability of occurrence in the corpus to decide whether to maintain or not, the new token. One of the most known algorithms is the *Byte Pair Encoding* (BPE) (Gage, 1994; Sennrich et al., 2015), which starts the analysis of the corpus from a character level of all words and iteratively associates pairs of characters until it reaches a stop criterion, like an upper limit on the vocabulary size. Other examples of subword tokenizations can be found in the *WordPiece* algorithm (Wu et al., 2016), used by the BERT model (Devlin et al., 2019), and *SentencePiece* (Kudo and Richardson, 2018), both adapted from the aforementioned BPE.

2.2 TEXT NORMALIZATION

Text normalization is a set of preprocessing techniques used to decrease word inflection. Some frequently used techniques in this process are converting uppercase to lowercase, normalizing special characters, mapping URLs and e-mail addresses to unique tokens like "URL" and "EMAIL" and converting numerals to a single value, like zero. Other more advanced technique is the lemmatization, used to reduce the variability caused by inflections like gender and number and extracting only the radical of the words analyzed.

The above techniques are very popular and many of them can be found already implemented in Natural Language Processing tools, like the *python* packages *nltk*[1] and *spaCy*[2]. Other specific techniques can be implemented with the help of regular expressions or by training custom models for each desired task.

2.3 WORD FILTERING

Word filtering is a simple preprocessing technique, intending to limit the number of features extracted from a corpus, focusing on the most significant features. Usually, we apply two types of filter: (i) removing the rare tokens by specifying a threshold and filter the resulting tokens by their frequency in the corpus, with the intent of removing typos and rare words; and (ii) removing stop words, which are uninformative tokens that do not carry much information and have no relevance in a NLP task, such as conjunctions, articles and prepositions.

2.4 TEXT REPRESENTATION

To use texts in NLP techniques, a necessary preliminary step is the conversion of texts into computer-readable formats. This process is usually done by converting tokens into numerical representations, such as vectors. In this section, some of the most used text representation techniques in the literature will be presented.

2.4.1 Bag-of-Words

Bag-of-Words (BOW) is a text vectorization method that uses the vocabulary of the corpus as feature. The words of a document are represented as a simplification of a one-hot-encoding vector, with the value 1 when the word occurs and zero otherwise.

Considering that the corpus has n documents and m unique words, the BOW technique will use a matrix $n \; x \; m$ to represent the texts. Since the vocabulary of the corpus can be extremely large, it will lead to sparse representations in the vector space given that each text representation will have the length of the entire vocabulary.

[1] https://www.nltk.org/
[2] https://spacy.io/

As the name suggests, the BOW method is an unordered bag, in the sense that it disregards grammar and word order, considering only its counts. This information loss regarding the context makes it impossible to use this type of representation in scenarios where contextual information is required.

2.4.2 TF-IDF

Given that the distribution of word frequency in most languages follows an asymmetric distributions, with common words appearing multiple times and other rare words almost not appearing, the use of simple count of word frequency to represent texts like BOW can distort textual representation.

To overcome this problem, it is possible to apply a normalization method called Term Frequency-Inverse Document Frequency (TF-IDF), which is obtained in two steps. The first step is to calculate the number of times a term occurs in a document, called term frequency. This method was first proposed by Luhn (1957) and is detailed in Equation 2.1.

$$tf(t,d) = \frac{f_{\mathrm{t,d}}}{\sum_i f_{\mathrm{i,d}}} \tag{2.1}$$

Where the term frequency of term t in document d is given by the division of the frequency f of the term t in document d and the sum of frequencies of all terms i in document d.

The second part of TF-IDF is given by computing the inverse document frequency. This concept was developed by Spärck Jones (1972) to statistically quantify the specificity of each term in a corpus. Equation 2.2 describes the calculations.

$$idf(t,D) = log\frac{\mid D \mid}{\mid \{d \in D : t \in d\} \mid} \tag{2.2}$$

In the equation above, D denotes the total number of documents in the corpus while the quotient is the number of documents d that contain the term t (i.e., $tf(t,d) \neq 0$). The inverse document frequency of term t in D is given by the logarithm of the fraction of the total number of documents D and the number of documents where the term t appears.

Finally, we can obtain the term frequency-inverse document frequency as the simple product of the two previous equations.

$$tfidf(t,d,D) = tf(t,d) * idf(t,D) \tag{2.3}$$

The product of Equation 2.1 and 2.2 is a weight that increases when the frequency of term t is high in a small group of documents and decreases when the frequency is small or if it appears in many documents. Thereby, tf-idf serves as a normalization for relevant terms that

occur only in certain document classes and reduces the significance of very common or very frequent terms across all document classes.

2.4.3 Embeddings

Embeddings can be seen as continuous learned representations for highly-dimensional categorical data, such as text, image and audio. Specifically, in the field of NLP, embeddings are associated with the tokenization process, representing each token that composes the defined vocabulary as a set of real values in Euclidean space. When used with word tokenizers, they produce word embeddings, which maps each word w from the vocabulary V to a real-valued vector in an embedding space of dimensionality D. It is also possible to obtain embeddings of paragraphs or even entire documents.

The most used embeddings present in NLP tasks are trained with neural networks using self-supervised tasks such as next sentence prediction and masked word prediction. This kind of training approach consists of estimating the joint distribution of a sequence of tokens in a corpus while trying to capture some semantic aspects of the mapped token. In many situations, the generated embeddings can be shared and used in multiple NLP tasks, in which case it is necessary to keep the same preprocessing and tokenization applied when generating the original embeddings to obtain the corresponding tokens.

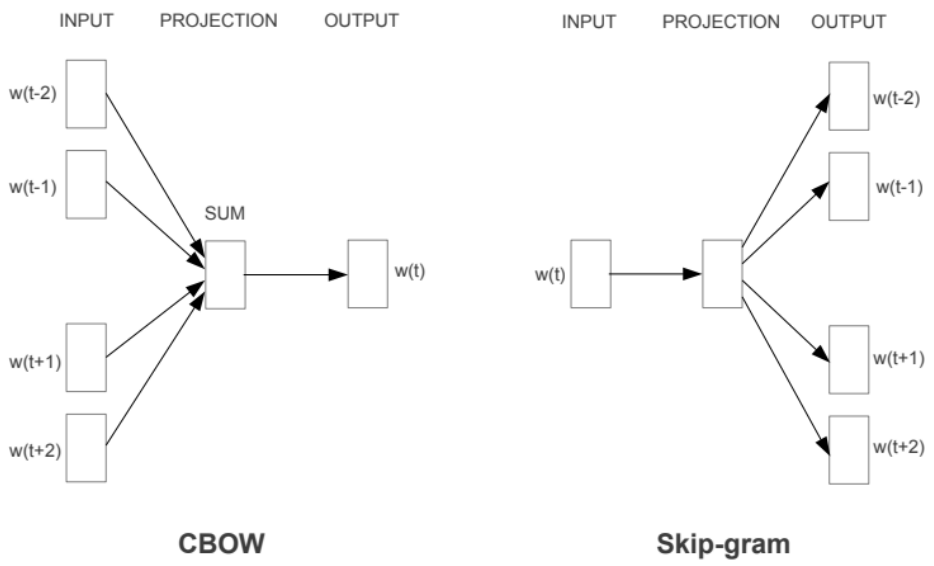
Since it is possible to train the models to obtain embeddings from original texts without the need for manual labeling, the datasets for this kind of task are easily obtained, with multiple open-source data available on the Internet.

2.4.3.1 Word2vec

Word2vec was proposed by Mikolov et al. (2013) and consists of an architecture used to produce word embeddings. The idea is to use neural networks to project words into an Euclidean space that preserves the semantic relations between words in a way that similar words are projected close to each other. Two different self-supervised tasks were proposed by the authors: Continuous Bag-of-Words (CBOW) and Skip-Gram. Figure 2.1 shows the architecture of both Skip-Gram and CBOW models.

From the figure above, we can see that CBOW architecture consists of predicting a target word given its context. It uses a feedforward neural network to project the words of a corpus in a shared hyperspace. The model is called Continuous Bag-of-Words since it does not consider the order of words to make the projection but, unlike the BOW technique, it uses a continuous representation to capture their context. The Skip-Gram architecture performs the inverse task of CBOW and tries to predict the context words given the target word. The current word is fed to a log linear model, which considers the distance of the surrounding words to the target word to adjust the weights of the model.

Figure 2.1: CBOW and Skip-Gram tasks for word2vec



Source: Mikolov et al. (2013).

Regardless of the choice of the training task, some hyperparameters can be optimized to obtain the best representation of the tokens using word2vec, detailed below.

- `Window size`: context to be considered when training the models. For example, a window size of 2 indicates that the model will consider two words to the left and two words to the right of a central word in the training task.
- `Vector size`: size of the output embedding that will represent the words.
- `Minimum count of words`: how many times a word must appear in the corpus to be included in the training task. This parameter is useful to exclude rare words and typos from the final vocabulary.
- `Number of iterations`: number of training steps used during model training.

2.5 TRANSFER LEARNING

Transfer learning is a paradigm in machine learning which comprises reusing learned representations across different tasks/domains. Using the definition of Pan and Yang (2010), if we have a source domain D_S with an associated task T_S and a target domain D_T with an associated task T_T, the objective of transfer learning is to improve the learning of a predictive function $f_T(.)$ in D_T with the knowledge of D_S and T_S, the source and target being different from each other.

Transfer learning techniques have the advantage of needing fewer data to adapt the source model to a target model, also being less costly and faster for training. In the field of NLP, transfer learning is generally applied with language models pretrained in large amounts of "open

domain" corpora such as Wikipedia or large datasets of books and papers available online that are later adapted to a particular task or domain using a specific corpus, like medical reports or legal documents.

2.6 LANGUAGE MODELS

A Language Model (LM) is used to predict the probability of a sequence of words, usually decomposed into sub-word units. LMs have an important role in NLP tasks, being used but not limited to Speech Recognition, Spelling Correction, Machine Translation, and Natural Language Generation.

Given a text sequence $P(W)=P(w_1,w_2,...,w_T)$, the joint probability can be factored as the product of conditional word probabilities (Equation 2.1):

$$\begin{aligned} P(w_1, w_2, ..., w_{\mathrm{T}}) &= P(w_1)P(w_2|w_1)...P(w_{\mathrm{T}}|w_1...w_{\mathrm{T\text{-}1}}) \\ &= P(w_1)\prod_{i=2}^{T} P(w_{\mathrm{i}}|w_1...w_{\mathrm{i\text{-}1}}) \end{aligned} \tag{2.4}$$

where i is the time step and T is the total number of words in the text sequence. Put differently, by estimating the joint probability of observed words, one can infer how likely a sequence of words (ie.: sentence, paragraph) is, that is, given the sentence *"my dog is cute"*, we can calculate the probability of the word *my* at the beginning of the sentence, the probability of the word *dog* given that the previous word was *my* and so on.

Just like the statistical language models (n-gram), the most recent language models based on neural networks are trained from self-supervised tasks, requiring no labeled data and making it easier to obtain abundant texts for training. Moreover, it is possible to jointly fine-tune neural language models and the task specific layer (ie.: softmax), something not possible for n-gram language models. Below we present some details of BERT, a language model that will be explored more thoroughly in our research.

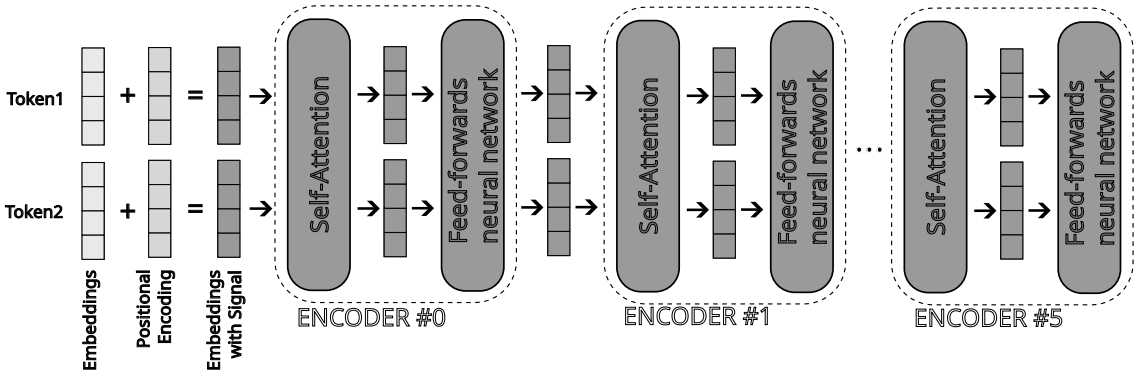
2.6.1 Transformers

The Transformer architecture was initially proposed by Vaswani et al. (2017), that mostly relies on attention mechanisms. Before the emergence of the Transformer, the most used architecture for NLP was the Recurrent Neural Network (RNN), which processes texts in a sequential manner, making parallelization difficult with underutilization of modern Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). To overcome this limitation, the Transformer architecture proposed a self-attention structure, allowing parallelization by computing the operations of all tokens of a sentence at the same time.

The Transformer architecture basically consists of two components called encoders and decoders. The encoding partition is composed of six identical encoder layers that receive the

input and compute an output to pass to the decoder component which, in its turn, is composed of six decoding layers. We explain in more details the encoder layer through Figure 2.2.

Figure 2.2: Encoder structure inside the Transformer architecture



Source: Adapted from Alammar (2018).

In the example shown in Figure 2.2, we have a sentence of two tokens, that are represented in embeddings with a positional encoding component. This mechanism describes the position of a token in a sequence in the form of a vector. The vector is given by a function that maps a predefined pattern learned by the model to retain the relative position of each token in a sentence. Considering that the structure of the model does not use recurrence or convolution operations, this function is required for the model to make use of the order of the sequence. The authors of Transformer proposed the use of sine and cosine functions alternately to encode the positions of the tokens with the following equations:

$$PE_{(\text{pos},2\text{i})} = sin\left(\frac{pos}{10000^{2\text{i}/\text{d}_{\text{model}}}}\right)$$

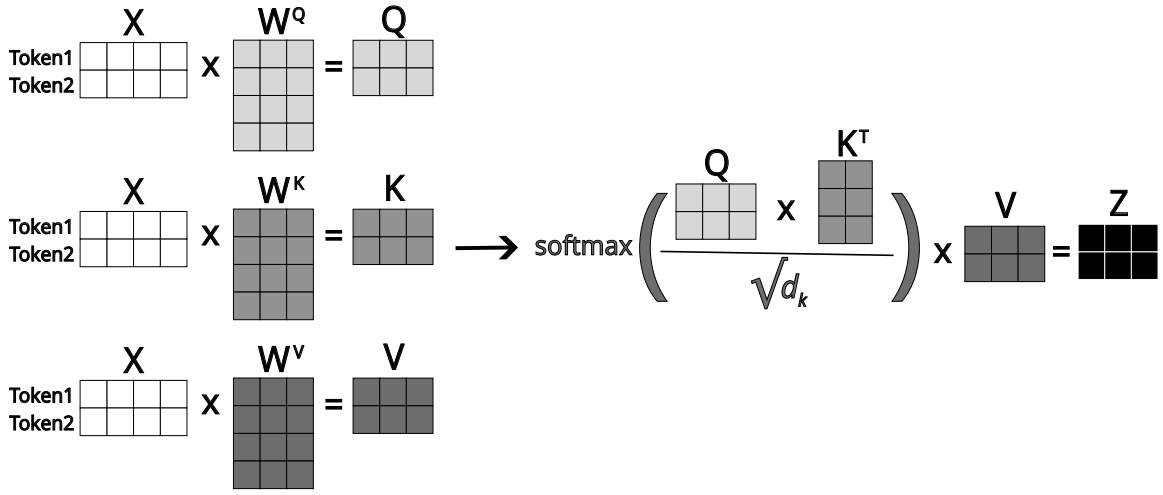
$$PE_{(\text{pos},2\text{i}+1)} = cos\left(\frac{pos}{10000^{2\text{i}/\text{d}_{\text{model}}}}\right) \tag{2.5}$$

Where *pos* is the position of the token in the sentence and i is the dimension of the embeddings. The encodings are summed to the embeddings, and together they form the input of the Encoder.

