João Filipe Ferreira Almeida

CATEGORY INFERENCE FOR ELECTRONIC PROGRAM GUIDE ENRICHMENT

UNIVERSIDADE D COIMBRA 1 2 9 0

1 2  9 0

UNIVERSIDADE D
COIMBRA

João Filipe Ferreira Almeida

CATEGORY INFERENCE FOR ELECTRONIC PROGRAM GUIDE ENRICHMENT

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems, advised by Professor Hugo Oliveira (DEI) and Luís Cortesão (Altice Labs) and presented to Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2021

Faculty of Sciences and Technology

Department of Informatics Engineering

Category inference for Electronic Program Guide enrichment

João Filipe Ferreira Almeida

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof. Hugo Oliveira (DEI) and Luís Cortesão (Altice Labs) and presented to the
Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2021

UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

Abstract

Currently, the Recommendation Systems have shown an increase in their use, since they allow better delivery of a service to its users. However, for these systems to be robust and efficient, a correctly classified content base is required. In the case of television program classification, its classification can, besides improving the content shown in the Electronic Program Guide, improve the database used for the Recommendation Systems and consequently the system that makes use of this data. To automate the process, Natural Language Processing appears as a very useful set of techniques, since it allows machines to understand and process natural language, as well as classify large amounts of data automatically.

This document, elaborated in the context of a Dissertation to obtain a Master's degree and internship at the company Altice Labs in articulation with Altice/MEO, presents the process of developing a model capable of classifying television programs by their category, based on the text of their synopsis, using Natural Language Processing techniques. We start by discussing the theoretical context of some important concepts for the problem, followed by a review of similar works to understand the techniques typically implemented and associated results. Next, we present, in detail, the tasks developed and the decision path in the context of this research/internship, as well as how the final product is being integrated into the MEO systems to be used directly on their database.

The development of an intelligent model for program classification allows Altice/MEO to automate its classification processes and results in a consequent improvement of both the content presented in the Electronic Program Guide (EPG) and the Recommendation Systems that make use of that classification, as well as helping indicate business opportunities by improving the detail on the client's profile.

Keywords

Natural Language Processing; Text classification; Data mining; Electronic Program Guide

This page is intentionally left blank.

Resumo

Atualmente, os Sistemas de Recomendação têm apresentado um aumento na sua utilização, uma vez que permitem uma melhor entrega de um serviço aos seus utilizadores. No entanto, de forma a que estes sistemas consigam ser robustos e eficientes, é necessária a existência de uma base de conteúdo corretamente classificado. No caso de classificação de programas de televisão, a sua classificação pode, para além de melhorar o conteúdo mostrado no Guia de Programas Eletrónicos, melhorar a base de dados utilizada para os Sistemas de Recomendação e consequentemente o sistema que faz uso desses dados. De forma a automatizar o processo, o Processamento de Linguagem Natural surge como um conjunto de técnicas muito útil, uma vez que permite às máquinas entender e processar linguagem natural, assim como classificar grandes quantidades de dados automaticamente.

Este documento, elaborado em contexto de Dissertação para obtenção de Mestrado e estágio na empresa Altice Labs em articulação com a Altice/MEO, apresenta o processo de desenvolvimento de um modelo capaz de classificar programas de televisão pela sua categoria, com base no texto da sua sinopse, utilizando técnicas de Processamento de Linguagem Natural. Começamos por discutir o contexto teórico de alguns conceitos importantes para o problema, seguindo-se uma revisão a trabalhos semelhantes de forma a perceber as técnicas tipicamente implementadas e resultados obtidos. Seguidamente, são apresentadas, com detalhe, as tarefas desenvolvidas e decisões tomadas no contexto desta investigação/estágio, às quais acresce uma explicação de como o produto final está a ser integrado nos sistemas da MEO para ser utilizado diretamente nas suas bases de dados.

O desenvolvimento de um modelo inteligente para classificação de programas permite à Altice/MEO a automatização dos seus processos de classificação e um consequente melhoramento do conteúdo apresentado no Guia de Programas Eletrónicos (EPG) e nos Sistemas de Recomendação que fazem uso dessa classificação, assim como ajudará também a indicar oportunidades de negócio ao melhorar o detalhe no perfil dos clientes.

Palavras-Chave

Processamento de Linguagem Natural; Classificação de texto; Mineração de dados; Guia de Programas Eletrónicos

This page is intentionally left blank.

Acknowledgements

Throughout the writing of this dissertation, I have received a great deal of support and assistance.

First, I would like to thank my advisors, Professor Hugo Oliveira and Engineer Luís Cortesão, whose expertise and goodwill to help and guide me were essential for clearing all the doubts I had and formulating a good methodology. Your insightful feedback helped me to bring my work to a higher level and oriented me toward taking the right steps.

I would also like to thank the elements from the MEO team, Iris Costa and Nuno Pimenta for their wonderful collaboration. I want to thank you for all the suggestions provided during the development of this project and for pushing me to achieve better results gradually.

In addition, I would like to thank my parents and sister for their sympathetic ear. You were always there for me, even when things would get more complicated. Finally, I could not have completed this dissertation without the support of all my friends, of which I would like to mention Jessica Mégane and Soraia Santos. Jessica, you helped me a lot in developing the first iteration of this project and in structuring the dissertation. And Soraia, your efforts were essential to the conclusion of this document, by helping me build most of the tables and schemes presented and also keeping a clear mind and focus on both the project and the writing.

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

BERT Bidirectional Encoder Representations from Transformers. ix, xiv, 11, 12, 15, 18–20, 49, 53, 54, 62, 64, 66, 70

BoF Bag-of-Feature. 9

BoW Bag-of-Words. ix, 9, 19–21

CBOW Continuous bag-of-words. 10

CNN Convolutional Neural Networks. 15

EPG Electronic Program Guide. iii, 1, 23

GloVe Global Vectors for Word Representation. ix, 10, 11

GRU Gated Recurrent Unit. 19, 20

IDF Inverse Document Frequency. 9

KNN K-Nearest Neighbors. xvi, 13, 18, 20, 21, 27, 31, 32, 35, 36, 66

LDA Latent Dirichlet Allocation. 45

LSTM Long Short-Term Memory. 14, 19–21

ML Machine Learning. 5, 12–14, 18, 20

MLM Masked Language Model. 11

NLP Natural Language Processing. xiv, 5, 6, 11, 12, 15, 20, 49, 70

NLTK Natural Language Toolkit. 26

NSP Next Sentence Prediction. 12

PMM Parametric Mixture Model. 18, 20

PoS Part-of-Speech. 8

RDF Resource Description Framework. 27, 29

RNN Recurrent Neural Network. 19, 20

SMOTE Synthetic Minority Over-sampling Technique. 12, 21

SVM Support Vector Machine. xiv, 13, 14, 18, 20, 21, 27, 31, 35, 37, 66

TF-IDF Term Frequency - Inverse Document Frequency. ix, 9, 18–21, 27, 29, 31, 43, 48, 50, 51, 53, 54, 59, 66

UC Universidade de Coimbra. 1

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

Recommending systems are increasingly being incorporated everywhere because they allow the customer to have personalized suggestions based on their preferences, consequently improving the user experience of the services where they are incorporated. The visualization of television programs is a way in which a customer's consumer profile can be enriched. To identify similar programs and further predict user preferences, it is important to know the type of each program, i.e., it is important to have the programs in the Electronic Program Guide (EPG) classified according to their genre so that they are displayed accordingly to the taste of each user.

This dissertation, with the theme "Category inference for Electronic Program Guide enrichment", is developed in the context of the 2nd grade of Master degree in Informatics Engineering in Universidade de Coimbra (UC).

In the context of this internship at Altice Labs, working together with MEO, there is a considerable number of programs in the EPG that are not yet classified, and so, the recommender systems and knowledge management components that they may integrate are not as effective and complete as desired. Also, currently, at MEO, the annotation of the program's genres is excessively manual, being the movies genres usually extracted from the IMDB website one by one. However, classifying each program manually is a laborious task and a broad automatic classification is challenging. In addition, for example, the sports programs are not labeled by their sport, and in this type of category it may be important to label them correctly since two different sports usually have distinct audiences (e.g. Football and MotoGP or NBA).