The joint representation of the embedding and the positional encoding form the final token representation, the embedding with position signal. It is then passed as an input to the first encoder layer (represented as Encoder #0 in the figure), that is composed by two main structures called self-attention layer and feed-forward neural network. The output of the first encoder is passed to the next encoder with similar structure and so forth, until the last encoder layer.

The attention mechanism used in the self-attention layer was first proposed by Bahdanau et al. (2015) and latter adapted to self-attention in the Transformer architecture. To better illustrate the self-attention layer, we present in Figure 2.3 the summary of calculations done by the model.

Figure 2.3: Attention mechanism



Source: Adapted from Alammar (2018).

From Figure 2.3, we can see a matrix X representing the tokens of a sentence (in the example, the sentence was simplified to two tokens), as well as three different (weight) matrices, designated as W^Q, W^K, and W^V. The weight matrices are initialize with random values and are learned during the training of the model. Formally, the attention function is used to map the query vector and a pair of vector's key-value to an output, with all vectors being associated with the tokens of the sentences. The result is computed through a weighted sum of these values, with the weights associated with each computed value obtained from a function that measures the compatibility of the query with the respective key. Therefore, given the query vector denoted by q and the key-value vectors denoted by (k_i,v_i), the attention mechanism results from the weighted sum of the value vector, given by equation 2.6:

$$a = \sum_i \alpha_{\mathrm{i}} v_{\mathrm{i}} \tag{2.6}$$

Where α_{i} is the weight associated to the value v_{i}, representing a compatibility score between the query q and the keys k_{i} through a function g:

$$\alpha_{\mathrm{i}} = \frac{exp(g(q, k_{\mathrm{i}}))}{\sum_i exp(g(q, k_{\mathrm{i}}))} \tag{2.7}$$

The attention mechanism proposed by Vaswani et al. (2017) and denoted as Scaled Dot-Product Attention is defined as the dot product of the query with all keys divided by $\sqrt{d_{\mathrm{k}}}$, d_k being the query and key vector dimension, applied to a softmax function to obtain the value

weights. Let Q be the query matrix that represents all query vectors together and K and V the matrices that jointly represent the key-value vectors, the resulted attention matrix can be obtained by:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_{\mathrm{k}}}}\right)V \tag{2.8}$$

To increase the power of the attention mechanism, the authors proposed to perform this operation multiple times initializing the values of the matrices W^Q, W^K, and W^V with different random values in a procedure called multi-head attention. In the original paper, they proposed using eight different attention functions in parallel, and later concatenating the outputs. According to the authors, this mechanism allows the model to attend to different parts of the information being processed each time it is applied, improving the model's performance compared to the use of a single head attention mechanism.

At the end of the multi-head self-attention layer, the output is passed as input to a feed-forward neural network, a part of the Transformer encoder composed of two fully connected layers. These layers implement two linear transformations using a ReLU activation between them in the original paper and a GeLU activation in BERT.

The final output of the encoder part of the Transformer architecture is a set of attention vectors that are passed to the decoder structure. The decoder layer is comprised of a similar structure of the encoder layer, with a self-attention layer and a feed-forward neural network, but has a encoder-decoder attention layer in between. This specific structure receives the output from the encoder partition, that is used by the model to focus on proper parts of the sentence.

The decoder layer is very similar to the encoder, but it attends only to past tokens, forming a sequence of words until reaching a stop criterion, such as obtaining a special token that signalizes the end of the sentence. At the last decoder layer, the output is a vector with the size of the vocabulary. This vector is passed to a softmax function that outputs the probability of each token, and the model chooses the token with higher probability as the output of the step.

The Transformer architecture inspired the development of several language models architectures, with great improvement due to its parallelization features and improving the state-of-the art across many different NLP benchmarks. One particular successful language model is BERT, which uses only the encoder part of the Transformer and will be detailed in the next section.

2.6.2 BERT

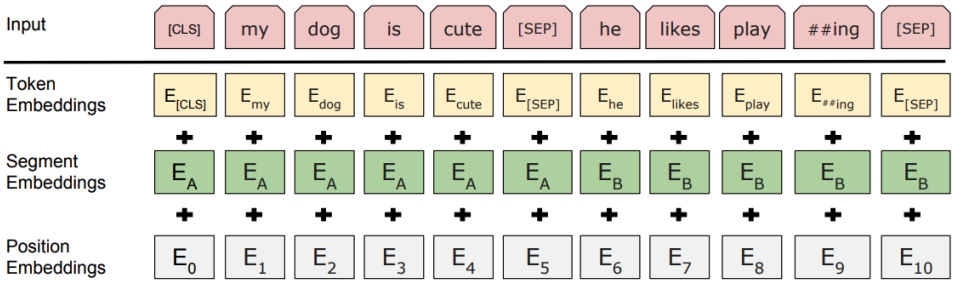
BERT (Bidirectional Encoder Representation from Transformers) is a language model proposed by Devlin et al. (2019) and is based on the Transformers architecture (Vaswani et al., 2017), more specifically the Transformers encoder. One of its main innovations compared to previous language models is the use of a deep bidirectional training task, resulting in word representations

that consider left and right contexts at the same time. In the next subsections, we will present some fundamental concepts of previous models adapted by BERT as well as its architecture and strategies for training.

2.6.2.1 Input and output representations

The authors propose two self-supervised tasks: masked language modelling (MLM) and next sentence prediction (NSP). In MLM, a single sentence (with some words masked) is fed to the input layer and the model is trained to unmask those words. In NSP a pair of sentence is fed and the model is trained to predict if the second sentence follows the first. The first position of the input is always a special token designed as [CLS] which can be used similarly to a sentence representation for classification tasks. When using two sentences as input, BERT demands a special token [SEP] to indicate where the first sentence ends and the second one starts. When using a single sentence, the same token is used at the end of it. In addition to this special token, BERT uses segment embeddings to indicate from which sentence a specific token belongs to. The tokens are produced using the WordPiece algorithm with a 30,000 token vocabulary reported on the original paper. Each token is represented as the sum of the corresponding token, position, and segment embeddings. Figure 2.4 illustrates the process of tokenization and embeddings produced by BERT.

Figure 2.4: BERT input example



Source: Devlin et al. (2019).

2.6.2.2 Model pretraining

BERT pretraining uses two distinct self-supervised tasks that allows training the model in a deep bidirectional way. The input for these tasks is given by a sequence containing two distinct sentences that pass through the Masked Language Modeling (MLM) and the Next Sentence Prediction (NSP) tasks.

The Masked Language Modeling task randomly masks 15% of the words of the sentences, replacing them with the following tokens: (1) the [MASK] token 80% of the time, (2) a random token 10% of the time, and (3) left unchanged 10% of the time. This mechanism differs from usual language model tasks that predict the next or previous words in a sentence, allowing

the model to see all the tokens in the layer and train it bidirectionally without revealing the masked token. The language model then tries to predict the masked tokens produced by this self-supervised task through a softmax function over the vocabulary.

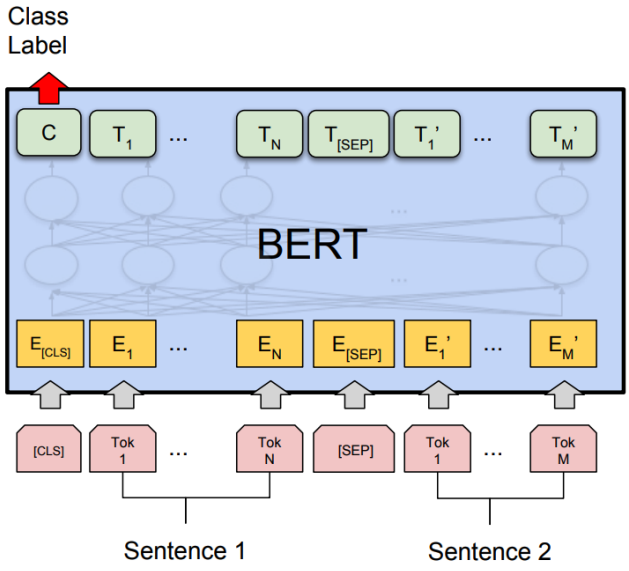
The Next Sentence Prediction task receives two sentences and is asked to predict whether the first sentence is followed by the second sentence or if it is a random sentence from the corpus. That is, the model is trained on pairs of positive (second sentence follows the first) and negative samples (randomly sampled pairs). This means that it is possible to sample a balanced set of positive/negative samples for any given corpus.

2.6.2.3 Fine-tuning

BERT is a computationally expensive model to train from scratch (according to the authors, it took four days on 4 cloud TPUs in Pod configuration, 16 TPU chips total for BERT base) but it is possible to use a pretrained model and fine-tune it in a relatively inexpensive way, in 1 hour using a single Cloud TPU or in a few hours on a GPU (Devlin et al., 2019).

The fine-tuning can be based on a specific task like classification, question answering, or similarity scoring, or can be used to adapt BERT's vocabulary to domain-specific data, in which case we can use the MLM and NSP tasks in the customized corpus to obtain specialized embeddings. For the classification task, BERT uses the hidden state of the special token [CLS] in the last layer as the representation of the text sequence passed to it and apply a softmax layer to predict the probability of the labels, as illustrated in Figure 2.5.

Figure 2.5: BERT classification example



Source: Devlin et al. (2019).

2.6.2.4 Feature-based approach

As mentioned above, it is possible to extract word representations from BERT, similar to other word embeddings techniques. The difference is that BERT is trained bidirectionally and extracts contextual embeddings, which means that, unlike context-free techniques like word2vec that obtain fixed embeddings per token, BERT produces variable token embeddings according to the context given by other tokens in the sentence. In other words, it is not possible to extract embeddings as a lookup table, but only considering the context of the token applied to the sentence it belongs to.

Considering that BERT is composed of 12 identical stacked layers, it is possible to extract representations from any of them to use as features in a classification task, for example. Experiments carried out by the authors of the original paper tested several types of embeddings and concluded that the best results were obtained by concatenating the last four hidden layers of BERT to extract the token representations (Devlin et al., 2019).

2.7 LOGISTIC REGRESSION

The logistic regression model is commonly used for binary classification. It is in its essence a linear model (parameterized by a vector of weights w and bias b). The sigmoid function is used to perform an operation called "squashing", with each value being in the range (0,1) and all the values summing to 1. The function is given by the equation:

$$y = \frac{1}{1 + e^{-z}} \tag{2.9}$$

And z is defined as:

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b \tag{2.10}$$

Note that z is a common linear regression function and y is a sigmoid function, which transforms the domain of z from $[-\infty, \infty]$ to [0,1].

For multinomial logistic regression, that is, for tasks involving more than two classes to be predicted, the sigmoid function is is replaced by the softmax function to compute the probability of each class k present in the data. Given a vector $z=[z_1,z_2,...,z_k]$ containing the k values associated with the classes, the softmax function is used for squashing, similar to the sigmoid function in the binary logistic regression. The softmax function is given by:

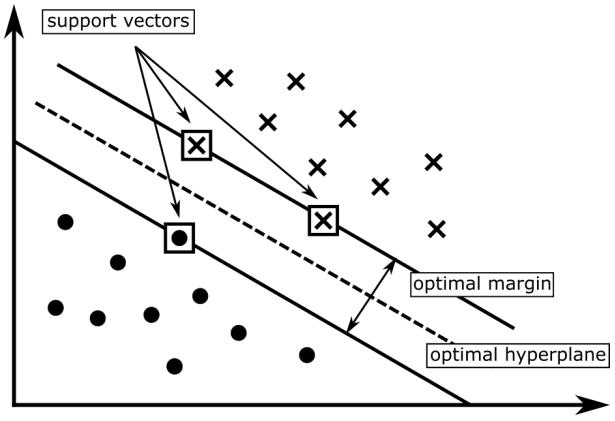
$$softmax(z_i) = \frac{e^{z_j}}{\sum_{j=1}^{k} e^{z_j}}, 1 \leqslant i \leqslant k \tag{2.11}$$

As said before, the softmax function is an exponential function that transforms any real value to a range of (0,1), respecting the proportionality of the values, that is, larger values tend to be transformed near 1 while smaller values get probabilities near to zero (Jurafsky and Martin, 2019).

2.8 SUPPORT VECTOR MACHINES

Support Vector Machine (SVM) is a model proposed by Cortes and Vapnik (1995), where the decision boundary between two classes is defined by the data points "supporting" the maximal margin between these classes (the support vectors). More specifically, when used as a linear classifier, it creates a hyperplane to separate distinct categories, with an illustration of the process presented in Figure 2.6.

Figure 2.6: SVM structure in two dimensional space



Source: Adapted from Cortes and Vapnik (1995).

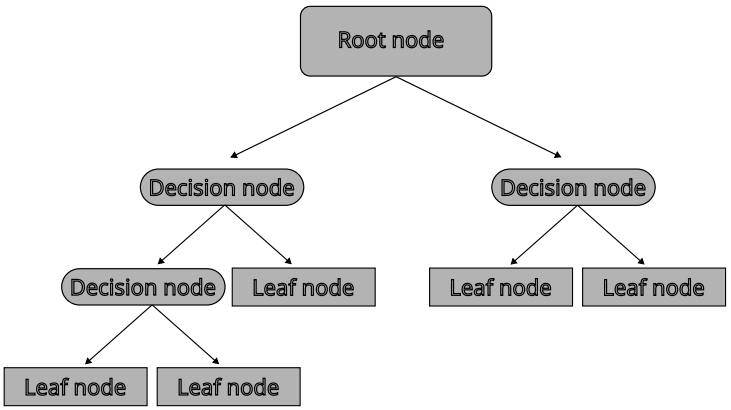
A major drawback of the linear approach is that most real world datasets cannot be easily divided by these simpler surfaces, reflecting in a poor or even nonexistent decision function. To overcome this limitation, one can use the kernel trick, which maps the data points to another feature space where they are linearly separable. This mapping process is related solely to the data size and not the dimensionality of the space, being an efficient method to deal with high dimensional feature spaces. The most common kernel functions for this transformation are the linear, gaussian, sigmoid, and radial basis functions.

Finally, although SVM is originally designed for binary problems, it is possible to train the classifier using the one-versus-all strategy when facing multiclass problems. The main idea is to split the multiclass data into several binary problems, by alternating the label of the predominant class as zero and the rest as one or vice-versa, and the training c binary classifiers, c being the number of classes of the dataset. For the final prediction, we select the class with the higher score to classify an observation.

2.9 DECISION TREES

Decision tree is an algorithm based on recursive partitions that follows an intuitive tree shape. The rationale is to split the data in contiguous regions using in its structure the concept of branches and nodes, and some measure of impurity. At each node, a decision rule determines whether to split the data in two or to stop in a leaf with a category associated with it. Figure 2.7 presents an example of the described decision tree structure.

Figure 2.7: Decision tree structure.



Source: Elaborated by the author (2022).

Decision trees are known to be easily interpretable and effective to identify relevant features for classification problems. They also can deal with a wide range of input data without the need for normalization and even containing a number of missing values. Besides, since it is a non-parametric method, there are no underlying assumptions about the distribution of the data. One disadvantage is that decision trees can become very large and complex, leading to overfitting without proper generalization of the model.

It is very common to use ensemble methods with decision trees, to improve the robustness of the model and produce better performance than the use of a single tree.

2.10 ENSEMBLE LEARNING

In machine learning, ensemble learning is a paradigm that uses several learners to predict attributes, instead of the classical techniques that rely in only one classifier. The objective of this strategy is to obtain stronger and more accurate classifications leveraging of the power of multiple learners. We detail in the next sections two of the main used ensemble strategies: bagging and boosting.

2.10.1 Bagging

A very popular and important strategy of ensemble learning is the bootstrap aggregation, or bagging. It consists of using a bootstrap method, that is, constructing multiple synthetic data by sampling the training data with replacement, and after training the learners with the subsets. The prediction of each model is then aggregated to obtain the final label.

An application of the bagging method is the random forest algorithm, first proposed by Ho (1995). While a decision tree uses one model to classify the data, random forest trains multiple trees and gets a result based on the output of all trees. Since trees are notoriously noisy, they benefit greatly from the averaging of multiple classifiers. Random forests are less prone to overfit, generalizing unseen data very well. On the other side, they reduce interpretability, since the result is an average of multiple classifiers.

2.10.2 Boosting

Boosting is a method of ensemble learning that had great impact in machine learning, whose strategy is to leverage the power of multiple weak learners to construct a strong classifier. The weak learners are models that perform slightly better than a random guess, generating a final model correlated to the underlying structure of the data and reducing bias. One inconvenience of this method is that it tries to learn each data point exactly, being susceptible to overfit and not generalizing well to unseen data.

An example of boosting is the gradient boosting (Friedman, 2000), which tries to optimize the learning process by sequentially training individual models, each model learning from the errors of the previous models.

Gradient boosting uses a combination of weak learners, usually decision trees, and estimates them iteratively to get a strong model. When using decision trees, the trees are added one at a time and a gradient descent procedure is used to minimize an arbitrary differentiable loss function by a parametrization of the tree that reduces the residual loss.

2.11 SEMI-SUPERVISED LEARNING

Semi-supervised learning (SSL) is a set of machine learning techniques that use a combination of labeled and unlabeled data during training. In general, the dataset can be divided into two parts, with a small portion $X_s=(x_1,x_2,...,x_m)$ with associated labels $Y_s=(y_1,y_2,...,y_m)$ and a larger partition of data points $X_u=(x_{m+1},x_{m+2},...,x_{m+u})$ for which the labels are unknown (Chapelle et al., 2006).

SSL methods are mostly used when labeled data is scarce and difficult or expensive to obtain while unlabeled is readily available. In the words of Chapelle et al. (2006):

> "SSL classification methods attempt to utilize unlabeled data points to construct a learner whose performance exceeds the performance of learners obtained when using only the labeled data."

Under certain assumptions, unlabeled data can be used to improve the construction of a better classifier than the ones built using only the labeled data points by providing additional information and optimizing the classifier performance. Some of the main assumptions to use SSL methods were raised by van Engelen and Hoos (2019) and are described below.

- `Smoothness assumption`: this assumption states that, if two examples are close in the input space, their labels should be the same. It is important to note that this characteristic permits the propagation of labels in the supervised data to the unsupervised data by the proximity of features.
- `Low-density assumption`: the decision boundary of a classifier should not pass in a high-density area in the input space, that is, the division of the classes must be placed in a space where few data points are observed. This assumption also ensures the smoothness assumption, since if the decision boundary is placed in a high-density region, predicted labels will be dissimilar for similar data points.
- `Manifold assumption`: the manifold assumption presumes that the input space where the data is represented is composed of multiple lower-dimensional substructures called manifolds and data points that are placed on the same manifold have the same label. Therefore, if it is possible to find the manifolds and the data points associated with them, it is possible to assign the labels of the unsupervised data on the same manifolds.

2.11.1 Self-training

Self-training is the name given to a set of SSL techniques that consists of training a supervised algorithm using labeled data to produce a classifier and later using the classifier to predict the same classes in the unlabeled data, propagating the labels learned. Yarowsky (1995) was the first to propose the use of self-training to predict the meaning of words based on their context to avoid the ambiguity of words in documents.

Generally, self-training methods select samples with greater confidence in the predicted label of the unsupervised data to compose the supervised base with the pseudo-labels. The classifier is retrained using both supervised and pseudo-supervised data in an iterative process until a pre-established stopping criterion is met.

2.11.2 Co-training

Co-training is a technique proposed by Blum and Mitchell (1998) which consists of using two different views of the data to train classifiers using a small portion of supervised data and trying to obtain labels on the unsupervised data from those classifiers. Each view or feature extracted from the data must provide different and complementary information, ideally being independent and sufficient for correct classification. At each iterative step of the method, the samples with

higher confidence predictions of each classifier on the unlabeled data are added to the labeled data.

The intuition behind this method is that, given an unlabeled example x, with two different extracted features (x_1,x_2), if x_1 produces a highly confident label in the first classifier $f_1(x_1)$, it can be used to produce a labeled example x_2 of f_2 and increase the knowledge of the second classifier. Since x_1 and x_2 are independent views of x, x_2 produces a random sample of f_2 and can improve the classifier's performance.

2.11.3 Unsupervised data augmentation

Unsupervised Data Augmentation (UDA) is a semi-supervised method introduced by Xie et al. (2019) that combines labeled and unlabeled data to create robust classification models. The main idea is to create a classifier using a small portion of labeled data and apply the same classifier in each example of the unlabeled data and its augmented version, trying to predict the same label on the original and the synthetic example. It also relies on a novel technique called Training Signal Annealing (TSA), which is a way to control the examples that are passed from the unlabeled data to the labeled data to prevent overfitting.

2.11.3.1 Data augmentation

Data augmentation is a family of methods used to increase the number of samples in a dataset. It consists of the application of transformations to an example seeking to create another realistic example without changing its internal structure, or label in the case of supervised data. The requirement for an augmentation technique to be valid is that the transformation of x into its augmented version $\hat{x}$ maintains the ground-truth label of both real and synthetic examples.

There are several well-established data augmentation techniques in the field of image classification that present positive results and multiple implementation forms. In the field of text classification, it is more difficult to achieve such a state since minimal perturbations in a text can result in significant syntactic or semantic modifications. The application of data augmentation in texts can vary from the simple random replacement of words in a text to more sophisticated techniques that use language models to replace synonyms or generate paraphrases to enlarge the experimental dataset.

In the context of UDA, the authors mentioned two data augmentation strategies that can be applied to text data and were used with their model. We detail the methods below.

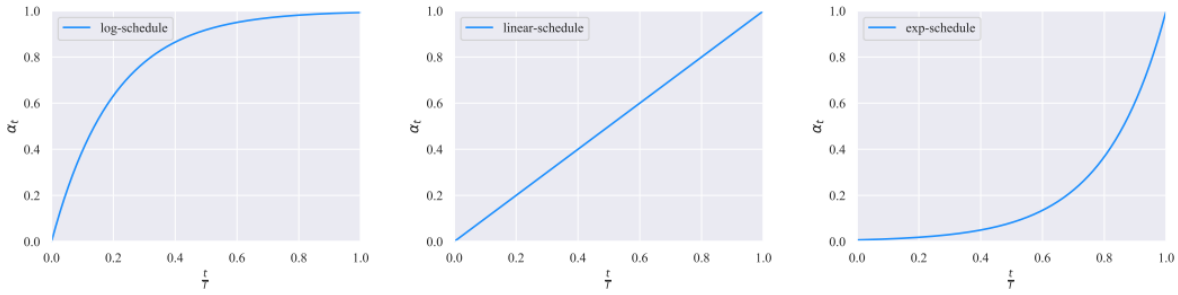
- `Word replacing with TF-IDF`: the goal of this transformation is to replace words in a sentence with other words present in the vocabulary. However, considering that some words are more important than others to characterize a class, the replacement is performed considering the TF-IDF score of each word, intending to replace only words that are uninformative while keeping the relevant ones.

- `Back-translation`: this term refers to the operation of translating a sentence to another language and later translating it back to the original idiom, creating a paraphrase as an augmented example of the original sentence without losing its semantics.

2.11.3.2 Training signal annealing

Training Signal Annealing is a technique used to prevent overfitting in semi-supervised learning. Since SSL generally has an imbalance between the size of labeled and unlabeled datasets, models trained on a limited size of labeled examples tend to overfit while the unlabeled examples are under fitted. Furthermore, if we only choose to use unlabeled data that the model is too confident about, it will tend to overfit these examples without gathering new information to reshape its decision boundaries. TSA tries to mitigate this problem by creating a "training signal", using only examples that are below a predefined threshold that increases according to a schedule. The threshold is understood as the confidence in the prediction of a label, which increases during the iterations of UDA's training steps. The author proposes three different functions to TSA, illustrated in Figure 2.8.

Figure 2.8: TSA schedules



Source: Xie et al. (2019).

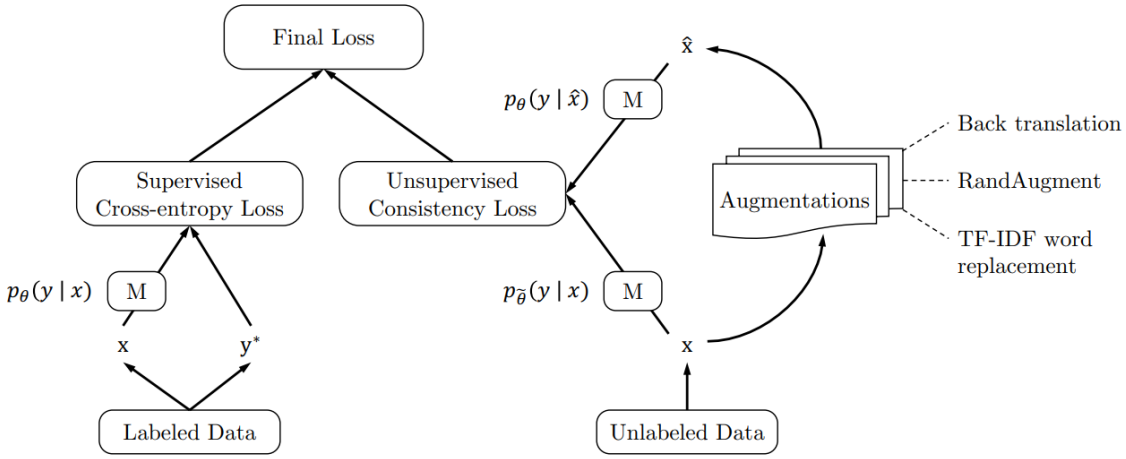
The figure above illustrates the three TSA strategies, with t being the actual training step and T as the total number of training steps. We can see, in the case of the log-schedule, that at the beginning of the training we already use a high confidence threshold to select the examples, while the exp-schedule uses the inverse rationale. It is advised to use the exp-schedule when the model is prone to overfit, keeping a low threshold during most of the training time and releasing the signal mostly at the end.

2.11.3.3 Training strategy

UDA opts to use data augmentation techniques in the unsupervised portion of the data because it is generally the larger part of the data available. The optimization occurs in two perspectives, the first being the model trained with the labeled portion of the data with corresponding cross-entropy loss. The second part tries to reduce the distance between the label of the unsupervised data

example and its augmented version, through an unsupervised consistency loss. Those two loss functions are weighted using a factor λ, with the whole process illustrated in Figure 2.9.

Figure 2.9: UDA training strategy



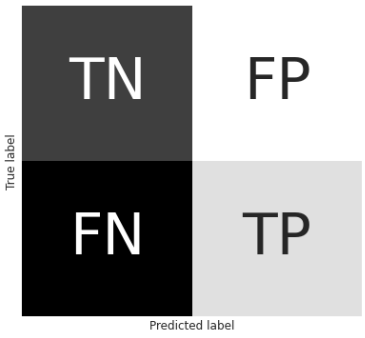
Source: Xie et al. (2019).

The strategy is that the noise injected by the augmented examples enforces the model to be insensitive to noise and more tolerant to changes in the input space. It also gradually propagates the information about the labels in the supervised data to the unsupervised portion.

2.12 EVALUATION METRICS

There are several metrics to evaluate the goodness of fit of a classifier. In this section, we will present the selected metrics used in this research to evaluate the results obtained by the classifiers. We separated the global model metrics from the metrics that concern the performance of classes, for a more precise definition of the concepts. As a starting point, we will follow the terminology described below, with the help of a simplified confusion matrix, illustrated in Figure 2.10.

Figure 2.10: Example of confusion matrix



Source: Elaborated by the author (2022).

- `True positive (TP)`: positive observations correctly predicted in the corresponding class.
- `False positive (FP)`: positive observations incorrectly predicted in the corresponding class whose true label belongs to another class.
- `False negative (FN)`: negative observations incorrectly predicted in another class whose true label belongs to the corresponding class.
- `True negative (TN)`: negative observations correctly predicted in another class.

2.12.1 Per-class Metrics

The metrics illustrated in this section relate to the individual performance of the classes that integrate our supervised dataset, providing a detailed examination of the model's performance.

2.12.1.1 Precision

The precision related to a given class can be understood as the model's ability to differentiate observations from other classes, not mislabeling them in the category in question. Broadly speaking, given the sum of observations that the model classified as being of a certain class, precision would be the percentage of correctly performed positive predictions. In multiclass problems, it takes into account the corrected labeled data in comparison to the observations labeled in all other classes. From the notation defined in the confusion matrix, we can extract the equation that summarizes this concept as illustrated below, for each class c.

$$P_c = \frac{TP_c}{TP_c + FP_c} \tag{2.12}$$

Where TP_c is the number of correct predictions of class c and FP_c is the number of incorrectly predictions of the same class.

2.12.1.2 Recall

Recall is a complementary measure to precision, as it can be understood as the ability of a classifier to correctly label observations of a given sample. In other words, considering the observations that truly belong to a certain class, the recall would measure the proportion of them that were correctly identified by the classifier in comparison to all examples in this category that were mislabeled in any other class in the case of a multiclass problem. Below, we present the recall metric with the help of the notation specified in the confusion matrix.

$$R_c = \frac{TP_c}{TP_c + FN_c} \tag{2.13}$$

We can define the recall with TP_c as the number of correct predictions of class c and FN_c as the number of observations incorrectly predicted as being from other classes and with true label being of class of c.

2.12.1.3 F1 Score

The F1 score can be understood as a metric that summarizes the results of precision and recall. It is a harmonic mean of these values that can vary between 0 and 1, with values closer to 1 indicating a better performance of the model in a given class. When broadly analyzing the performance of the model in the different categories, the F1 score can bring valuable insights by summarizing the performance, as illustrated with the following metric.

$$F1_c = 2 * \frac{P_c * R_c}{P_c + R_c} \tag{2.14}$$

Where P_c and R_c c are the respective precision and recall of class c, as defined before.

2.12.2 Global Metrics

The metrics described in this section represent the global aspects of the classifiers, providing a quick overview of the models as a whole in contrast to the individualized metrics by class.

In this regard, we separate flat and hierarchical metrics, the latter being differentiated from the former by considering internal hierarchy structures eventually present in supervised datasets.

2.12.2.1 Accuracy

In classification problems, accuracy can be understood as the simple percentage of correct predictions made by the classifier and is obtained by dividing the sum of the examples correctly predicted by the total of samples used in the test set, as shown in the equation below.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.15}$$

Accuracy values range from 0 to 1, with 1 representing a model with all predictions correct. In multiclass classification problems, a prediction is considered correct if the class with a higher score is equal to the class in the label.