Therefore, given the size of the data involved, there is the need to infer the genre of a program automatically, given the information there is available about it - mainly the synopsis - and apply this approach to the unclassified programs, to complete the existing information. We have access to the history of the EPG and online repositories with information about the programs. In general, the main objectives are:

- Develop an automatic classifier, targeting a prioritary category of programs
- Improve information available about the programs allowing to enhance the effectiveness of the recommendation systems and user profiling for business opportunities
- Prepare it for integration in the MEO's ecosystem, by cleaning and commenting the code and providing a tutorial on how to install and use it
- Provide the team members with insightful metrics obtained during the course of this

investigation

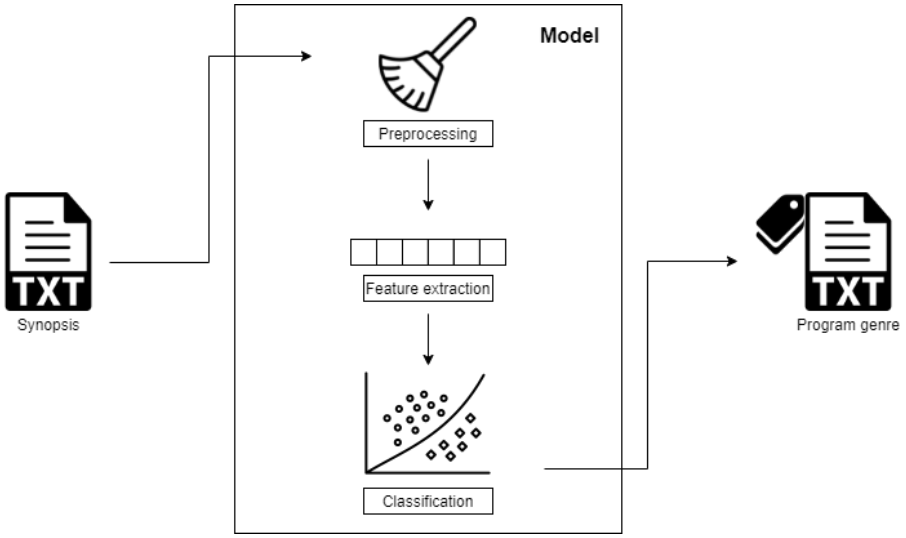

Figure 1.1: Model to be developed
and a broad automatic classification is challenging

To achieve these objectives, we adopted an automatic text classification, using text cleaning techniques to make the text as clear and objective as possible, extract features from synopses of manually classified programs, and try different classifiers, as represented in figure 1.1. This process was performed by studying and testing different parameters on preprocessing, feature extraction, classifiers, and other techniques considered relevant in the scope of this project, such as over-sampling, and having their performance compared.

The course of action for development was divided in two iterations. The first one, which occurred during the first semester, served the purpose of performing preliminary experiments in a movies dataset for familiarizing with the dataset, tools and techniques that were going to be used. In the second iteration, which took place during the second semester, the development of the sports classifier and its integration on MEO's systems take place.

By the end of this project, we managed to develop an automatic text classification model that is already being incorporated directly in the MEO ecosystem for classifying the database entries related to sports programs. Our classifier showed to have a substantial impact in the completion of the data available, allowing for recommending systems improvement and also better profiling for each client, allowing for discovery of potential business opportunities guided by each customer's liking. Along with this classifier, some other models were developed and were delivered for being used as a potential basis on classifiers that are still to be fully developed. Moreover, during the development of this project, several metrics were also provided for helping the MEO team obtaining more insights into their data and also for aiding with the decision-making process. The results obtained show that, in some circumstances, it is better to have a combination of simple techniques, and also, we learned that it was important to have tested different options for each of the model's phases (preprocessing, feature extraction and classification) in separate.

This document starts by presenting, in chapter 2, the background of this project and the theory behind the various components. In chapter 3, some of the related work that was considered relevant for decision making is presented. Chapter 4 describes some preliminary experiments that were made for familiarizing with the needed tools, dataset, and also

obtaining a baseline for the upcoming tests. In chapter 5, there is an exposition of the decision-making process for priority identification, as well as a detailed explanation of all the steps performed for creating a reliable sports programs classifier that can be used quickly on large amounts of data. One of the issues that is also discussed is the need to create a training dataset since there was no classified data available at the beginning. Chapter 6 describes how the final product is being integrated into the MEO ecosystem to work together with the program's database. We will also discuss the different versions of the product, the main changes between each, and how the whole process works. Finally, chapter 7 presents a brief overview of the project, where we highlight the main contributions, the impact of the product, and what could still be done.

This page is intentionally left blank.

Chapter 2

Background

Text classification is an ongoing problem, and so, in recent years, there has been much scientific literature about methods that can improve the performance of a text classification process. In this project, we have a problem where text is received as input and needs to be classified according to its genre. Therefore, in this chapter, we will overview different existing algorithms, techniques, and evaluation methods in text feature extraction, Natural Language Processing (NLP) and text classification.

Automatic text classification (Dalal and Zaveri, 2011) is a process that involves assigning a text document to a set of pre-defined classes automatically, by using Machine Learning (ML) techniques.

Usually, there are three major steps when classifying text, which will be defined in more detail in the following sections:

1. Preprocessing the text
2. Extracting the features
3. Classifying the text

In general, a text classification algorithm can be applied to four different levels of scope:

- Document level: Considers a full document
- Paragraph level: Considers a single paragraph
- Sentence level: Considers a single sentence
- Sub-sentence level: Considers expressions within a sentence

The process of classifying text is mostly applied in information retrieval systems (Dwivedi and Arya, 2016), which obtain information that is usually documented on an unstructured nature. However, there are some other application domains such as information filtering (Du et al., 2003), sentiment analysis (Ferilli et al., 2015), recommender systems (Dan and Severin, 2005), knowledge management (Ur-Rahman and Harding, 2012a), document summarization (Ur-Rahman and Harding, 2012b), among others.

NLP (Iosifova et al., 2020) is a set of techniques applied in computer comprehension, analysis, manipulation and generation of natural language. Some of its applications are

machine translation, speech recognition, optical character recognition, part of speech tagging, information retrieval, and so on. It aims at allowing machines to understand natural language, being then able to perform tasks that were until then only at the reach of human beings. For this purpose, Online Knowledge Databases, which are online repositories of linked data, can also be used to help with processing natural language (this will be addressed in Chapter 4).

The term data mining (Hand, 1998) is used to describe the process of trawling through data in the hope of identifying patterns. The main objective of data mining is not allowing to model the fleeting random patterns that may be present in a dataset at a given moment, but to model the underlying structures which give rise to consistent and replicable patterns. Nowadays, it is common for a dataset to have millions of entries, and performing data mining is very useful for detecting frauds or intrusions (Rambola et al., 2018), for example. It is useful for various areas such as bioinformatics, criminal investigation, and text classification (Kamruzzaman et al., 2010).

According to Guarino et al. (2009), ontology is "a special kind of information object or computational artifact". Nicola Guarino refers to it as "An engineering artifact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words". Creating an ontology formally defines a common set of terms that are used to describe and represent a domain, like a more complex and formalized version of vocabulary. Its main goals are to encode a domain with semantics that machines can understand, share knowledge, reuse knowledge, make domain assumptions explicit, analyze domain knowledge and enable large-scale machine processing.

Figure 2.1 shows the main relations between these five concepts, in a very simplistic way.

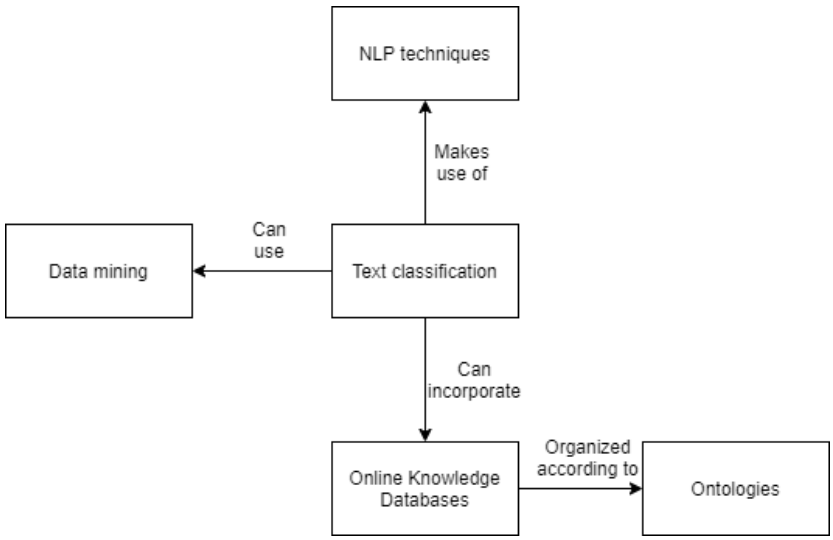

Figure 2.1: Relations between Text Classification, NLP, Online Knowledge Databases and Ontologies

2.1 Text preprocessing

Before feeding the data we want to tag to a classifier, it is crucial to first clean the dataset, removing implicit noise and allowing for a more informative featurization, which will be

addressed later.

Tokenization

This first method works by breaking a stream of text into meaningful elements called tokens, which can be words or phrases for example (Gupta and Malhotra, 2015). For text classification, it is usually required for a parser to process the tokenization of the documents, for example:

This sentence will be tokenized so that it can be classified later.

Results in the following tokens (example 1):

{"This" "sentence" "will" "be" "tokenized" "so" "that" "it" "can" "be" "classified" "later"}

Capitalization

To reduce the feature space, it is common to deal with inconsistent capitalization by reducing every letter on a text to lower case. However, this method can cause interpretation problems in some words like "US" (United States) to "us" (pronoun).

For example, the sentence that resulted from the previous section would be then like this (example 2):

{"this" "sentence" "will" "be" "tokenized" "so" "that" "it" "can" "be" "classified" "later"}

Noise Removal

Although it is sometimes performed during text tokenization, noise removal consists in removing unnecessary characters such as punctuation and special characters that will not be important for text classification algorithms.

Spelling Correction

Spelling correction is not so commonly used given the fact that the documents that may be subject to classification have usually been reviewed for typos. However, when classifying comments or social media content, for example, it can be an important step. There are some effective methods of performing spelling correction, like using Trie and Damerau-Levenshtein distance bigram (Christanti et al., 2018), where Damerau-Levenshtein distance is used to determine the edit distance between two words by the operations made with associated costs (deletion, insertion, substitution, and transposition). Trie is the data structure used to represent the known words, which groups words based on their prefix, improving the processing speed.

Stop-word removal

When classifying text, many words appear very often in all kinds of text and, therefore, do not contain important significance to be used by the classification algorithms, such as

{"a", "and", "after", "before", ...}. These are called Stop-Words (Saif et al., 2014), and usually, the best way to deal with them is by removing them from the texts that will be submitted for classification.

In this case, after applying stop-word removal, the tokens from example 2 would become something like (example 3):

{"sentence" "tokenized" "classified"}

Stemming

When training the classifier, one important factor to have into account is that words obtained from the same root, which often have the same meaning, frequently appear. As those words should be associated as one, which is their root form, to reduce the feature space and provide a better training and cleaner input for the classifier, stemming the text is a good way to do it. It consists of consolidating different word forms into the same feature space by removing affixes.

For the previous example of tokens (example 3), after stemming, we would have (example 4):

{"sentenc" "token" "classifi"}

Lemmatization

Lemmatization works similarly to Stemming, but instead of removing the affixes, it replaces the suffix of a word with a different one to get the basic word form (lemma). It typically considers the Part-of-Speech (PoS) of the words (noun, verb, adjective, adverb), which can be obtained automatically with a PoS tagger.

Example 3, after performing lemmatization, would look like (example 5):

{"sentence" "tokenize" "classify"}

2.2 Feature extraction

Before feeding each preprocessed text to a classifier, its content, to take advantage of various Machine Learning algorithms, should be converted into values that are interpretable by a classifier, usually numerical, called features, that represent the text.

2.2.1 N-Gram

When considering feature extraction algorithms, they can be applied to a set of words at a time instead of individual words, to keep track of context and groups of words that occur frequently together, like New York or The Three Musketeers, extracting more information. These groups of words are called N-Grams (Cavnar and Trenkle, 2001), being N the number of words considered.

An example of using 2-Gram:

This sentence will be tokenized so that it can be classified later.

The previous sentence has the following 2-grams:

{"This sentence" "sentence will" "will be" "be tokenized" "tokenized so" "so that" "that it" "it can" "can be" "be classified" "classified later"}

2.2.2 Bag-of-Words (BoW)

The BoW model is one of the simplest representations of text documents. This model is widely used in text classification given its simplicity and interpretability. It represents a body of text, like for example a document, as if it was a bag of words, like a list of unique words that are present in the document while ignoring the order in which they appear.

Joining the example in 2.1 with the tokens from the sentence *"This is another sentence"*, the BoW would look like:

{"This" "sentence" "will" "be" "tokenized" "so" "that" "it" "can" "classified" "later" "another"}

Notice how, for example, the second "be" is not present in this bag of words, since only distinct words are considered.

The Bag-of-Feature (BoF), or BoW as a feature, for the first sentence, would then look like this:

{2,2,1,2,1,1,1,1,1,1,1,0}

and for the second sentence:

{1,1,0,0,0,0,0,0,0,0,0,1}

which are vectors with the count of the number of appearances of each word in the original sentence.

When there are various texts we want to classify, every word in all the texts is considered as an entry for the vector. Nonetheless, if they are not present in the text that is being classified at a given time, their value will be zero.

Even though this method is very simple and easy to implement, there is the limitation of scalability, because the vocabulary of all texts considered can potentially grow into large numbers and every word must be considered for every text.

2.2.3 Term Frequency - Inverse Document Frequency (TF-IDF)

Sparck Jones (1972), proposed Inverse Document Frequency (IDF) as an extension to the BoW model.

The new approach would assign a higher weight to words with more relevance in the portion of text to be considered, given its frequency in the other portions of text. This combination of both term frequency and inverse document frequency is now known as TF-IDF.

The weight of an individual term in a document is given by:

$$W(d,t) = TF(d,t) * \log\left(\frac{N}{df(t)}\right) \tag{2.1}$$

In this equation, N represents the total number of documents and *df(t)* is the number of documents containing the term t in its corpus. Also, the first term in the equation improves the recall while the second term improves the precision of the word embedding (Tokunaga and Iwayama, 1994), which corresponds to mapping a set of words or phrases to vectors of numerical values. Both these metrics (precision and recall) will be addressed later, in section 2.5.

2.2.4 Word2Vec

A simple single-layer architecture based on the inner product between two word vectors has been proposed by Mikolov et al. (2013), using both Continuous bag-of-words (CBOW) and skip-gram models.

Word2Vec was presented as an improved architecture for word embedding. It uses shallow neural networks with one hidden layer, Skip-gram model, and CBOW to create a vector for each word.

The goal of this model is to maximize the value of:

$$\arg\max_{\theta} \prod_{w \in T} \left[\prod_{c \in c(w)} p(c \mid w; \theta) \right] \tag{2.2}$$

where w represents the words, T refers for text, c stands for context and θ is a parameter for the conditional probability.

This is a very powerful method for taking into account the context of a word and can also be used as a dimensionality reduction algorithm since the size of the vector representing each word can be fixed. However, since in this project we will want to classify a group of words (that can be a sentence or more, depending on the synopsis, but will be a group of tokens after preprocessing), this method will not be so powerful, since other methods already deal with those conditions directly.

2.2.5 Global Vectors for Word Representation (GloVe)

GloVe (Pennington et al., 2014) is another powerful technique utilized for word embedding. It combines the advantages of both global matrix factorization and local context window methods. Like in Word2Vec (2.2.4), every word is represented by a high dimension vector, and the training is also made based on surrounding words. It is efficient by training only on the nonzero elements in a word-word co-occurrence matrix, rather than on the entire sparse matrix or individual context windows and produces a vector space with meaningful substructure to be used in word analogy tasks, just like Word2Vec, like using the cosine similarity (between the vectors) to determine the analogy level of two words. In this case, the objective function is:

$$f\left(w_i - w_j, \widetilde{w}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{2.3}$$

where w_i refers to the word vector of word i and P_{ik} to the probability of occurrence of word k in the context of word i.