2.12.2.2 Flat Precision, Recall, and F1 score

In addition to the per-class metrics of precision, recall, and F1 score, it is also possible to calculate the global values for the model. There are several ways of calculating these metrics, with some of them being optimized for multi-label classification or imbalanced datasets. In our research, we will consider the macro version of the metrics, which can be understood as a mean of the precision, recall, and F1 score in each of the c classes. The corresponding metrics are exposed below.

$$fP = \frac{\sum_{i=1}^{c} \frac{TP_i}{TP_i + FP_i}}{c} \tag{2.16}$$

The flat macro precision *fP* is defined by the sum of each individual class precision divided by the number of classes c. That is, given the Equation 2.12 defined before, the flat macro precision is the average of each precision per class P_c.

$$fR = \frac{\sum_{i=1}^{c} \frac{TP_i}{TP_i + FN_i}}{c} \tag{2.17}$$

Similar to precision, the flat macro recall *fR* is defined as the division of the sum of all individual recall values per class and the total number of classes c, with the individual recall values calculated with Equation 2.13.

$$fF1 = \frac{2 * \sum_{i=1}^{c} \frac{fP_i * fR_i}{fP_i + fR_i}}{c} \tag{2.18}$$

Finally, flat macro F1 score, or *fF1*, uses the individual F1 score per class obtained in Equation 2.14 to calculate the global metric from the harmonic mean of individual values.

We chose to define the macro versions of the metrics in this research, given our dataset structure, that will be demonstrated in future sections, as perfectly balanced. Besides the macro aspect of the metrics, they are classified as flat to contrast with the so-called hierarchical metrics. This definition is related to a specific data structure type that has a hierarchical organization of two or more levels, that is, it has a broader level of classes and other levels below, with each level below being more specific than the previous one.

2.12.2.3 Hierarchical Precision, Recall, and F1 score

The metrics presented in this section take into account the underlying hierarchical structure of datasets, penalizing predictions whose upper classes are not related, and benefiting otherwise. The hierarchical precision (*hP*), hierarchical recall (*hR*), and hierarchical F1 score (*hF1*) illustrated here were presented by Kiritchenko et al. (2006) and are defined as:

$$hP = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}'_i|} \tag{2.19}$$

Where $\hat{C}_i$ is the set of all ancestors (precedent classes) of the true class C_i, and $\hat{C}'_i$ is the set of all ancestors of the predicted class C'_i. Then, the hierarchical precision hP is given by the size of the set resulted from the intersection of the two sets $\hat{C}_i$ and $\hat{C}'_i$, divided by the size of set $\hat{C}'_i$, for each observation i in the dataset.

$$hR = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}_i|} \tag{2.20}$$

Similarly, the hierarchical recall hR is given by the fraction with numerator equal to the hierarchical precision hP, but with denominator being the size of set $\hat{C}_i$. That is, instead of using the set of the true class ancestors, we calculate the recall using the ancestors of the predicted class.

$$hF1 = \frac{2 * \sum_i \frac{hP_i * hR_i}{hP_i + hR_i}}{m} \tag{2.21}$$

The final metric *hF1* is calculated as a harmonic mean between the precision and recall values, similar to the flat metric formula but considering hierarchical precision and recall. A detailed example on the use of these metrics can be found in Chapter 4, with a real calculation done in Chapter 5.

3 RELATED WORK

This chapter presents the state of the art based on two different perspectives, text classification in general and the research of NLP tools applied to the legal area.

3.1 TEXT CLASSIFICATION

NLP is a field of research with increasing interest in recent years. The sub-area of text classification is a common task in NLP and has a wide variety of methods and applications. For industrial applications, one of the main requirements is the development of methods that can be used productively under small amounts of training samples (few hundreds up to low thousands samples per class).

Specifically addressing small data sets, we can mention the research of Bouillot et al. (2014) which suggested the use of different features extracted from text, an inner-class measure and an inter-class measure, both based on TF-IDF. The impact of the metrics in small datasets was studied, achieving good performance in classifying ten different categories but with the best result using a decision tree with the classical TF-IDF. Barz and Denzler (2020) tested the use of a simple cosine loss function instead of the usual categorical cross-entropy loss in order to fine-tune deep learning models for small datasets. The experiments showed better results using the cosine loss in low regime data in text classification (up to 100 documents per class) but still underperformed the BERT model. The experiments of Devi and Saharia (2020) tested multiple classifiers with different data sizes to measure the impact of a few labeled examples and obtained better performance with Multinomial Naïve Bayes using both TF-IDF and word frequency. In the research of Glaser et al. (2021), the effects of different methods were investigated aiming to improve textual classifiers in data scarcity scenarios. The authors tested several techniques with different datasets, concluding that there was not a universal form of optimization, but that the improvement depended on the scenario and the intrinsic characteristics of the data.

Multiple researchers addressed the use of data augmentation to overcome the small dataset issue or just to boost the performance of text classifiers in general. In this topic, we can mention the research of Wei and Zou (2019), which proposed a set of simple techniques for augmenting textual data, called Easy Data Augmentation (EDA). The operations are described as: (i) Synonym Replacement: replacing random words from a sentence with one of its synonyms; (ii) Random Insertion: inserting random synonyms of existing words of a sentence; (iii) Random Swap: swapping the position of two random words in a sentence; and (iv) Random Deletion: removing words in a sentence. This set of techniques quickly became popular in the NLP field due to the ease of implementation. The final results of the original paper showed that, although they do not present results with a great performance boost, they are still useful techniques to prevent overfitting in cases where labelled data is scarce. Huong and Truong Hoang (2020) tested the

EDA method in sentiment analysis, achieving the best result using a Random Forest classifier that improved the average accuracy by almost 10% using the augmented dataset in comparison to the use of the original data without augmentation. In the case of Feng and Mohaghegh (2021), they experimented with the use of EDA techniques by merging two different types of augmentation (Random Swap and Random Deletion) into the same sentence for all sentences in their dataset. The experiment was conducted in a binary sentiment classification problem, with the application of a CNN classifier. The results showed that the combination of more than one technique made the tested models worse, indicating the possibility of having created synthetic examples that are too far from the ground truth. Bootstrapping techniques as listed in the survey of Waegel (2013) can also be used to augment examples from a small dataset and a set of rules. An adaptation of this idea can be found in the work of Abulaish and Sah (2019), which combined Latent Dirichlet Allocation (LDA) and trigrams to create an augmented dataset for a binary classification problem. After applying a convolutional neural network (CNN) classifier, they obtained better results in the use of augmented data over the use of the original dataset. Anaby-Tavor et al. (2020) created a new augmentation method called LAMBADA (Language Model Based Data Augmentation), which uses the GPT-2 language model to generate synthetic data. They trained SVM, long short-term memory (LSTM), and BERT classifiers in three different datasets with a high number of classes and a relatively small quantity of labeled examples and demonstrated in the results that the proposed method outperformed state-of-the-art classifiers when labeled data was scarce. The study conducted by Shleifer (2019) tested two augmentation methods, with random token perturbation and back translation. It was demonstrated that the latter presented better results than the ULMFit model when there are few labeled examples. Specifically addressing the classification of documents in the legal area, Csányi and Orosz (2021) conducted an overview of text augmentation methods and the possibilities of application in this scenario. Several augmentation techniques were compared, such as EDA, round-trip translation (similar to back translation), augmentation by semantic similarity, text generation from language models and generation of synthetic examples from the definition of a representation space of the classes to be predicted. From the review of the techniques, the authors concluded that legal texts must be handled with care, due to their complexity and that none of the techniques studied in the article can be applied as a golden rule in any classification problem and any dataset.

Another approach to deal with small data when there are some unlabeled examples related to the labeled data is to use semi-supervised learning techniques. Martindale et al. (2020) used label propagation and label spreading algorithms to expand the labels of the supervised data to the unsupervised one and applied decision trees, random forests, and SVM classifiers. They obtained assessment metrics over 80% across all tested methods in the classification of nine categories. A promising SSL technique applied to text data is the co-training strategy proposed by Blum and Mitchell (1998). In the original paper, they used two perspectives of the data to train a Naïve Bayes model in each one and apply them to the unlabeled data to generate synthetic labeled examples. Their method was tested in website text data and outperformed purely supervised

learning. A variant of this methodology was proposed by Chen et al. (2019) on the classification of forum posts using CNN combined with word and character embeddings and included a verification step based on text similarity to synthesize the labels in the unsupervised data. They achieved an average of 2.77% of improvement to the baseline model accuracy and 3.2% in the F1 score. Xie et al. (2019) combined data augmentation and SSL techniques to create a novel model called Unsupervised Data Augmentation (UDA). The idea is to minimize a loss function composed of two distinct functions called supervised cross-entropy, obtained with labeled data, and unsupervised consistency loss, used with unlabeled data and their augmentation counterpart. Using only 20 labeled examples of the IMDb dataset, their model outperformed state-of-the-art results using all the available supervised data (25k examples). They also demonstrated that their method benefits from transfer learning techniques, like the use of BERT, and proposed a novel mechanism to prevent overfitting on semi-supervised tasks. Finally, Dehghani et al. (2017) proposed a hybrid model, suitable for situations where there is a small dataset with reliable labels and a large volume of weakly labeled data. The idea is to use a student model, trained from data with unreliable labels, and a teacher model, trained from data with consistent labels. From these instances, it would be possible to associate a confidence level from the teacher for the labels predicted by the student and improve the classifier. Results showed that, both in the sentiment classification task and the document ranking task, there was an improvement in training speed and performance compared to other state-of-the-art semi-supervised techniques.

3.2 NLP IN THE LEGAL DOMAIN

The legal domain presents a high volume of complex unstructured data with great potential for benefits from a more technical and automated analysis of its contents. In Brazil, there is an aggravating factor of high judicialization, which can be illustrated by the fact that, in December 2019, there were more than 77 million legal proceedings pending completion, according to a survey carried out by the National Justice Council (CNJ - Conselho Nacional de Justiça, 2020). This fact points to the importance of automating textual analysis and classification in the area of law, the reason why we will bring the important research done in this field.

Sulea et al. (2017) used court decisions of the Supreme Court of France to predict the law area, the ruling, and the date of the ruling of the analyzed cases. They used ensemble systems of SVM classifiers combined with simple word representations (mostly unigrams and bigrams) and achieved a 0.965 average F1 score in predicting the law area between 8 classes. Undavia et al. (2018) proposed to classify US Supreme Court documents into 15 categories through various combinations between feature representation and classification models. The best result was achieved with the combination of word2vec and Convolutional Neural Networks (CNN), which obtained an accuracy of 0.724. Research conducted by Silva et al. (2018) in Brazil tried to obtain a model to classify the documents received by the Supreme Court of Brazil into six classes using

CNN with an embedding layer. They achieved an F1 score of 0.91 and an accuracy of 0.90 with the proposed architecture.

Some studies conducted to explore more recent language model techniques were also applied to the legal area, like the one conducted by Soh et al. (2019), that compared multiple text representation techniques, such as GloVe (Pennington et al., 2014), Latent Semantic Analysis (LSA) (Deerwester et al., 1990), ULMFit (Howard and Ruder, 2018), and BERT base and large (Devlin et al., 2019), in the task of multilabel classification of 30 categories. The experiment was conducted with different portions of the dataset, using 10%, 50%, and 100% of the available data. An interesting result found was that transfer learning approaches performed well in low regime data (10% of the data) with both BERT base and large and especially with ULMFit. Glove's performance achieved better results with 50% of the data while LSA obtained the best F1 score with 100% of the data. Another experiment conducted by Clavié et al. (2021) compared the performance of the BERT model against simpler language models based on LSTM such as ULMFit. The authors' concern was to optimize computational resources, seeking to obtain good performances with relatively low training costs. From the study carried out in four different tasks of textual classification in the legal area, it was observed that the ULMFit language model trained with legal corpora presented good results in tasks that involved short texts, outperforming both generic and specialized BERT language models. An important observation was the underperformance of the LSTM-based language model for longer texts, probably due to its difficulty in retaining long chunks of information, inherent to its recurrent neural network structure. In the case of BERT models based on attention mechanisms, there was an improvement in the results for long documents. Nonetheless, the ULMFit language model proved to be useful and competitive in cases of lower textual complexity.

Elwany et al. (2019) used legal texts to fine-tune the BERT model and later applied a simple neural network for the task of predicting two classes of legal agreements. The authors showed an improved performance using fine-tuned BERT compared to the use of generic BERT, with an F1 score of 0.901. In the same line of research, Tai (2019) experimented with fine-tuning the BERT model to the legal vocabulary, specifically in the German language. The author later applied the fine-tuned model to the task of classifying case laws according to their jurisdiction (6 classes) and level of appeal (4 classes) and did not find relevant differences between the use of generic German BERT and the fine-tuned version in any of the challenges proposed. We point out that these tasks already had a high performance of 0.99 F1 score for predicting the jurisdiction and 0.97 F1-score for the level of appeal using base German BERT. Chalkidis et al. (2020) also tested the impact of legal vocabulary on the BERT language model, comparing the original BERT trained with generic vocabulary and BERT models trained with legal documents both from scratch and and by fine-tuning a pre-trained model. They tested the models in NER and classification tasks, obtaining better results using adapted legal vocabulary, especially in tasks that require in-domain knowledge as in the case of multi-label classification. Finally, Clavié and Alphonsus (2021) compared the performance of a simple baseline model (SVM) with both

generic and specialized BERT language models used for classification. Although there were great improvements using the BERT models compared to the baseline, there was not much performance increase using the specialized BERT versions when compared to the generic BERT. The result goes in the opposite direction to other experiments conducted with the specialization of language models, for example with scientific vocabulary (Beltagy et al., 2019) or specifically fine-tuned with biomedical corpora (Lee et al., 2019), in which the use of specialized BERT models have already been found to increase performance considerably. One of the hypotheses of this phenomenon raised by the authors indicates that perhaps the language models are not yet capable of capturing the depth of the legal vocabulary, whose complexity lies in its multiple legal concepts.

4 METHODOLOGY

The methodology applied in this research can be subdivided into three main parts: preprocessing, feature extraction, and training of the classifiers. Below are the detailed processes contained in each of these steps.

4.1 PREPROCESSING

The preprocessing routine applied to our data consists of techniques for normalizing the textual content. Two different methodologies for preprocessing were used, related to the two feature extraction techniques that will be presented in the next section.

4.1.1 Word2vec preprocessing

To use word2vec, we applied the same preprocessing steps used to generate the model tokens to guarantee that our database tokens corresponded to the existing tokens in the vocabulary of this model and were correctly identified for training. The steps used in this preprocessing were:

- `Tokenization`: the texts were converted into tokens, using word-level tokenization.
- `Lowercasing`: we lowercased all uppercase letters to standardize our corpora.
- `Text normalization`: we replaced URL and email structures with the tokens "URL" and "EMAIL" and replaced all numerals with the '0' token.

4.1.2 BERT preprocessing

To use BERT language model, we downloaded the resources available in the Hugging Face platform[1], as suggested by the authors. By using this resource, we obtained the preprocessing routines stored in a file called "tokenizer", which contains all the necessary text processing rules for identifying the tokens generated in the pretrained model with the tokens of the text to be entered by the user. This specific tokenizer was constructed using the BPE algorithm (Sennrich et al., 2015), using the SentecePiece library (Kudo and Richardson, 2018), and Portuguese Wikipedia articles, totalling a vocabulary of 30,000 subword units. At the end, to make the vocabulary format compatible with BERT, the result is converted to the WordPiece (Schuster and Nakajima, 2012) format.

Therefore, for our methodology, the tokenizer of the BERT Portuguese model was obtained and the sentences of our dataset were passed to it, processing the tokens and identifying them with the vocabulary of the pretrained model. The tokenization added the special tokens

[1]https://huggingface.co/neuralmind

"[CLS]" and "[SEP]", mandatory in any BERT task, and included subword units identified with the symbol "##" at the beginning of the token.

4.2 FEATURE EXTRACTION

We used two techniques for obtaining vector representations of our corpora, word2vec and BERT. For each of these models, we chose to use a pretrained version with generic vocabulary and a specialized version, trained in an unsupervised manner with the Internal Procedures Dataset. Each of these techniques was applied to our training, validation, testing, and unsupervised datasets to extract the respective features.

4.2.1 Generic word2vec

Hartmann et al. (2017) trained several algorithms for feature extraction in their research using multigenre corpus in Portuguese and made the resulting embeddings available for download[2]. We chose the word2vec model with a Skip-Gram task and a vector size of 600 as a feature extraction baseline method. The model was selected for its proximity to attributes with the chosen specialized word2vec model that will be described in the next subsection.