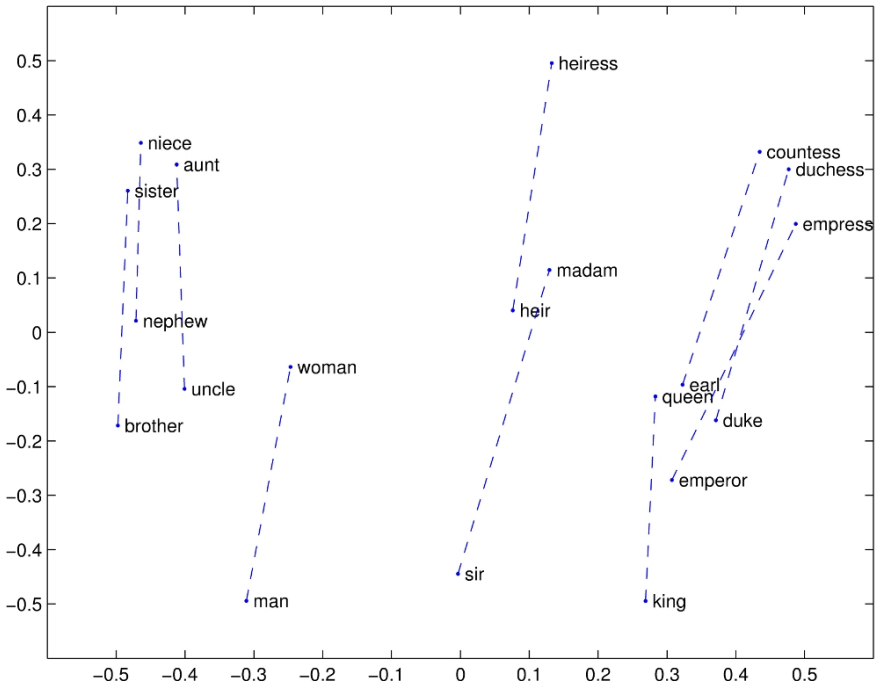

Figure 2.2: Vector representation obtained with GloVe

In figure 2.2 we can observe some examples of vector representations of words by using GloVe. We can see that, for example, the vectors that connect the masculine words to their feminine equivalent all go upwards and have approximate directions. Even so, like in 2.2.4, this method is best at the word level.

2.2.6 Bidirectional Encoder Representations from Transformers (BERT)

When it comes to State of the Art performance in NLP, in general, Google AI Language's BERT, introduced by Devlin et al. (2019), is a model that has to be considered given its results in a wide variety of tasks, including Question Answering and Natural Language Inference. Its innovation comes by applying the bidirectional training of a Transformer (whose meaning will be defined later in this section). By being trained bidirectionally, the model can have a deeper sense of context and flow. BERT's architecture can be visualized in figure 2.3. We can see the encoder stack (in yellow) that has 12 layers in the base model and 24 in the large model. In the BERT architecture there are also large feed-forward networks (768 and 1024 hidden units, concerning the base and large models).

In NLP, a Transformer (Vaswani et al., 2017) is an attention mechanism that learns contextual relations between words (or sub-words) in a text. Its base form includes two mechanisms: an encoder that reads the text input and a decoder that produces a prediction for the task. In BERT's case, only the encoder mechanism is necessary since the goal is to generate a language model.

Transformers make use of attention functions, that are mappings of a query and a set of key-value pairs to an output, where all elements are vectors. Some of its main differences in comparison to Word2Vec are the use of contextualized representations, representations for sequences of tokens and not only individual tokens, and also being possible to use a pre-trained model that can be fine-tuned for a variety of tasks.

Masked Language Model (MLM) is one of the two pre-training strategies used by BERT.

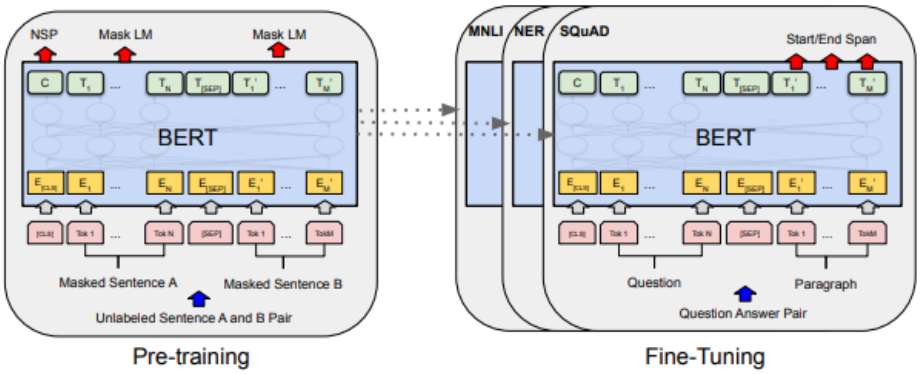

Figure 2.3: Architecture of BERT Devlin et al. (2019)

This strategy consists of replacing 15% of the words in each sequence fed into the model by a [MASK] token. Afterward, the model is trained to predict the original value of the [MASK] tokens only by the context provided by the remaining words.

The second training strategy used by the BERT model is Next Sentence Prediction (NSP). It consists of BERT receiving pairs of sentences as input and learning to predict if the second sentence in the pair is the subsequent sentence in the original document. Both these tasks contribute to developing a contextual semantic representation and coherence between sentences, during pre-training.

The BERT model is implemented in a way that allows its incorporation for various NLP tasks, by only adding a small layer to the core model. For example, in applications involving text pairs, usually, every pair is independently encoded before applying bidirectional cross attention (Parikh et al., 2016, Seo et al., 2018). BERT uses the self-attention mechanism to unify these two stages. The fine-tuning can be performed for tasks like Sentence Classification, Natural Language Inference, Semantic Textual Similarity, Question Answering, among others.

2.3 Over-sampling

In ML context, it is usual for the dataset to not have the same number of samples for every label, which may influence the classifier to choose certain labels to the detriment of others, since they were not so present in the training phase. To address this, it is common to implement an over-sampling technique on the labels with fewer samples, to even out the training dataset.

When thinking about the idea behind over-sampling, the most simple and intuitive method that comes to mind is RandomSampling. This method consists in simply selecting random samples from the labels with less occurrence with equal probability and repeat them in the dataset until those labels have as many samples as wanted.

Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002) also generates new samples for the labels that need them, but differs from the RandomSampling in the way it generates them. With SMOTE, the minority class is over-sampled by creating "synthetic" examples, operating in feature space rather than over-sampling with replacement. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining the k minority class nearest neighbors.

2.4 Automatic Classification algorithms

After processing the text to be classified, it is necessary to perform its classification. For this we have several techniques that can be used and, in this section, we will briefly discuss some of them.

Logistic Regression

Logistic Regression (Peng et al., 2002), one of the first methods introduced for classification, is a linear classifier that predicts probabilities on binary data but can be applied to multiclass datasets using a One vs All scenario, for example. Its goal is to simply train for the probability of its output variables being 0 or 1 given the input variables. Even though it works well for predicting categorical outcomes, independence of data is required, since it is assumed by the probabilities used by the model.

Naïve Bayes Classifier

The Bayes theorem (Joyce, 2019) based model (Frank and Bouckaert, 2006), is a simple technique to assign class labels to problem instances. This method assumes independence between features and uses conditional probabilities to select the most likely estimate as to the output. Despite usually obtaining good results (Zhang and Li, 2007), this type of classifier makes strong independence assumptions between the features, assuming all features are conditionally independent (Rennie et al., 2003).

K-Nearest Neighbors (KNN)

Widely used for text classification (Bučar and Povh, 2013, Guo et al., 2006, Liu et al., 2019, Shang et al., 2006), the KNN Classifier (Li et al., 2003) tags an entry in the input set by assigning scores to the various classes, taking as a reference point the k already classified neighbors. The value of k is a variable that needs to be optimized based on context, as it determines the number of nearest neighbors to consider when calculating the scores. Although this classifier has many advantages like being easy to implement, adapting to any kind of feature space, and handling multi-class cases, some drawbacks are data storage constraints and having a performance that is very dependent on finding the best distance measuring function.

Gradient boosting

Gradient boosting is a ML technique that produces a prediction model using an ensemble of weak models, usually decision trees. The idea of gradient boosting was introduced by Breiman (1997), being the explicit algorithms developed by Friedman (2002, 2000). These algorithms optimize a cost function over a function space, iteratively choosing a function that points in the opposite direction of the gradient.

Support Vector Machine (SVM)

Initially developed by Vapnik and Chervonenkis and adapted into nonlinear formulation by Boser et al. (1996), the Support Vector Machine aims at finding a hyperplane in an

N-dimensional space, being N the number of features of the data points. Support Vectors are points in the data that are closer to the hyperplane, which influence its position and orientation. These points are used to maximize the margin of the classifier to minimize the structural risk. This classifier can be extended to non-linear problems with the use of a kernel function that maps the non-linear data points into a higher dimension so that the classification process becomes easier, with the help of decision surfaces. In figure 2.4 we have a visual representation of how the optimal hyperplane is defined in a SVM. The colored points in the figure represent the Support Vectors and, by maximizing the margin, some new points that appear closer to the hyperplane than the Support Vectors can still be correctly classified.

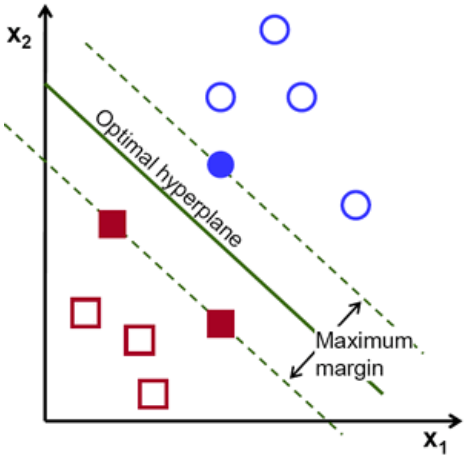

Figure 2.4: Choosing the optimal hyperplane in a Support Vector Machine

While it is one of the most efficient algorithms in ML (Karamizadeh et al., 2014), SVMs have the problem of lack of transparency, caused by the use of a high number of dimensions to make the classification decisions.

Long Short-Term Memory (LSTM) networks

LSTM networks (Hochreiter and Schmidhuber, 1997) are a special type of Recurrent Neural Network, which are Neural Networks that can use their internal state to process sequences of inputs. This classification algorithm makes it easier to remember past data in memory, resolving the vanishing gradient problem of typical RNNs, which occurs because the gradient that carries information used in the parameter update (backpropagation) becomes smaller and smaller until the parameter updates become insignificant. In this type of network, three gates are present in each block: input gate, output gate, and forget gate. The input gate discovers which value from input should be used to modify the memory with a *sigmoid* function deciding which values to let through and *tanh* function to decide the level of importance of those values, as we can see in 2.4. The output gate uses the input and memory of the block to determine the output with *sigmoid* and *tanh* functions working similarly, as is present in 2.5. Finally, the forget gate discovers what details should be discarded from the block by using a *sigmoid* function, with consideration to the input and previous state, with the equation 2.6.

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \end{aligned} \tag{2.4}$$

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{2.5}$$

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right) \tag{2.6}$$

To overcome the bias when later words are more influential than earlier ones, Convolutional Neural Networks are also an interesting option.

Backpropagation is an algorithm for training feedforward neural networks, computing the gradient of the loss function concerning the weights, updating the weights to minimize loss.

Convolutional Neural Networks (CNN)

Although originally built for computer vision, CNNs (Lai et al., 2015) can also be used for text classification (Amin and Nadeem, 2019). The convolution process is just like a sliding window (filter) function applied to a matrix, resulting in new perceptions over the data. CNNs are formed by several layers of convolutions with nonlinear activation functions. Also, a CNN automatically learns the value of its filters depending on the task it is supposed to perform. It can be applied to NLP because in this case, we have an input matrix in which each row is a vector representing a word. Therefore, the filters slide over full rows of the matrix (words) and typically hover 2-5 words at a time. Pooling layers will then sub-sample their input. Pooling is useful because it provides a fixed size output matrix and reduces the output dimensionality while keeping the most salient information (by using a max pool).

Because each row of the input matrix corresponds to an individual word, this method is used when the feature extraction algorithm works for one word at a time.

Transformers

Already described in detail in Section 2.2.6, Transformer based architectures like BERT can also be fine-tuned to perform a variety of tasks, among which is text classification. In Chi et al. (2019), a general solution for fine-tuning BERT in a text classification task is described, after performing exhaustive experiments to investigate different methods. Some experimental findings are that the top layer of BERT is more useful for text classification, it is possible to overcome the catastrophic forgetting problem by using an appropriate layer-wise decreasing learning rate and within-task and in-domain further pre-training can significantly boost BERT's performance, for example.

2.5 Evaluation metrics

Whilst it is important to classify the text correctly, it is just as important to have the correct metrics to keep track of the performance and measure progress efficiently and clearly. For that, in this section, we will discuss some of the most popular metrics for measuring the performance of a classifier. In figure 2.5 we can see the interpretation of a confusion matrix. TP stands for true positives, TN for true negatives, FP for false

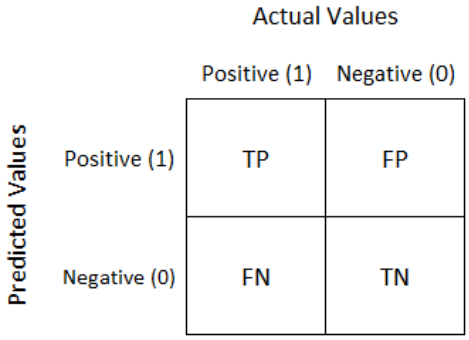

Figure 2.5: Confusion matrix interpretation (Narkhede, 2018)

positives, and FN for false negatives, being positives categories attributed by the classifier and negatives the remainder.

This accuracy metric indicates the fraction of correct predictions made by the classifier.

$$\text{accuracy} = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{2.7}$$

The sensitivity indicates the fraction of known positives that are correctly predicted.

$$\text{sensitivity} = \frac{TP}{(TP + FN)} \tag{2.8}$$

Specificity stands for the ratio of correctly predicted negatives.

$$\text{specificity} = \frac{TN}{(TN + FP)} \tag{2.9}$$

Both sensitivity and specificity can be useful when obtaining information about the performance obtained for each of the classes.

Precision and Recall work just like sensitivity and specificity, but in this case, they are applied to multiclass problems, to represent the performance for all the classes.

$$\text{precision} = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l=1}^{L} TP_l + FP_l} \tag{2.10}$$

$$\text{recall} = \frac{\sum_{l=1}^{L} TP_l}{\sum_{l=1}^{L} TP_l + FN_l} \tag{2.11}$$

F1-Score represents the harmonic mean of both precision and recall, being usually the used metric for measuring a classifier's performance, since it penalizes the extreme values, although it can be complemented by others. Also, F1-Score is better when the classes are unbalanced, which is often the case in real-life classification problems.

$$F1 - \text{ Score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.12}$$

This page is intentionally left blank.

Chapter 3

Related work

Text classification problems have been widely studied in many real-world applications with various approaches that are worth being reviewed. In this chapter, some interesting approaches to text classification in real scenarios will be introduced, as well as their datasets, algorithms, and metrics.

As we are trying to understand the best approaches in text classification, we will try to narrow down this research to be as close as possible to the theme. The following articles are close to what will be developed and were analyzed taking into account their year of publication, the dataset used, number of classes, methods used to extract features, classifiers used, and results obtained. The last reviewed article compares Bidirectional Encoder Representations from Transformers (BERT)'s performance to some other traditional Machine Learning (ML) algorithms used for classification.

Ho (2011) attempted to classify movies by genre, based on the synopsis, in a multi-class problem context, where one movie can be associated with more than one genre. The data used was from IMDB[1], with 16,000 different movies' information, and only the most common 10 genres of the 22 in IMDB were used for this problem, with 80% of the entries being for training and 20% for testing. The genres considered were action (13.2%), adventure (9.1%), comedy (30.0%), crime (13.1%), documentary (16.3%), drama (49.1%), family (11.0%), romance (15.65%), short films (33.4%) and thriller (13.6%). Different classification methods were used, such as Support Vector Machine (SVM) with One-vs-All scenario, Multi-label K-Nearest Neighbors (KNN), Parametric Mixture Model (PMM)[2] and a Neural Network with backpropagation, which is an algorithm for training feedforward neural networks, computing the gradient of the loss function concerning the weights, updating the weights to minimize loss. All these classifiers used Term Frequency - Inverse Document Frequency (TF-IDF), with 10656 features, to represent the data as a vector, which was normalized into a unit vector, to take into account the variation in length between the synopsis of different movies. The best result was achieved by the SVM, with an F1-Score of 0.55. KNN reached a score of 0.52, the PMM approach got a maximum score of 0.49 and the best Neural Network obtained a score of 0.52.