4.2.2 Specialized word2vec

We used the dataset of internal procedures of the Public Prosecutor's Office of the State of Paraná to train a word2vec model from scratch with the Skip-Gram task, minimum word count equal to 5, window value of 10, and vector size of 600. These parameters were chosen to take into account the results of Noguti et al. (2020) that performed a similar task in the same dataset we are using in these experiments.

4.2.3 Generic BERT

Since BERT is considered the state of the art for NLP, there are several pretrained models available on the Internet but few of them cover the Portuguese language. In the research of Souza et al. (2019), they trained a model on more than 3 million documents available on the brWaC corpus (Wagner et al., 2018) with a vocabulary of 30k subword units using BERT architecture. The model is available for download[3] in two different configurations called Portuguese BERT-base and Portuguese BERT-large, which generate different vectors of size 768 and 1024 respectively. We used the smaller model because BERT-large presents a more complex structure and increases computational costs.

Portuguese BERT-base model was trained with 8 epochs, batch size of 128, a learning rate of 1e-4, and a sequence of 512 tokens, which means the maximum size of sentence tokens in

[2]http://nilc.icmc.usp.br/embeddings

[3]https://github.com/neuralmind-ai/portuguese-bert

the input for BERT is equal to 512. The authors reported that the training of BERT-base took 7 days on a TPUv3-8 instance, indicating the high computational cost of training BERT from scratch.

For this experiment, we tested three different extraction strategies to choose the best way of representing our sentences with BERT's layers, using the first layer, the last layer, and a concatenation of the four last layers.

4.2.4 Specialized BERT

From the BERT model mentioned above, we used the Internal Procedures dataset to fine-tune it in a domain-specific vocabulary, more precisely, in the legal domain, using the script available in the transformers repository[4]. Due to computational limitations, our hyperparameter search was restricted to the learning rate values. We maintained the masking rate to 15% of the tokens, the batch size to 32, the optimizer as AdamW (Loshchilov and Hutter, 2017) and did not increase the default vocabulary.

4.3 CLASSIFIERS

We tested the supervised techniques by applying five classifiers on the training set, evaluating on the validation set, and reporting the final results on the test set. Also, experiments were carried out on the impact of data augmentation on the training set. Using the same supervised classifiers, we applied augmentation techniques in the training set to verify if there would be an increase in the performance of these techniques in the presence of a larger volume of synthetic data.

Concerning the SSL techniques, the semi-supervised dataset was used in addition to the aforementioned databases, which, in the case of UDA, also had data augmentation strategies applied to it.

4.3.1 Supervised learning

We used the four representations mentioned above (generic word2vec, specialized word2vec, generic BERT, and specialized BERT) and applied them to the linear classifiers logistic regression and support vector machines, and the ensemble classifiers random forest and gradient boosting. The logistic regression was chosen due to the fact that it is a simple and effective model, easy to train, and not very dependent on parameter optimizations. The SVM model is a popular classifier used in NLP, especially with multiclass classification problems. We also evaluated two boosted trees: random forest and gradient boosting. For optimization, we performed a grid search on the hyperparameters of the four aforementioned classifiers using the training dataset with 5-fold cross-validation.

Specifically with the BERT models, we performed a classification task by incorporating one output layer additionally to its architecture and fine-tuned in our data. Due to the great

[4]https://github.com/huggingface/transformers/tree/main/examples/tensorflow/language-modeling

complexity and need for high computational resources, the search for parameters, in this case, was reduced only to the batch size of the training and validation sets.

Finally, we tested the impact of data augmentation in purely supervised algorithms by using TF-IDF word replacement and back translation techniques in our training set and retraining the models. The validation and test sets were not modified during these experiments.

4.3.2 Semi-supervised learning

We tested three semi-supervised learning techniques, self-training, co-training, and UDA, with reduced hyperparameter search, considering the limited computational resources available. The chosen techniques seemed to be natural choices in our experiments, since we have portions of labeled and unlabeled data that can be used in self-training and multiple representations of text, suited for co-training. We also have a representation based on the BERT model, favoring the application of UDA. Another advantage was that these SSL methods were already tested in the NLP field and presented promising results, which motivated our investigations.

4.3.2.1 Self-training

To apply the self-training strategy we chose the logistic regression model, which is simple, fast, and robust to variations in hyperparameters along with the word2vec representation of our texts. The steps to perform self-training are detailed below.

1. Select T_W as the training set with word2vec features and U as the semi-supervised set.
2. Obtain a set of unlabeled examples U' by randomly sampling n examples from U without replacement.
3. Fit a logistic regression model log_reg_W with T_W.
4. Use log_reg_W to obtain the class-conditional probabilities of U' in each class and select the top c examples with highest probability in each class as long as it is greater than a threshold t.
5. Add the selected results to the training set T_W with synthetic labels predicted by the classifier.
6. Refill U' sampling the same number of examples that were selected to the training set from U.
7. Repeat the steps of 3 to 6 until reaching a stop criteria.

We monitor the accuracy of the logistic regression classifier throughout the iterations performed as well as the number of samples in the training set until reaching a stop criterion. When reaching the stop criteria, the final logistic regression trained with the augmented training set is used to evaluate the performance in the test set.

4.3.2.2 Co-training

As co-training uses two different perspectives of the data points in its structure, we used both word2vec and BERT representations to implement this technique. The steps to perform it in our data are very similar to those used in self-training and are described below.

1. Select T_W as the training set with word2vec features, T_B as the training set with BERT features and U as the semi-supervised set.
2. Obtain a set of unlabeled examples *U'* by randomly sampling *n* examples from U without replacement.
3. Fit two separated logistic regression models, log_reg_W with T_W and log_reg_B with T_B.
4. Use log_reg_W to obtain class-conditional probabilities of *U'* in each class and select the top *c* examples with highest probability in each class as long as it is greater than a threshold *t*.
5. Use log_reg_B to obtain class-conditional probabilities of *U'* in each class and select the top *c* examples with highest probability in each class as long as it is greater than a threshold *t*.
6. Concatenate the selected results of items 4 and 5, removing repeated examples and add them to the training sets T_W and T_B with synthetic labels predicted by the classifiers.
7. Refill *U'* sampling the same number of examples that were selected to the training set from U.
8. Repeat the steps of 3 to 7 until reaching a stop criteria.

At each iteration, we verify the accuracy of the two classifiers as well as the number of samples added to the training set until reaching a stop criterion. After selecting the augmented dataset, we used the two final logistic regression models to verify the performance in the test set. The two models were combined using the sum rule (Kittler et al., 1998) which simply sums the probabilities of each class and attributes the class with a greater value.

4.3.2.3 UDA

In the case of UDA, we only used the features of BERT since this technique was originally designed to play along with the BERT classifier. From the unlabeled data, we applied two data augmentation methods, TF-IDF replacement and back translation, the latter being used with Google Translation API that relies on the Neural Machine Translation model (Wu et al., 2016). With the augmented data, we applied UDA's algorithm testing the TSA variations and the training and evaluation batch sizes. We did not perform a hyperparameter optimization concerning sentence size or learning rate because of their computational cost.

4.4 EVALUATION METRICS

In this section, we will highlight the metrics used in this research and the rationale behind these choices. Similarly to Section 2.12, we have separated the metrics into global and per class, for a more focused explanation of the concepts.

4.4.1 Per-class Metrics

The metrics used to evaluate the results in each of the classes were precision, recall and F1 score. We believe that the three metrics provide a summary that allows a focused analysis of the most problematic classes, as well as the best performance areas for the model.

4.4.2 Global Metrics

Given that the training, validation, and test set were balanced, with the same number of examples for each of the 50 classes, we chose to use accuracy as the main evaluation metric in our experiments. In addition to accuracy, from a flat view, we also provide the f1 score, precision and recall metrics of the model. Lastly, we decided to use hierarchical metrics as a complement information of our classifier's performance. To better illustrate the calculation of these metrics, we provide the following examples.

Consider the case where we have the real class "Medication" related to the higher class "Health" being predicted as "Surgery", also belonging to the parent class "Health". In this example, we have:

Example 4.1

$\hat{C}_i = \{"Medication", "Health"\} = 2$

$\hat{C}'_i = \{"Surgery", "Health"\} = 2$

$\hat{C}_i \cap \hat{C}'_i = \{"Medication", "Health"\} \cap \{"Surgery", "Health"\} = \{"Health"\} = 1$

$hP = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}'_i|} = \frac{1}{2}$ $\qquad$ $hR = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}_i|} = \frac{1}{2}$

Where $\hat{C}_i$ is the set of the true class and subclass, $\hat{C}'_i$ is the set of the predicted class and subclass, and $\hat{C}_i \cap \hat{C}'_i$ is the intersection between the two sets. In this example, we also have hP and hR as the calculated hierarchical precision and recall, respectively.

In this other example, we have the observation in the real class "Medication" related to the higher class "Health" being predicted as "Divorce", which is related to the higher class "Family". Our metrics change in this case, as demonstrated below:

Example 4.2

$\hat{C}_i = \{"Medication", "Health"\} = 2$

$\hat{C}'_i = \{"Divorce", "Family"\} = 2$

$\hat{C}_i \cap \hat{C}'_i = \emptyset = 0$

$hP = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}'_i|} = \frac{0}{2}$ $\qquad$ $hR = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}_i|} = \frac{0}{2}$

Again, we have the same representations of $\hat{C}_i$ as the set of the true class and subclass, and $\hat{C}'_i$ as the set of the predicted class and subclass, but with an empty set representing the intersection $\hat{C}_i \cap \hat{C}'_i$.

The F1 score metric is calculated as the harmonic mean of the precision and recall values, as usual in other non-hierarchical metrics. We will present these metrics as complementary information to our analysis in Chapter 5.

5 EXPERIMENTS

Our research relied on several datasets obtained from MPPR's demands management system. We describe the methodology used to extract and prepare the data in this chapter. We also present the results of our experiments regarding the feature extraction choices we made as well as the supervised and semi-supervised (fine-tuned) models. We evaluated our models based on the accuracy of the classifiers considering that our supervised dataset is perfectly balanced, with the same number of examples in each of the 50 classes in the train, validation, and test partitions. In the final section, we will deepen the analysis of the results of the best model obtained, with the aid of the metrics exposed in Section 4.4.

5.1 DATASETS

We based our research on multiple datasets extracted from the demand management system of the Public Prosecutor's Office of the State of Paraná. This chapter describes the details of the extraction and preparation of the data used, with an overview shown in Figure 5.1 (in parenthesis the number of examples).

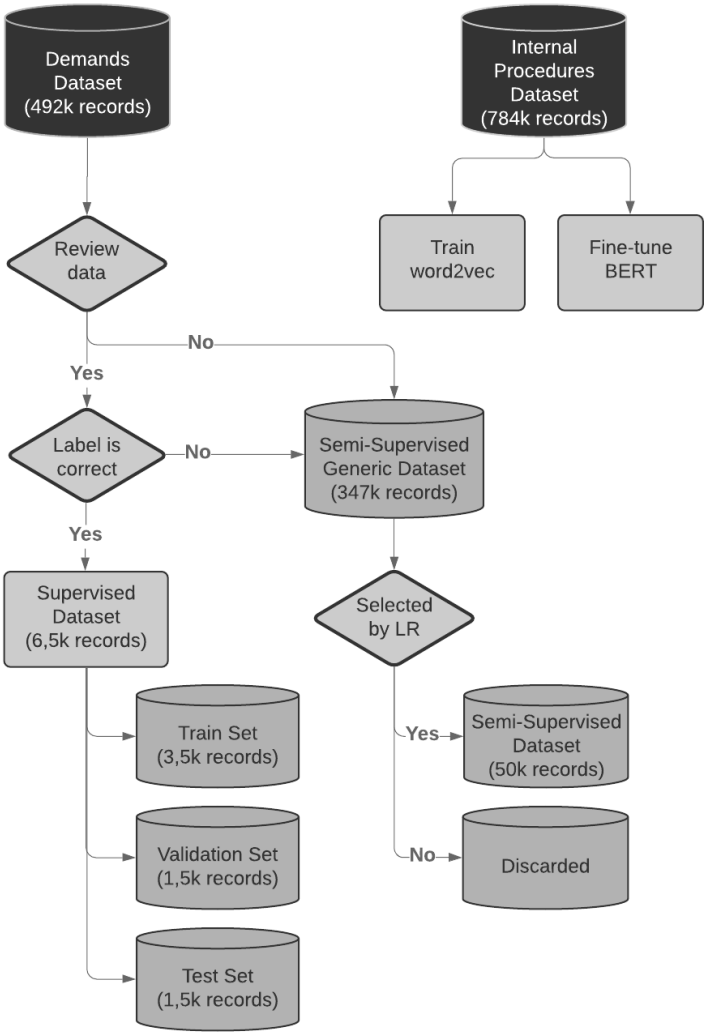
5.1.1 Labeled data

Initially, we obtained a dataset with the records of the demands of the population to the Public Prosecutor's Office of the State of Paraná, between the years 2016 and 2020, containing 492,312 observations (demands dataset). The records refer to the metadata of the registered demands, such as the name of the person who requested the service, the registration unit, the subject, and a brief description of the demand.

In the MPPR system, the subject of the demand is registered by selecting a multiple choice field containing 164 options, such as "Medicines", "Nursery Vacancy", "Theft", among other options that are used to filter the demands by their subject and forward them to the competent sector for resolution. Considering that many classes had scarce records, we decided to select 50 of the 164 classes to analyze in this research. Therefore, specialists in the law area selected 50 topics, considering the frequency of examples available in the dataset and the most consistent and less ambiguous subjects. It is important to highlight that these subjects can be hierarchically grouped into larger classes of more general subjects such as Health, Education, or Criminal, as illustrated in Figure 5.2. However, for the scope of this research, we approached the situation as a flat classification problem and not through a hierarchical structure.

Initially, a small sample was collected for exploratory analysis of the data obtained. In this experiment, it was verified a lack of standardization of the records, leading to the conclusion that it was necessary an audit and validation phase of the information preliminary to its use

Figure 5.1: Datasets scheme



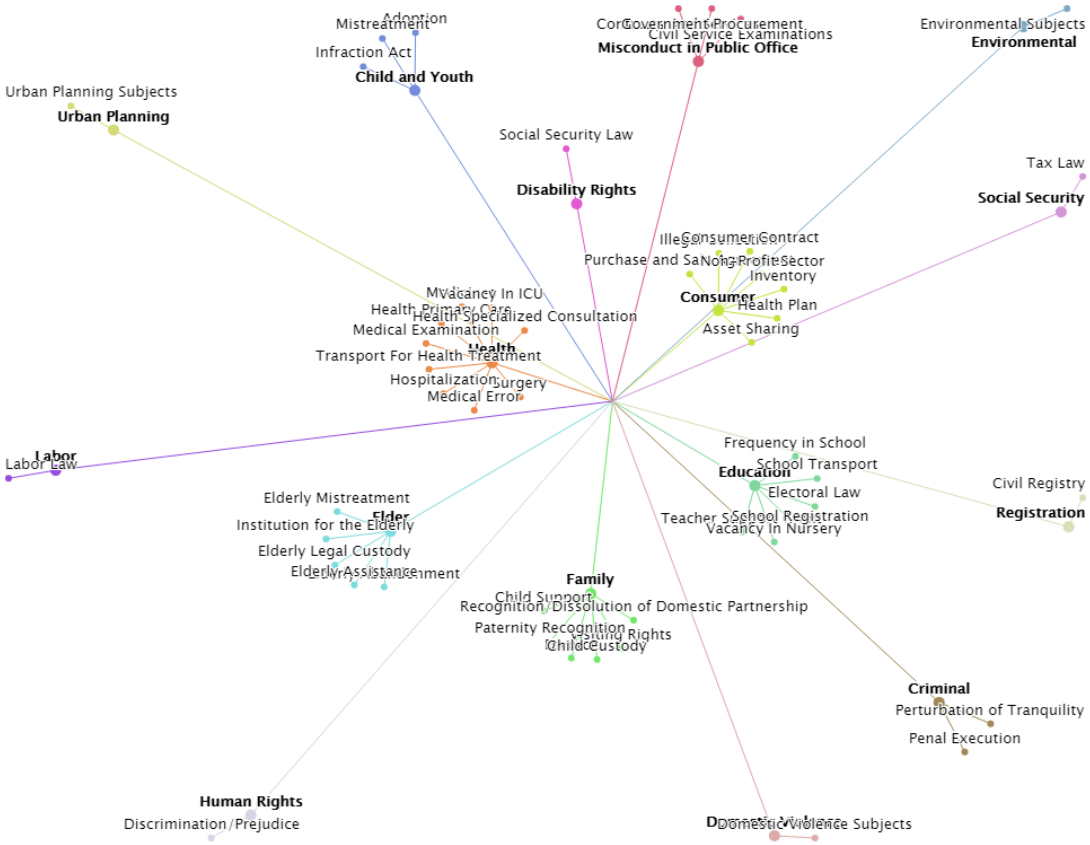
Source: Elaborated by the author (2020).

instead of the direct use of the label already registered in the system. The first phase of the analysis was carried out with the validation of 30 examples of each of the 50 subjects, totaling a dataset of 1,500 observations.