Hoang (2018), used 255,853 movies' synopses also from IMDB, to test different methods to predict their genres. The genres considered and their percentages of movies were drama (46.0%), comedy (27.9%), thriller (11.2%), romance (11.0%), action (9.7%), family (8.1%), horror (7.7%), crime (7.3%), adventure (7.0%), animation (6.2%), fantasy (5.9%), sci-fi (5.6%), mystery (5.4%), biography (5.2%), music (4.9%), history (4.7%), war (2.8%),

[1] `https://www.imdb.com`

[2] probabilistic model used to represent the presence of sub-populations inside a general population

western (2.4%), sport (2.4%) and musical (2.2%). Tested methods were Naïve Bayes (simple, with Bag-of-Words (BoW) features and with conditional independence assumption), XGBoost classifier (an advanced and efficient implementation of the gradient boosting algorithm) with Word2Vec vectors as features (using pre-trained 300-dimensional embeddings trained on part of the Google News corpus for 3 billion words) and Gated Recurrent Unit (GRU) Neural Network, which is similar to Long Short-Term Memory (LSTM) but combines the forget and input gates into a single update gate. Because the average of embedding vectors of words in a document proved to be a weak representation, Word2Vec + XGBoost performed poorly. On the other hand, with the GRU network as the probabilistic classifier, an F1-Score of 0.56 was obtained with a hit-rate of 80.5%, which is very slightly better than what was achieved by Ho's SVM (Ho, 2011), and so, it shows that combining a probabilistic classifier with a probability threshold regressor works better for the multi-label classification problem. If we take into account that this approach considers 20 genres instead of 10 however, the results do become much better. The XGBoost approach obtained a score of 0.49 and the best Naïve Bayes (multinomial) obtained 0.53 F1-Score.

Ertugrul and Karagoz (2018) attempted to classify movie genres from plot synopses using bidirectional LSTM using individual sentences for training. The tags of individual sentences from the same synopsis are then submitted to the majority voting for the final decision. It was determined that bi-LSTM performs better compared to basic Recurrent Neural Network (RNN)s and Logistic Regression as a baseline. The dataset was obtained using MovieLens[3] datasets for getting movie names and OMDb API[4] to get the corresponding genres and plots. Only Thriller, Horror, Comedy, and Drama genres were considered and there were a total of 6,360 movies and 22,278 sentences, given that the movies were uniformly distributed (1,590 for each genre) but the sentences did not since not all the plots had the same size. For preprocessing, all text was converted to lowercase, punctuation was removed and stop-words were eliminated. TF-IDF is used for extracting features to be used by the Logistic Regression estimator. The dataset was split 70% for training, 15% for validation, and 15% for testing and the best result was an F1-Score of 0.6768 obtained with bi-LSTM with a sentence-level approach. The Logistic Regression obtained a best of 0.6394 for the document-level approach and the RNN obtained 0.6261 for the sentence-level approach.

In most recent work, Mangolin et al. (2020), approached the same problem in a multimodal way, by considering trailer video clips, subtitles, synopses, and movie posters taken from 152,622 movie titles from The Movie Database (TMDb)[5], but we will focus on the synopses data. The dataset was made available as a contribution to the work. In this case, there were a total of 18 classes for classification, and the best result was obtained when using both LSTM on the synopses and CNN on movie trailer frames, with an F1-Score of 0.628. Another interesting outcome of this project is that the results corroborate the existence of complementarity among classifiers based on different sources of information in this field of application. The techniques used for the synopsis part of the classification were N-Grams, which is a technique where more than a word can be considered at a time to be turned into a feature, TF-IDF for extracting features after applying N-Grams (1-gram, 2-gram, 3-gram, and 4-gram) and LSTM for performing the classification. The best method for classifying only based on synopsis data was by simply applying LSTM on the synopsis, with an F1-Score of 0.488.

González-Carvajal and Garrido-Merchán (2020) present a comparison between BERT

[3] `https://grouplens.org/datasets/movielens/`
[4] `http://www.omdbapi.com`
[5] `https://www.themoviedb.org/?language=pt-PT`

and some traditional Machine Learning techniques like Logistic Regression or Multinomial Naïve Bayes. There are four datasets used for separate experiments. The first is taken from IMDB and contains 50,000 movie reviews, divided in positive and negative reviews. The second uses RealOrNot tweets[6] written in English, which contain tweets about real and fake disasters. The third is a Portuguese news dataset[7], which contains articles from the news classified into nine different classes (*"ambiente"*, *"equilibrioesaude"*, *"sobretudo"*, *"educação"*, *"ciência"*, *"tec"*, *"turismo"*, *"empreendedorsocial"* and *"comida"*). The fourth and final dataset corresponds to Chinese hotel reviews, for a sentiment analysis problem[8]. The reviews are classified into two different classes, which are positive and negative. Independently of the dataset used, the BERT model always outperforms the classical Natural Language Processing (NLP) approach mixing the TF-IDF technique with an automatic machine learning methodology.

Even though every dataset is different and, in our project, there will be more classes than in every one of the before mentioned examples, this analysis is important to define the performance that is possible to obtain in this kind of text classification problem. Also, the last article showed that the BERT model is capable of obtaining better results than traditional ML techniques for various datasets, even using different languages. However, it would be interesting to review an approach using the BERT model applied to movie classification because this would give an idea of how to perform the best fine-tuning for this kind of task, but apparently, there has not yet been an approach using both BERT and movie genre classification based on its synopsis.

Also, while performing the research on related work, there were no approaches involving movie datasets in Portuguese, so the project that we will develop will be a novelty in the Portuguese language context and will contain a certain level of innovation, as it will be a unique approach with emphasis on its Portuguese dataset.

Table 3.1 shows some of the reviewed articles that are related to our problem and their main characteristics to have in account for building an initial idea of how to approach this kind of problem.

Author(s)	Year	Dataset	Classes	Feature Extraction	Classifier(s)	Best results
Ho K	2011	IMDB, 16,000 entries, multiclass approach, not balanced	Most common 10 in the 22 available	TF-IDF	SVM (One-vs-All), Multi-label KNN, PMM, Neural Network with backpropagation	SVM with 0.55 F1-score
Hoang Q	2018	IMDB, 255,853 entries, multiclass approach, not balanced	20 classes	BoW, Word2Vec	Naïve-Bayes (BoW features and conditional independence assumption), XGBoost classifier (with Word2Vec features), GRU (Neural network similar to LSTM but forget and input gates are merger into a single update gate)	GRU with 0.56 F1-score
Ertungul and Karagoz	2018	MovieLens and OMDb API, 6,360 movies, 22,278 sentences, balanced	4 classes (thriller, horror, comedy and drama)	TF-IDF	Logistic Regression RNN bidirectional LSTM (all with sentence level and document level approaches)	bi-LSTM with sentence level approach with 0.68 F1-score
Mangolin et al.	2020	TMDb, 152,622 entries, dataset made available,	18 classes	N-Grams TF-IDF	LSTM (here we will focus only on the approach used for the synopses)	LSTM with 0.49 F1-score

Table 3.1: Reviewed approaches summary (those using synopsis for genre classification)

When it comes to unlabeled data in great amounts, it is justified to use classification based on key-words, either for the final classification itself or for getting a labeled dataset that can be used for ML.

[6] `https://www.kaggle.com/c/nlp-getting-started`

[7] `https://www.kaggle.com/c/fasam-nlp-competition-turma-4/data`

[8] `https://github.com/Tony607/Chinese_sentiment_analysis/tree/master/data/ChnSentiCorp_htl_ba_6000`

An and Chen (2005) proposes a robust algorithm to induce concepts from training examples, which is based on enumeration of all possible key-words combinations. In this approach, one key-word combination represents one subspace. By checking every subspace whether all documents of a category are covered or not, it is possible to find the "best" subspace to describe that category. For a description of each category, there can be a very big number of key-word combinations and it is impossible to check each combination for each document, and so, in this experiment, pruning power is introduced to reduce the number of possible combinations to a reasonable number, making the algorithm able to run in personal computers with limited memory space. The rules produced by this algorithm proved to be more accurate and interpretable than other concept learning algorithms, such as ID3.

In Lorensuhewa et al. (2002), a simple key-word based matching technique is used to classify a furniture design style by examining its text description. To limit the scope of the problem, they used descriptions from five furniture design styles, namely Jacobean (23.73%), Classical (21.82%), Early Victorian (12.08%), Queen Anne (20.13%) and Chippendale (22.25%), and each text description was labeled by its style category. A domain-specific dictionary was used to limit matching only to domain-specific key-words and all words not found in the dictionary were discarded. In this context, this technique recognized the design style with approximately 75% accuracy, performing similarly to SVM and decision tree-based classifiers.

When it comes to over-sampling techniques, Glazkova (2020) compared several methods for the problem of multi-class topic classification. In this paper, the authors compared the basic Synthetic Minority Over-sampling Technique (SMOTE) method with its two modifications (Borderline SMOTE and ADASYN) and the random over-sampling technique. For the classifiers, KNN, SVM, feed-forward network, LSTM and bidirectional LSTM are used in combination with the synthetic over-sampling methods. With regard to text representation, BoW, BoW + TF-IDF and Word2Vec were used. For this task, the quality of the KNN and SVM algorithms was more influenced by class imbalance than neural networks.

This page is intentionally left blank.

Chapter 4

Preliminary experiments

In addition to research on the state of the art and related work for figuring out the best approach to solve the problem, some preliminary experiments have been made, in the first iteration of this project, for familiarizing with the tools used and the dataset itself. Also, it is very important to have a baseline model and associated results to serve as a comparison for the more complex models that were developed and tested later.

4.1 Familiarization with the dataset

One of the fundamental steps when classifying data is to know the dataset as well as possible so that the best approach and filtering of the data can be considered right from the start. For that matter, the initial weeks were partly spent studying the genres available for classification, the structure of the database, and some examples of grids sent by the media providers to observe what kind of information is available for classification.

Database structure

To store the already classified programs, as well as the information related to the Electronic Program Guide (EPG), there is a database created and used by MEO, mainly in Portuguese, which is very helpful for extracting training and testing data. However, in the data corresponding to August 2020, which was the chosen data to work with for this first phase of the project, 59% of the programs genres, from a total of 235,567 entries, are not defined. In the excel files exported from the database, there are three major dimensions of analysis that need to be taken into account: Channel, Program, and TV Guide. Inside each dimension, there is information related to the dimension itself and relating the instance of one dimension to another. For example, in an entry from TV Guide, both the channel ID and program ID must be indicated, as this table's main function is to connect the data from the other two.

The dataset used for the preliminary experiments contains 94,656 entries, corresponding to the classified programs of the previously referred data, with 43 different genres as labels. To avoid overfitting, entries with the same synopsis were removed, reducing the length to 17,251 entries.

The tables 4.1, 4.2 and 4.3 are representations of the variables considered for each dimension of analysis. The examples presented are in Portuguese because it is the language

Dimension	Atributes	Examples
Program	Program ID	12527978
	Program name	Investigação criminal: Los Angeles
	Program description	Magazine dedicado aos amantes da modalidade
	Program sub-genre	Drama, Comédia, Aldultos, Futebol, Andebol
	Episode	8, 10, ...
	Cast	Nola Augustson, Patrick McCullough,...
	Producer	David Yates,..
	Director	Malcolm Marmorstein,...
	Year of production	1988
	Duration	2100

Table 4.1: Constituents of Program entry

Dimension	Atributes	Examples
Channel	Call Letter	AMC, AMCHD
	Channel name	AMC (SD), AMC (HD)
	Channel description	Um canal que lhe oferece séries ...
	Channel resolution	HD, SD, 4K
	Channel ID	13881
	Channel genre	Desporto, Documentários, Filmes
	Channel type	FTA, Interactivo, Pay, Premium...
	Language	Português, Inglês
	Exclusive Flag	Sim, Não
	Channel Restart Flag	Sim, Não
	Channel Automatic Recording Flag	Sim, Não
	Aggregate Channel	RTP1, RPT2, SIC, TVI

Table 4.2: Constituents of Channel entry

used in the majority of the dataset. These are the variables that may be taken into account while performing the classification, being the Program description and Program sub-genre in table 4.1 the two most important variables, both for training and classification.

Content Cataloguing

The cataloging of content is performed one program at a time for a pre-defined group of channels and genres. Cataloging with genres and sub-genres is performed in Entertainment, Lifestyle, Movies, Series, Documentaries, Children, Sports and Music types of content for generalist channels and channels of these genres. The information content is cataloged with the genre but not the sub-genre. Depending on the type of content, the same program may have sub-genres that belong to different genres. As an example, the Master Chef program is classified as "Entretenimento/Concurso" and also "Lifestyle/Culinária". Accordingly, we can conclude that we are facing a multiclass and multilabel problem in most of the cases.

In table 4.4, all genres and sub-genres considered are displayed, as well as the cataloging process used for attribution of tags. There are a total of **62 possible tags** that may classify a certain program, and the classification of a program results from a combination of tags.

Dimension	Attributes	Examples
TV Guide	Program ID	12527978
	Program description	Magazine dedicado aos amantes da modalidade
	Channel ID	13881
	TV Guide ID	377727261
	Channel description	Um canal que lhe oferece séries de grande ...
	Prog. Start. Dt.	20180418
	Prog. Start Hr.	23:34:45
	Prog. End. Dt.	20180418
	Prog. End. Hr.	23:54:33
	Position in Grid	Position in channel grid (Ex.: 1, 2, etc)

Table 4.3: Constituents of TV Guide entry

TV Content	Genres	Sub-genres	Cataloguing proccess
Movies and Series	Filmes e séries	Ação, Animação, Aventura, Biografia, Comédia, Crime, Curta, Desporto, Documentário, Drama, Família, Fantástico, Ficção Científica, Guerra, História, Juvenil, Mistério, Musical, Romance, Thriller, Western	Based on the IMDB website. Usually contains three sub-genres maximum. When there are more than three sub-genres, an editorial analysis is performed to reduce the number of sub-genres, eliminating redundancies and mantaining the most relevant ones.
Entretainment and Lifestyle	Entretainment	Concurso, Reality TV, Telenovela, Talk Show, Variedades, Musical, Humor, Cultura, Música	Based on content (reading of synopsis, information sent by the content providers and respective grids).
	Lifestyle	Culinária e Gastronomia Saúde e Bem Estar, Viagens, Casa e Jardim, Compras, Moda, Social, Motor	
Documentaries	Documentários	Biografia, Ciência e Tecnologia Natureza, Desporto, Cultura e Sociedade, História	Based on content (reading of synopsis, information sent by the content providers and respective grids).
Children	Infantil	Bebés, Primeiros Anos, Mais Crescidos, Teen, Filmes	Based on content (reading of synopsis, information sent by the content providers and respective grids).
Sports	Desporto	Futebol, Motorizado, Radical, Combate, Ténis, Basquetebol, Futsal, Golfe, Atletismo	Based on words or key-expressions, univocal to the universe of the modality/sport that appears in the titles of sports channels and sporting events that appear on generalist channels.
Music	Música	Clássica Club and Dance Pop & Rock	By channel, being Mezzo and Stringray Clássica classified with "Clássica", Trace Urban and Clubbing TV as "Club and Dance" and Afro Music, Trace Brazuca, Alma Lusa, VH1, MTV Live, MCM POP, MCM Top, Trace Toca, Stingray iConcerts, Stingray Loud and Stingray Retro as "Pop & Rock"
Information	Informação		There are no sub-genres. The content is catalogued according to the channel it is being displayed on.

Table 4.4: Genres considered and individual classification process

Processing the dataset for later usage

It was decided that, for these first phases of development, the information of all the programs broadcasted in the month of August 2020 would be used as the dataset. As the development gets more complex, so will the dataset, allowing the results to be more accurate and the scenario more realistic.

The dataset consisted of three excel files corresponding to the three dimensions with relevance for the classification process. Thus, it would be practical to group all the information needed in one file to improve the time used for reading the dataset and simplify the Dataframe organization. With the help of a Python script, using NumPy[1] and Pandas[2], the relevant data to be used in these preliminary experiments was grouped in one CSV file, that contains the following constituents:

- Program key (the unique identifier for the program)
- Synopsis of the program
- Name of the program
- Sub-genre of the program
- Channel key (the unique identifier for the channel)
- Channel name
- Channel genre
- Channel description

4.2 Familiarization with the tools

Also in the beginning, even though the Python programming language and Big Data tools like Pandas and NumPy were already familiar, Natural Language Processing and Text Classification were not, so there was a need to not only research about the theory behind all that but also to see how some of those new concepts like stop-word removal, stemming, TF-IDF, etc, were used in practical terms and the Python tools that helped to implement them.

For that matter, based on Prateek Joshi's example (Joshi, 2019), we built a prototype using the same tools and the dataset available in `http://www.cs.cmu.edu/~ark/personas/`. This was useful to identify two major Python libraries that can be used to help classifying text. The first one was Natural Language Toolkit (NLTK)[3], which has a lot of functions that implement the major text preprocessing phases. The second one was Sci-kit Learn[4], one of the main libraries for Machine Learning in Python, which can be used for both extracting features from the text and perform the classification.