From this initial experiment, a second step of reviewing the registered information was carried out seeking to expand the size of the reliable data for this study, with the help of a team specialized in analyzing the public's demands at the MPPR. The objective was to review the descriptions of the demands and respective classification subjects, excluding the records that eventually presented the wrong label associated with them. Considering the short time available, it was determined that only 100 observations from each of the 50 themes would be validated, totaling a dataset of 5,000 observations.

In this experiment, the experts had to analyze 9,387 service records to obtain the desired number of correct samples (5,000), which means that 4,387 records were discarded due to label

Figure 5.2: Hierarchical structure of subjects in labeled data



Source: Elaborated by the author (2022).

inconsistency during the audit, a rate of 46.73% of data entry errors. This number highlights the lack of standardization and reinforces the need to automate the task in order to obtain more reliable and qualified information on the institution. Table 5.1 presents the 50 selected classes and their validation statistics, with the number of analyzed examples and the percentage of labels registered incorrectly in the system and found during the audit. The first three letters of each category correspond to the higher hierarchical class mentioned above.

After this second stage, the review of the data was completed, resulting in a final dataset containing 6,500 records with an average of 87 words per sentence. They were randomized and divided into three partitions for train, validation, and test with 3,500, 1,500, and 1,500 examples, respectively. The final training set was composed of 70 examples of each class while the validation and test set contained 30 examples per class each.

5.1.2 Unlabeled data

The registered demands that were not reviewed or presented the wrong label in the step previously described were allocated in another database called Semi-Supervised Generic Dataset. As there was not enough staff and time to review all the examples, it became opportune to study

Table 5.1: Breakdown of manually verified examples in the demands dataset

Subject	Evaluated Examples	% of Incorrect Classification
CRI - Penal Execution	129	22.48
CRI - Perturbation of Tranquility	129	22.48
CON - Illegal Collection	205	51.22
CON - Purchase and Sale Agreement	169	40.83
CON - Consumer Contract - Water, Light, Telephone, Internet, Credit Card	139	28.06
CON - Health Plan	139	28.06
CIV - Non-Profit Sector	152	34.21
CIV - Inventory	134	25.37
CIV - Asset Sharing	164	39.02
EDU - Teacher Support Grant	123	18.70
EDU - Frequency in School	277	63.90
EDU - School Registration	157	36.31
EDU - School Transport	112	10.71
EDU - Vacancy In Nursery	109	8.26
ELE - Electoral Law	131	23.67
FAM - Child Support	124	19.35
FAM - Visiting Rights	149	32.89
FAM - Divorce	132	24.24
FAM - Child Custody	131	23.66
FAM - Paternity Recognition	117	14.53
FAM - Recognition / Dissolution of Domestic Partnership	133	24.81
HUM - Discrimination / Prejudice	142	29.58
ELD - Elderly Abandonment	505	80.20
ELD - Elderly Assistance	376	73.40
ELD - Elderly Legal Custody	468	78.63
ELD - Hospitalization in a Long Term Care Institution for the Elderly	447	77.63
ELD - Elderly Mistreatment	455	78.02
SOC - Social Security Law	216	53.70
DIS - Disability Rights	327	69.42
HEA - Health Primary Care	263	61.98
HEA - Surgery	108	7.41
HEA - Health Specialized Consultation	108	7.41
HEA - Medical Error	123	18.70
HEA - Medical Examination	113	11.50
HEA - Hospitalization - Voluntary, Involuntary, Compulsory	118	15.25
HEA - Medicines	121	17.35
HEA - Transport For Health Treatment	107	6.54
HEA - Vacancy In ICU	112	10.71
LAB - Labor Law	201	50.25
URB - Urban Planning	241	58.51
CHI - Adoption of Child and Youth	132	24.24
CHI - Infraction Act by Child and Youth	189	47.09
CHI - Mistreatment Of Child and Youth	232	56.90
ENV - Environment	141	29.08
MIS - Civil Service Examinations	131	23.66
MIS - Contracts And Services in Public Sector	227	55.95
MIS - Government Procurement	178	43.82
REG - Civil Registry - Loss / Theft / Alteration	211	52.61
TAX - Tax Law	160	37.50
DOM - Domestic Violence	180	44.44
Total	**9,387**	**46.73**

Source: Elaborated by the author (2020).

semi-supervised techniques that took advantage of such data. After cleaning the dataset by removing duplicates, we ended up with 347,977 examples in the dataset.

Considering that the examples contained in this partition were not limited to the 50 selected categories, but of all 164 possible classes in the record system, we performed a domain-relevance data filtering by fitting a simple logistic regression model with word2vec specialized features to the training set to infer the labels of the Semi-Supervised Generic Dataset and select the examples with higher confidence in the model. That way, for each of the 50 categories, we sorted the examples by their classification probabilities of being in that category and selected 1,000 examples of each class with a higher probability. We ended up with the final Semi-Supervised Dataset containing 50,000 observations which were used in all our SSL experiments.

We also obtained a dataset containing 784,358 records of the institution's internal procedures (Internal Procedures Dataset). The records corresponded to various metadata of the registered procedures, including a field filled generally with a short text, with the description of the subject in the document. As we verify an approximation of the vocabulary of the Demands Dataset with the vocabulary of the Internal Procedures Dataset, with more than 80% of the words of the first present in the second, we found appropriate their use to train and fine-tune language models that were used and will be detailed in the following sections.

5.2 FEATURE EXTRACTION

Initially, we needed to create four distinct routines to preprocess and extract the features of our texts, related to word2vec and BERT models, both in generic and specialized versions. The word2vec models were already pretrained and had their parameters set as detailed in Section 4.2.

For the generic BERT model, the majority of the parameters were maintained, lacking only a strategy for feature extraction. Since BERT has an architecture with twelve identical stacked layers, each one can be used for feature extraction. To choose the best way to represent the sentences using the BERT model, we tested three different extraction strategies: (1) using the last layer, (2) using the first layer, and (3) using a concatenation of the four last layers. To evaluate the performance we chose a simple logistic regression model, which is less dependent on hyperparameter optimization, and applied the three extraction techniques using the generic BERT model in the train and validation sets. Table 5.2 presents the validation results in our datasets.

Table 5.2: Evaluation of BERT using different layers and logistic regression in the validation set. In bold the best accuracy

Layer	**Accuracy**
Last Layer	**0.723**
First Layer	0.684
Four Last Layers	0.706

Source: Elaborated by the author (2020).

The concatenation of the four last layers did not outperform other representation layers in our datasets as was the case in the experiments done by Devlin et al. (2019). The best results

were achieved using the last layer as a feature extractor, therefore we chose this representation for feature extraction with BERT models.

With the feature extraction strategy set, the generic BERT model was already defined. For the specialized BERT model, we still needed to set the hyperparameters for its fine-tuning. To fine-tune BERT there are several hyperparameters we can vary, namely the batch size, the number of epochs, and the learning rate, among others. We only tested three different learning rates and a small batch size of 32 due to memory limitations in the GPU and kept other parameters as default. For each learning rate of 1e-4, 2e-5, and 5e-5 we retrained the model using the Internal Procedures dataset to expand the knowledge of the language model into most complex and specialized structures in the legal area. After the fine-tuning, we used the last layer to extract the features, according to the best strategy found in the previous experiment. With the features extracted from each of the three fine-tuned BERT models, we trained a logistic regression model using the training set, with the accuracy of the validation set presented in Table 5.3.

Table 5.3: Evaluation of BERT using different learning rates and logistic regression in the validation set. In bold the best accuracy

Learning Rate	**Accuracy**
1e-4	**0.690**
2e-5	0.677
5e-5	0.683

Source: Elaborated by the author (2020).

From the results obtained in our experiments, we stipulated four final feature extraction strategies as described below.

- Generic word2vec with default parameters of Hartmann et al. (2017) and average word embeddings as sentence representation.
- Specialized word2vec with default parameters of Noguti et al. (2020) and average word embeddings as sentence representation.
- Generic BERT with default parameters of Souza et al. (2019) and last layer as sentence representation.
- Specialized BERT with a learning rate of 1e-4 and last layer as sentence representation.

The specialized representations obtained in this study will be shared with the community, enabling their use in other experiments in the legal domain that share terminologies with our vocabulary. The original data that were used to train these models and the classifiers, unfortunately, cannot be shared at the moment, as they need to be anonymized to protect sensitive information from MPPR registered demands.

5.3 SUPERVISED LEARNING

With the four feature extraction strategies defined, we initially selected four supervised techniques to be tested, totaling sixteen combinations of features and models to fine tune in the original training set. In order to verify the effectiveness of data augmentation techniques alone, we also applied the sixteen combinations of models and features to the training set with the addition of synthetic data obtained from the back translation method of augmentation. This operation doubled the amount of data for training, considering that each example in the training set had an augmented version. For each of these thirty-two configurations, we performed a grid search with 5-fold cross-validation and optimized the hyperparameters as detailed in Appendix A. After obtaining the best parameters, we trained the final models one last time and evaluated the results in the test set, with the accuracy of each classifier presented in Table 5.4.

From the results, we can see that, using the original data, there was a better performance in the logistic regression model, followed by the SVM, with the random forest being the worst of the four models tested, regardless of the feature extraction strategy. This order of performance is maintained with the use of augmented data, however we verified an increase performance of the boosted tree models with the inclusion of synthetic data compared to the same classifiers trained exclusively with the original data, but without reaching the best performance in general. The use of the original data combined with the logistic regression model presents the highest performance of all classifiers, achieving 78% of accuracy. Another interesting result is the performance of the different feature extraction techniques. For all models, there is a higher performance when using the specialized word2vec version initially indicating that this more simple extraction technique fits better with these less complex models. This also indicates that word2vec greatly benefits from training in specific corpora, in this case, the legal language. The same cannot be said for BERT, which showed a drop in performance in all cases using the specialized version. It should be noted here that our version of BERT used sub-optimal parameters due to the high computational cost of training such model from scratch, which can harm the performance of the model. Even so, the specialized BERT model achieved results close to its generic version but did not outperform it in any classifier. We hypothesize that the use of the last layer of the BERT model as a feature through the subword tokens did not lead to an adequate representation of the sentences in conjunction with these more simplistic models. We also point out that Reimers and Gurevych (2019) demonstrated in their experiments a poor performance of BERT embeddings compared to GloVe embeddings in most of the experiments conducted by them.

Still in the supervised experiments, we tested the use of the BERT model in the classification task, simply adding an output layer in its architecture to predict the 50 existing categories of our dataset. We only trained with a small sequence length of 128 (instead of the maximum of 512) due to GPU limitations and limited the search for hyperparameters to the learning rate variations between 4e-5, 3e-5, and 2e-5, and the batch size of the training set

Table 5.4: Results of supervised classifiers in the test set. In bold the best accuracy for each classifier and data strategy

	Classifier	Feature	Accuracy
Original Data	Logistic Regression	Generic word2vec	0.755
		Specialized word2vec	**0.781**
		Generic BERT	0.719
		Specialized BERT	0.693
	SVM	Generic word2vec	0.705
		Specialized word2vec	**0.752**
		Generic BERT	0.696
		Specialized BERT	0.670
	Random Forest	Generic word2vec	0.567
		Specialized word2vec	**0.639**
		Generic BERT	0.571
		Specialized BERT	0.565
	Gradient Boosting	Generic word2vec	0.630
		Specialized word2vec	**0.687**
		Generic BERT	0.603
		Specialized BERT	0.597
Augmented Data	Logistic Regression	Generic word2vec	0.724
		Specialized word2vec	**0.744**
		Generic BERT	0.696
		Specialized BERT	0.685
	SVM	Generic word2vec	0.706
		Specialized word2vec	**0.753**
		Generic BERT	0.697
		Specialized BERT	0.670
	Random Forest	Generic word2vec	0.572
		Specialized word2vec	**0.649**
		Generic BERT	0.581
		Specialized BERT	0.541
	Gradient Boosting	Generic word2vec	0.638
		Specialized word2vec	**0.707**
		Generic BERT	0.595
		Specialized BERT	0.577

Source: Elaborated by the author (2022).

between 16, 32 and 64. Table 5.5 presents the results using both the generic and the specialized models.

We can see from the results that the use of BERT's architecture for classification obtained an improved accuracy when compared to previous models, both the generic and specialized versions. The generic version of BERT outperformed our specialized version, although it achieved close results. Some reasons for this behavior could be the use of an insufficient amount of specialized corpora and the reduced investigation in the model optimization due to low

Table 5.5: Results of BERT classifiers in the test set. In bold the best accuracy for each BERT version

	Learning Rate	Batch Size	Accuracy
Generic BERT	4e-5	16	0.799
		32	0.797
		64	0.799
	3e-5	16	0.799
		32	**0.801**
		64	**0.801**
	2e-5	16	0.787
		32	0.795
		64	0.782
Specialized BERT	4e-5	16	0.795
		32	0.799
		64	0.793
	3e-5	16	0.790
		32	0.796
		64	0.783
	2e-5	16	0.787
		32	0.798
		64	**0.800**

Source: Elaborated by the author (2022).

computational resources. This result is also related to the research of Clavié and Alphonsus (2021), with regard to the low performances of specialized models in the legal area when compared to generalized language models.

5.4 SEMI-SUPERVISED LEARNING

We tested three semi-supervised strategies to improve our classification accuracy, using self-training, co-training, and UDA, with the detailed results presented in the next subsections.

5.4.1 Self-training and co-training

Since self-training and co-training are very similar strategies for pseudo-labeling and were applied with the same parameter dependencies in this study, we will present the results of both SSL methods together, allowing a direct comparison of their performance.

As mentioned in Section 4.3.2.1, we used the logistic regression model as the classifier to obtain the labels of our semi-supervised set in both techniques. For self-training, we used the specialized word2vec for text representation due to the good performance presented in the previous section.

Considering that co-training demands different views of the data, we used word2vec with specialized configuration and BERT with generic configuration to represent our two distinct views of the data, as they were the best embeddings with the logistic regression model in the supervised tests. We then trained the classifiers in each of the representations and iterated over the semi-supervised set to search for candidates to enter the training set with synthetic labels provided by the models. With the final selected data, we trained two logistic regression models and used the sum rule to combine the predictions and evaluate the performance in the test set.

For both techniques, as stop criterion, we analyzed the accuracy of the validation set in each iteration and stopped the algorithm when there was a decrease in accuracy with the addition of new pseudo-labeled data. In the case of co-training, the accuracy of both classifiers should decrease to stop the iterations. We tested some variations in the parameters c, the number of examples of each class to select in each iteration, and t, the confidence threshold. We did not seek an optimization of the parameters but instead used some random values to check if they had any impact on the performance of the techniques. The experiments and their results in the test set are described in Table 5.6.

Table 5.6: Results of self-training and co-training in the test set. In bold the best accuracy for each technique

Probability Threshold	Number of Examples	Self-Training Accuracy	Co-Training Accuracy
0.4	1	0.741	0.747
	5	0.741	**0.753**
	10	0.746	**0.753**
	50	0.746	**0.753**
0.5	1	**0.747**	0.747
	5	0.741	**0.753**
	10	0.745	0.749
	50	0.743	**0.753**
0.7	1	0.746	**0.753**
	5	0.746	**0.753**
	10	0.746	**0.753**
	50	0.746	0.728

Source: Elaborated by the author (2020).

As can be seen from the results above, there was little variation with the change of the investigated parameters, which had no impact at all on many of the tested values. It is important to point out that, in the case of self-training, the logistic regression model was not very confident about the predictions of the semi-supervised set, which led to a small portion of the data passing through the threshold stipulated. For both techniques, the decrease of the accuracy occurred at the beginning of the iterations with few examples added in the training set, not exceeding a thousand examples added in any of the experiments.

Regarding the performance in general, self-training presented worse results when compared to the supervised logistic regression version and with the co-training algorithm while co-training did not outperform the logistic regression model trained exclusively with supervised data and the word2vec representations. The co-training algorithm presented a slightly better result than the self-training technique, but no significant improvements were observed by using these SSL techniques compared to the corresponding purely supervised versions. The best accuracy was obtained using co-training, achieving a value of 0.753 in 8 of the 12 experiments conducted.

5.4.2 UDA

We tested the UDA algorithm in both BERT models (generic and specialized) and analyzed the impact of different TSA and data augmentation strategies. Since our computational resources were limited, we only tested a sequence length of 128 and a learning rate of 3e-5, varying the training batch size between the values 32, 64, and 128. Tables 5.7 presents the results in terms of accuracy in the test set with the generic and specialized BERT models.

Table 5.7: Results of UDA with generic BERT in the test set. In bold the best accuracy for each BERT model

	TSA	Data Augmentation	Accuracy
Generic BERT	Linear	TF-IDF Replacement	0.799
		Back Translation	**0.803**
	Exp	TF-IDF Replacement	0.796
		Back Translation	0.797
	Log	TF-IDF Replacement	0.794
		Back Translation	**0.803**
Specialized BERT	Linear	TF-IDF Replacement	0.803
		Back Translation	0.805
	Exp	TF-IDF Replacement	0.796
		Back Translation	0.801
	Log	TF-IDF Replacement	0.803
		Back Translation	**0.807**

Source: Elaborated by the author (2022).

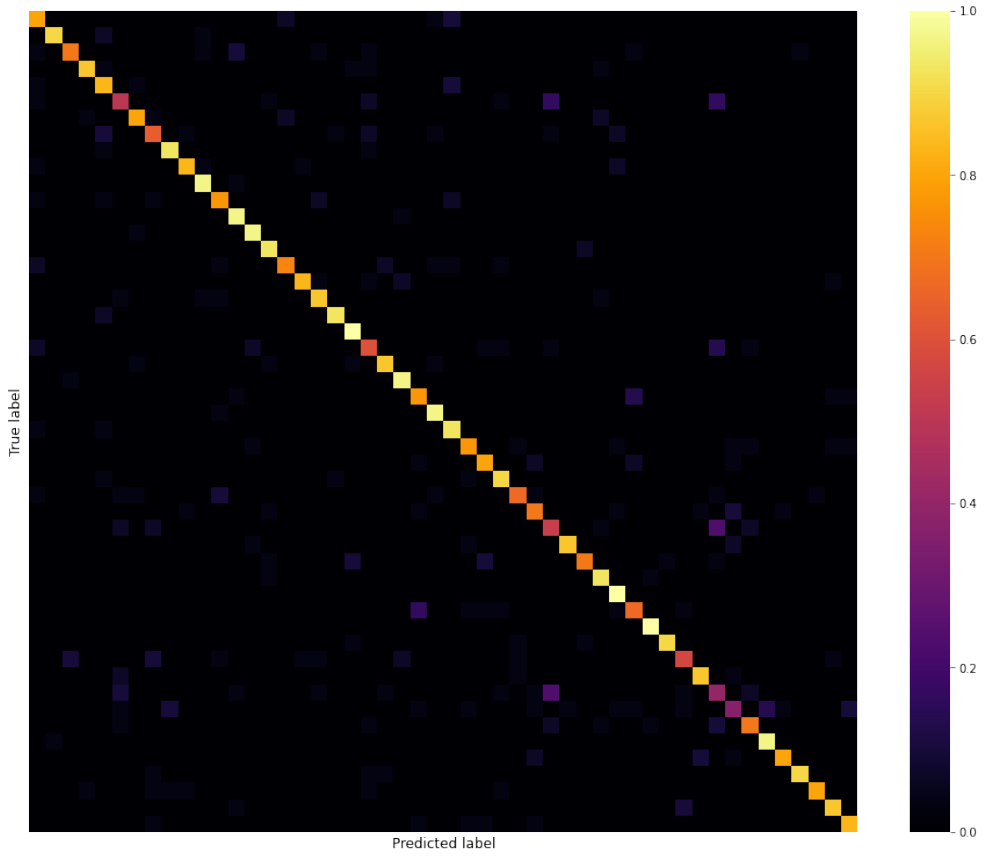
The results show an improvement in the performance of the classifiers compared to all previously tested models. Regarding the data augmentation technique, it is possible to verify that in all the combinations of parameters, back translation increased the accuracy compared to the TF-IDF replacement strategy. We also note better results with the linear and logarithmic TSA while the exp schedule did not work well with our data, underperforming in all tested cases and suggesting that the model does not have overfitting issues. Overall, we noticed an increase in accuracy with the use of the specialized BERT model, behavior contrary to the results found in the linear, ensemble and BERT purely supervised models.

In conclusion, the best performance was achieved using the log TSA along with specialized BERT and back-translation augmentation, with an accuracy of 80.7%. This was also the best performance in all experiments carried out in this research so far, demonstrating the strength of the combined use of semi-supervised learning, transfer learning, and data augmentation in one strategy. It is also a promising result considering that, in the original paper, limited results were obtained with UDA in the presence of multiple labels (they tested with a maximum of 5 labels) while our dataset has 50 classes and achieved good results with a relatively small number of labeled examples.

5.5 DETAILED ANALYSIS: UDA WITH BACK TRANSLATION AND LOG TSA

In this section we will explore in more depth the results of the best model obtained, that is, the UDA with back translation augmentation and log TSA. For this purpose, we will initially carry out an analysis of the individual performance of the 50 classes predicted by the model. Figure 5.3 presents the confusion matrix of our 50 classes for an overview of the classification results.

Figure 5.3: Confusion matrix produced by the UDA model



Source: Elaborated by the authors (2022).

From the confusion matrix, we can notice several well-performing classes. A few classes presented more relevant errors, while there was a greater degree of confusion between specific categories. For further detailing of the classes' performances, we present Table 5.8, with the 50

classes and their respective precision, recall, and F1 score, as well as the global flat metrics in the last line.

Table 5.8: Per class metrics for the UDA model

Subject	Precision	Recall	F1-Score
CRI - Penal Execution	0.938	1.000	0.968
CRI - Perturbation of Tranquility	0.867	0.867	0.867
CON - Illegal Collection	0.714	0.667	0.690
CON - Purchase and Sale Agreement	0.800	0.800	0.800
CON - Consumer Contract - Water, Light, Telephone, Internet, Credit Card	0.719	0.767	0.742
CON - Health Plan	0.867	0.867	0.867
CIV - Non-Profit Sector	0.963	0.867	0.912
CIV - Inventory	0.848	0.933	0.889
CIV - Asset Sharing	0.875	0.700	0.778
EDU - Teacher Support Grant	0.933	0.933	0.933
EDU - Frequency in School	0.757	0.933	0.836
EDU - School Registration	0.676	0.833	0.746
EDU - School Transport	0.789	1.000	0.882
EDU - Vacancy In Nursery	0.964	0.900	0.931
ELE - Electoral Law	0.879	0.967	0.921
FAM - Child Support	0.867	0.867	0.867
FAM - Visiting Rights	0.853	0.967	0.906
FAM - Divorce	0.833	1.000	0.909
FAM - Child Custody	0.846	0.733	0.786
FAM - Paternity Recognition	0.824	0.933	0.875
FAM - Recognition / Dissolution of Domestic Partnership	0.964	0.900	0.931
HUM - Discrimination / Prejudice	0.960	0.800	0.873
ELD - Elderly Abandonment	0.500	0.533	0.516
ELD - Elderly Assistance	0.364	0.400	0.381
ELD - Elderly Legal Custody	0.621	0.600	0.610
ELD - Hospitalization in a Long Term Care Institution for the Elderly	0.778	0.700	0.737
ELD - Elderly Mistreatment	0.577	0.500	0.536
SOC - Social Security Law	0.818	0.900	0.857
DIS - Disability Rights	0.655	0.633	0.644
HEA - Health Primary Care	0.739	0.567	0.642
HEA - Surgery	0.806	0.967	0.879
HEA - Health Specialized Consultation	0.853	0.967	0.906
HEA - Medical Error	0.840	0.700	0.764
HEA - Medical Examination	0.926	0.833	0.877
HEA - Hospitalization - Voluntary, Involuntary, Compulsory	0.812	0.867	0.839
HEA - Medicines	0.964	0.900	0.931
HEA - Transport For Health Treatment	0.893	0.833	0.862
HEA - Vacancy In ICU	0.879	0.967	0.921
LAB - Labor Law	0.821	0.767	0.793
URB - Urban Planning	0.724	0.700	0.712
CHI - Adoption of Child and Youth	0.857	0.800	0.828
CHI - Infraction Act by Child and Youth	0.767	0.767	0.767
CHI - Mistreatment Of Child and Youth	0.686	0.800	0.738
ENV - Environment	0.923	0.800	0.857
MIS - Civil Service Examinations	0.875	0.933	0.903
MIS - Contracts And Services in Public Sector	0.550	0.367	0.440
MIS - Government Procurement	0.879	0.967	0.921
REG - Civil Registry - Loss / Theft / Alteration	0.929	0.867	0.897
TAX - Tax Law	0.833	0.833	0.833
DOM - Domestic Violence	0.833	0.667	0.741
Flat Global Metrics	**0.809**	**0.807**	**0.805**

Source: Elaborated by the author (2022).

Initially analyzing the F1 score as a more general metric, we found out that the worst performances occur in the larger subject related to the elderly, with 4 of the 5 existing classes presenting an F1 score below 0.7. When verifying the accuracy and recall of these classes, we found that both have a low value, indicating that the model has difficulties in differentiating both false positives and false negatives. On the other hand, some categories presented high F1 scores, with three of them having a recall equal to 1, indicating the model's quality in correctly identifying these classes. Overall, we found a balance between accuracy and recall, with these global metrics being equivalent for the model in question.

Complementing the analysis of the individual performance of the 50 classes predicted by the model, we compared the accuracy of the classifier with the results obtained in the manual verification. We considered the manually verified data as a ground truth, measuring the performance of the legal clerks in each category, and directly compared with the performance of the model. In Figure 5.4, we illustrate the accuracy achieved by humans for each class (ground truth), indicated with the red marker, as well as the model's accuracy for the respective categories, indicated with the blue marker. We can easily visualize that, when the red marker is to the left of the blue marker in a given category, the model's accuracy is higher than the real accuracy and vice versa. Keeping that in mind, we visualize in the chart that only six categories present a rate of correct labels higher than the humans' annotation, which are related to classes with increased assertiveness in real life. We can also verify that, despite the underwhelming values of the F1 score for certain classes, there is still a great improvement compared to the real-life performance, which indicates the great utility of the classifier even with relatively low statistics.

Finally, we present the global metrics of accuracy, recall, and F1 score for hierarchical data. For a better comparative effect, we selected here two more models with good performance to compare these global metrics: the logistic regression model with specialized word2vec, and the generic BERT model with a learning rate of 3e-5 and batch size of 32. Table 5.9 presents the comparative results of flat and hierarchical metrics in our three best models.

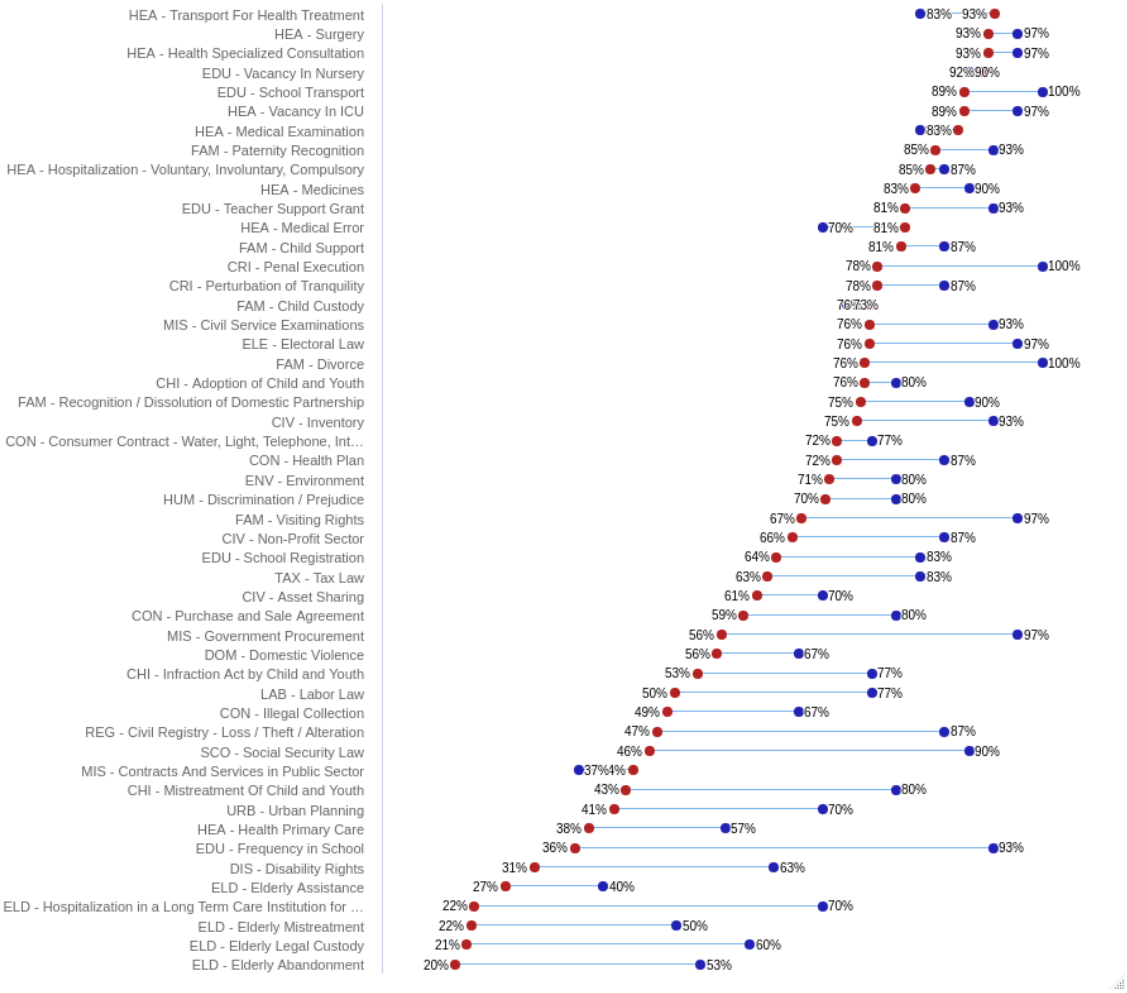
Table 5.9: Global metrics for the best classifiers

Type	Metric	Logistic Regression	BERT	UDA
Flat	Precision	0.787	0.804	0.809
	Recall	0.783	0.801	0.807
	F1 Score	0.781	0.800	0.805
Hierarchical	Precision	0.827	0.841	0.843
	Recall	0.828	0.842	0.844
	F1 Score	0.827	0.840	0.843

Source: Elaborated by the author (2022).

From Table 5.9, it can be seen that the three models presented high hierarchical metrics compared to the flat metrics, possibly indicating that the three classifiers are capable of capturing

Figure 5.4: Dumbbell chart of comparative performance between humans and the classifier, where blue represents the model accuracy and red represents human accuracy



Source: Elaborated by the authors (2022).

the underlying structure of the data and mislabeling the observations between the large classes with less frequency. Even so, we verified that in all metrics the UDA model presented higher values than the other two classifiers, reaffirming the quality of this model for the selected task, with an accuracy close to 81%. It is worth noting that the accuracy calculations for the top 3 classes show an increase of more than 10%, reaching 92% while the top 5 reaches a percentage of 94%, both considerably high value of accuracy.

6 CONCLUSIONS

This research sought to gather some of the most recent techniques in NLP associated with optimization in small datasets, namely the use of semi-supervised learning techniques through the use of self-training, co-training, and UDA algorithms, the impacts of transfer learning through word2vec embeddings and BERT language models and data augmentation processes using back translation and TF-IDF replacement.

In the problem addressed here, we were faced with a small labeled dataset and a lot of unlabeled data, and investigated whether it is possible to optimize the prediction results in the supervised dataset by leveraging the implicit knowledge of the unsupervised dataset. More specifically, our study investigated the specific legal domain, which has a particular vocabulary suitable for the study of adaptations in language models.

Initially, we analyzed different feature extraction approaches, with four variations between techniques and corpora used for training. Throughout our investigation, we noticed the relationship between simpler feature extraction techniques and simpler classifiers to achieve better results. We could notice a great impact in the use of specific domain vocabulary for the word2vec technique, but unfortunately, we were not able to obtain the same results with the BERT model, probably due to the lack of computational resources. Even so, we believe that the models can be useful for future investigations, reason why we open sourced both Portuguese models fine-tuned in the legal area for the community.

Concerning the purely supervised classifiers, the simplified linear models Logistic Regression and Support Vector Machine presented better performance compared to the tree-based bagging and boosting techniques. However, the best result was obtained using the BERT model with a final classification layer. Moreover, we noticed that the simple addition of augmented data within the supervised models was not enough to increase accuracy, indicating the need for a more refined strategy to take advantage of these synthetic data.

Regarding the semi-supervised models, we tested two simpler techniques, self-training and co-training, and no performance improvement was observed in any of the models. It is also notable that the varied hyperparameters in these classifiers did not show any impact on modifying the evaluated metrics, so a future investigation could focus more on modifying the classifier selection or the data selection strategies rather than on optimizing these parameters. Finally, the last studied model, Unsupervised Data Augmentation (UDA), was a combination of transfer learning, data augmentation, and semi-supervised learning. It achieved the best performance during the research, with an accuracy of 80.7%.

Our results showed that each of the optimization techniques tested had positive and negative points when applied to our data, but it was the combined use of all of them that obtained the best results. We were able to improve accuracy significantly by using the UDA technique,

outperforming legal clerks, that presented a record error rate of 46.7%. More than that, we also verified that, when using the top 3 predictions of the model, the accuracy increased considerably, reaching 92%. Therefore, the experimental results demonstrate the usefulness of the technique applied to a real problem, which makes it possible to implement the classifier in a real-time system, guaranteeing an interpretative standardization regarding the concepts of legal demands, as well as internal referral to the competent sectors, which leads to concrete benefits for the population and the modernization of the Brazilian public sector.

REFERENCES

Abulaish, M. and Sah, A. (2019). A text data augmentation approach for improving the performance of CNN. In *2019 11th International Conference on Communication Systems Networks (COMSNETS)*, pages 625–630.

Alammar, J. (2018). The illustrated transformer. Accessed in 2022/06/30.

Anaby-Tavor, A., Carmeli, B., Goldbraich, E., Kantor, A., Kour, G., Shlomov, S., Tepper, N., and Zwerdling, N. (2020). Do not have enough data? Deep learning to the rescue! *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:7383–7390.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Barz, B. and Denzler, J. (2020). Deep learning on small datasets without pre-training using cosine loss. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1360–1369.

Beltagy, I., Lo, K., and Cohan, A. (2019). SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, page 92–100. Association for Computing Machinery.

Bouillot, F., Poncelet, P., and Roche, M. (2014). Classification of small datasets: Why using class-based weighting measures? In Andreasen, T., Christiansen, H., Cubero, J.-C., and Raś, Z. W., editors, *Foundations of Intelligent Systems*, pages 345–354. Springer International Publishing.

Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., and Androutsopoulos, I. (2020). LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. The MIT Press. http://www.acad.bg/ebook/ml/MITPress-%20SemiSupervised%20Learning.pdf.

Chen, J., Feng, J., Sun, X., and Liu, Y. (2019). Co-training semi-supervised deep learning for sentiment classification of mooc forum posts. *Symmetry*, 12:8.

Clavié, B. and Alphonsus, M. (2021). The unreasonable effectiveness of the baseline: Discussing SVMs in legal text classification. *CoRR*, abs/2109.07234.

Clavié, B., Gheewala, A., Briton, P., Alphonsus, M., Laabiyad, R., and Piccoli, F. (2021). Legalmfit: Efficient short legal text classification with LSTM language model pre-training. *CoRR*, abs/2109.00993.

CNJ - Conselho Nacional de Justiça (2020). Justiça em números 2020: ano-base 2019. https://www.cnj.jus.br/wp-content/uploads/2020/08/WEB-V3-Justi%C3%A7a-em-N%C3%BAmeros-2020-atualizado-em-25-08-2020.pdf. Accessed in 2020/08/27.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Csányi, G. M. and Orosz, T. (2021). Comparison of data augmentation methods for legal document classification. *Acta Technica Jaurinensis*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Dehghani, M., Mehrjou, A., Gouws, S., Kamps, J., and Schölkopf, B. (2017). Fidelity-weighted learning. *CoRR*, abs/1711.02799.

Devi, M. and Saharia, N. (2020). Learning adaptable approach to classify sentiment with incremental datasets. *Procedia Computer Science*, 171:2426–2434.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Elwany, E., Moore, D., and Oberoi, G. (2019). BERT goes to law school: Quantifying the competitive advantage of access to large legal corpora in contract understanding. *CoRR*, abs/1911.00473.

Feng, J. and Mohaghegh, M. (2021). Hybrid model of data augmentation methods for text classification task. In *Proceedings of the 13th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - KMIS,*, pages 194–197. INSTICC, SciTePress.

Friedman, J. (2000). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29.

Gage, P. (1994). A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Glaser, I., Sadegharmaki, S., Komboz, B., and Matthes, F. (2021). Data scarcity: Methods to improve the quality of text classification. In *ICPRAM*.

Hartmann, N. S., Fonseca, E., Shulby, C. D., Treviso, M. V., Rodrigues, J. S., and Aluísio, S. M. (2017). Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In *Proceedings of Symposium in Information and Human Language Technology*. Sociedade Brasileira de Computação.

Ho, T. K. (1995). Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, page 278, USA. IEEE Computer Society.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Huong, T. and Truong Hoang, V. (2020). A data augmentation technique based on text for vietnamese sentiment analysis. In *Proceedings of the 11th International Conference on Advances in Information Technology*, pages 1–5.

Jurafsky, D. and Martin, J. H. (2019). *Speech and Language Processing*. Unpublished manuscript, Stanford University. `https://web.stanford.edu/~jurafsky/slp3/edbook_oct162019.pdf`.

Kiritchenko, S., Nock, R., and Famili, F. (2006). Learning and evaluation in the presence of class hierarchies: Application to text categorization. In *Advances in Artificial Intelligence*, volume 4013, pages 395–406. Springer Berlin Heidelberg.

Kittler, J., Hatef, M., Duin, R., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:226–239.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference*

on Empirical Methods in Natural Language Processing: System Demonstrations, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. (2019). BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

Loshchilov, I. and Hutter, F. (2017). Fixing weight decay regularization in Adam. *CoRR*, abs/1711.05101.

Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317.

Martindale, H., Rowland, E., Flower, T., and Clews, G. (2020). Semi-supervised machine learning with word embedding for classification in price statistics. *Data & Policy*, 2:e12.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013.

Noguti, M. Y., Vellasques, E., and Oliveira, L. S. (2020). Legal document classification: An application to law area prediction of petitions to public prosecution service. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Shleifer, S. (2019). Low resource text classification with ULMFit and backtranslation. *CoRR*, abs/1903.09244.

Silva, N., Braz, F., and de Campos, T. (2018). Document type classification for Brazil's supreme court using a convolutional neural network. In *The Tenth International Conference on Forensic Computer Science and Cyber Law-ICoFCS,*, pages 7–11.

Soh, J., Lim, H. K., and Chai, I. E. (2019). Legal area classification: A comparative study of text classifiers on Singapore supreme court judgments. In *Proceedings of the Natural Legal Language Processing Workshop 2019*, pages 67–77. Association for Computational Linguistics.

Souza, F., Nogueira, R., and Lotufo, R. (2019). Portuguese named entity recognition using BERT-CRF. *arXiv preprint arXiv:1909.10649*.

Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

Sulea, O.-M., Zampieri, M., Malmasi, S., Vela, M., Dinu, L. P., and van Genabith, J. (2017). Exploring the use of text classification in the legal domain. In *Proceedings of 2nd Workshop on Automated Semantic Analysis of Information in Legal Texts (ASAIL)*, London, United Kingdom.

Tai, C. M. Y. (2019). Effects of inserting domain vocabulary and fine-tuning BERT for german legal language. Master's thesis, Faculty of Electrical Engineering, Mathematics and Computer Science - University of Twente, Enschede - Netherlands.

Undavia, S., Meyers, A., and Ortega, J. E. (2018). A comparative study of classifying legal documents with neural networks. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 515–522.

van Engelen, J. E. and Hoos, H. H. (2019). A survey on semi-supervised learning. *Machine Learning*, pages 373–440. `https://link.springer.com/content/pdf/10.1007/s10994-019-05855-6.pdf`.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Waegel, D. (2013). A survey of bootstrapping techniques in natural language processing.

Wagner, J., Wilkens, R., Idiart, M., and Villavicencio, A. (2018). The brWaC corpus: A new open resource for brazilian portuguese. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*.

Wei, J. W. and Zou, K. (2019). EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR*, abs/1901.11196.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Xie, Q., Dai, Z., Hovy, E. H., Luong, M., and Le, Q. V. (2019). Unsupervised data augmentation for consistency training. *CoRR*, abs/1904.12848.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

APPENDIX A – HYPERPARAMETER INVESTIGATION

We used a five-fold cross-validation in all four classifiers listed below, using the scikit-learn[1] package. Each classifier was tested for the four feature representations and with the original and augmented training data, totalling 8 grid searches for each configuration. Below we present the hyperparameters investigated, equal in all eight versions, and the best value obtained with the grid search technique.

A.1 LOGISTIC REGRESSION

For logistic regression, we tested four hyperparameters and 300 models for each combination of feature and data type. The detailed parameters are described below, while the best parameters of each model are displayed in Table A.1.

- penalty: L1, L2
- c: numpy.logspace(-3,3,5)
- solver: liblinear, saga
- max_iter: 50,100,150

Table A.1: Best parameters obtained with grid search and logistic regression

Data	Feature	Penalty	Solver	C	Max Iter
Original	$W2V_{gen}$	l2	liblinear	31.6228	50
	$W2V_{spec}$	l1	saga	1000.0	100
	$BERT_{gen}$	l2	saga	1.0	150
	$BERT_{spec}$	l1	saga	1000.0	100
Augmented	$W2V_{gen}$	l2	liblinear	1000.0	50
	$W2V_{spec}$	l2	liblinear	1000.0	50
	$BERT_{gen}$	l2	liblinear	1000.0	50
	$BERT_{spec}$	l2	liblinear	1000.0	50

Source: Elaborated by the author (2022).

A.2 SVM

During the grid search investigation of SVM, we tested three parameters in the eight configurations, with 225 models tested in each. The resulted parameters considered optimal during the investigation are described in Table A.2, with the following hyperparameter values.

- c:1, 10, 100, 1000, 10000

[1] https://scikit-learn.org/

- kernel: linear, rbf, sigmoid
- gamma: 0.0001, 0.01, 1

Table A.2: Best parameters obtained with grid search and SVM

Data	Feature	Kernel	Gamma	C
Original	$W2V_{gen}$	rbf	0.0001	10000
	$W2V_{spec}$	rbf	0.0001	10000
	$BERT_{gen}$	rbf	0.01	10
	$BERT_{spec}$	rbf	0.01	100
Augmented	$W2V_{gen}$	rbf	1	100
	$W2V_{spec}$	rbf	1	100
	$BERT_{gen}$	rbf	0.01	1000
	$BERT_{spec}$	rbf	0.01	1000

Source: Elaborated by the author (2022).

A.3 RANDOM FOREST

Random forest algorithm was optimized with five hyperparameters and a total of 810 fits for each representation. The studied hyperparameters are detailed below, with the best configurations presented in Table A.3.

- n_estimators: 100, 300, 500
- max_features: auto, log2, sqrt
- max_depth: 20, 50, None
- min_samples_split: 2, 5, 10
- min_samples_leaf: 1, 2, 4

Table A.3: Best parameters obtained with grid search and random forest

Data	Feature	No. Estimators	Max Depth	Max Features	Min Sample Leaf	Min Sample Split
Original	$W2V_{gen}$	500	50	auto	2	10
	$W2V_{spec}$	500	20	auto	2	5
	$BERT_{gen}$	500	20	auto	4	10
	$BERT_{spec}$	500	20	auto	1	2
Augmented	$W2V_{gen}$	500	50	log2	1	2
	$W2V_{spec}$	500	50	log2	1	2
	$BERT_{gen}$	500	None	log2	1	2
	$BERT_{spec}$	500	50	log2	1	2

Source: Elaborated by the author (2022).

A.4 GRADIENT BOOSTING

The last classifier optimized with grid search was the gradient boosting, with 6 parameters investigated and a total of 720 fits for each set of feature and data type. This was the most time-expensive model to investigate given its sequential characteristic that decreases the efficiency of computational parallelism. Next, we describe the tested hyperparameters, followed by Table A.4 with the best combinations.

- n_estimators: 100, 300
- max_features: log2, sqrt
- max_depth: 3, 5, 8
- min_samples_split: 2, 5
- min_samples_leaf: 1, 2
- learning_rate: 0.05, 0.1, 0.2

Table A.4: Best parameters obtained with grid search and gradient boosting

Data	**Feature**	**No. Estim.**	**Learn. Rate**	**Max Depth**	**Max Feat.**	**Min Sample Leaf**	**Min Sample Split**
Original	$W2V_{gen}$	300	0.05	3	log2	2	2
	$W2V_{spec}$	300	0.05	3	log2	2	5
	$BERT_{gen}$	300	0.05	3	log2	1	5
	$BERT_{spec}$	300	0.05	3	log2	2	5
Augmented	$W2V_{gen}$	300	0.05	5	log2	2	2
	$W2V_{spec}$	300	0.05	5	log2	2	5
	$BERT_{gen}$	300	0.05	8	log2	2	5
	$BERT_{spec}$	300	0.05	8	log2	2	5

Source: Elaborated by the author (2022).