[1] `https://numpy.org`
[2] `https://pandas.pydata.org`
[3] `http://www.nltk.org`
[4] `https://scikit-learn.org/stable/`

4.3 An approach using Knowledge Databases

During the Knowledge-Based Systems course, an approach using Knowledge Databases for a more complete preprocessing was explored, to determine if those resources could be an added value if incorporated in the final model. This project also helped on acquiring a better understanding of how Natural Language can be converted into data that can be interpreted by a classifier and some new concepts that we thought would be interesting in the context of the project. The possible incorporation of knowledge databases could enrich the preprocessing step of the classification, which could contribute on a good scale for the project's outcome as having a better preprocessing results in an easier task for the classifier and consequently better results, and so, seemed like an interesting route to explore.

4.3.1 Data & Approach

The dataset used in this project belongs to MEO within the scope of this internship. For confidentiality reasons, only publicly available data were used, corresponding to the synopsis and genre of each considered program. To simplify the classification, the number of classes was also reduced to 4, keeping the four different genres related to movies with more occurrences on the dataset: *action*, *comedy*, *crime*, and *drama*.

We tested the classifier's performance only with text preprocessing, with the help of obtaining synonyms and hyperonyms, and with sentiment analysis separate, comparing the results. For the standard preprocessing, we started by applying a function that cleans the text, by putting everything in lowercase, tokenizing the text, removing punctuation, and replacing every number with a zero. After that, we set the stop-words and remove them from every text, so that we can work with words that are valuable for inferring the genres. Then, we apply a stemmer that receives every word and returns its base form, which will help the classifier associate words that have the same meaning.

For the other approaches to preprocessing, we created a Resource Description Framework (RDF) document by interpreting the meaning of triples, both for semantic relations between pairs of words and sentimental values of inflected forms of each lemma, to attenuate the need of using a lemmatizer. This document was created so that the triple information was organized in a way that could be queried using SPARQL (query language to extract information from RDF documents).

Before feeding the data into a classifier, we needed to represent it in the form of features, and for that, Term Frequency - Inverse Document Frequency (TF-IDF) was used.

At last, we feed the features into our classifier, which is initially a One Vs Rest classifier with a Logistic Regression estimator. It is a very simple classifier but we chose it for the initial tests because the main focus of the project is to see the difference in performance between using sentiment analysis, synonyms, or nothing and we gave preference to the optimization of the preprocessing performed on the texts rather than the classifier itself.

Later on, out of interest to see how a different type of classifier would have an impact on the results, a Support Vector Machine (SVM) classifier in a One Vs Rest scenario and a K-Nearest Neighbors (KNN) classifier were also implemented and tested, but this time only for different values of *threshold* (this will be explained later in this section) as it was the only parameter directly related to the classifier itself.

The metric used for interpreting the results was the F1-score, given we are working with a non-balanced multiclass scenario.

4.3.2 Implementation

In this section, we will talk about the implementation of different components of the project.

Preprocessing

The function *clean_ text* is used to lowercase the text, tokenize it (using the *word_ tokenize* module from the *nltk* library), remove the punctuation, and replacing the numbers by 0 (we did this instead of removing the numbers because in some cases knowing that a number exists can be useful for the classifier so that they all are interpreted the same). In figure 4.1 it is possible to see the frequency of the most frequent words after cleaning the text.

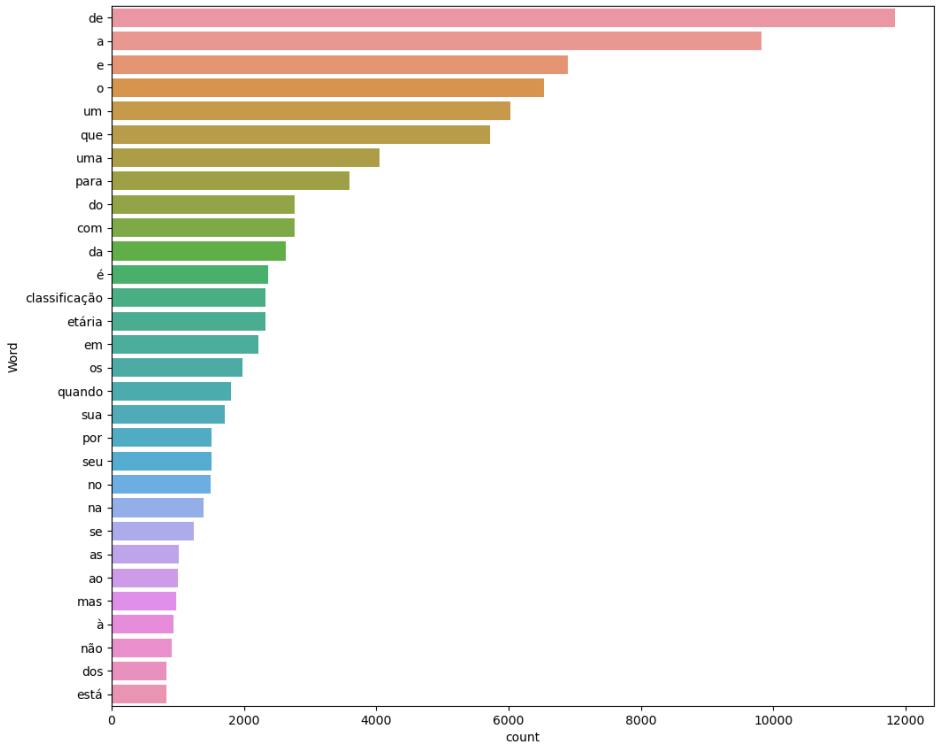

Figure 4.1: Words frequency after cleaning text

To remove the stop-words, we used the *stop-words* module from *nltk* to get a list of English and Portuguese stop-words that were used together with other Portuguese words from a repository on GitHub[5]. We also added some words manually, such as "classificação" and "etária". Figure 4.2, shows the frequency after removing the stop-words. It is interesting to note that the words with more frequency before removing the stop-words (Figure 4.1) are all gone from the scope of this graph after stop-word removal is applied.

To perform the stemming in the data, we used the *stem_ text* function included in the *stem* module from the *nltk* library. This function receives one synopsis at a time in

[5] `https://gist.github.com/alopes/5358189`

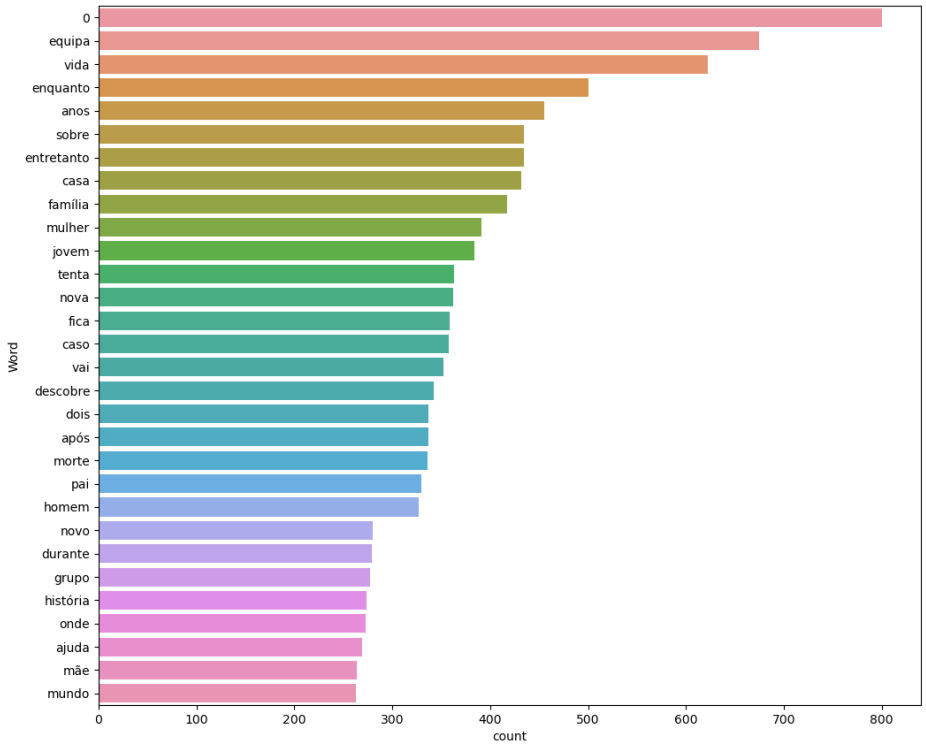

Figure 4.2: Words frequency after removing stop-words

the form of a list of words, obtained from the tokenization performed before, and returns the same synopsis with the words replaced by their root form. We can visualize how this procedure works by looking at Figure 4.3, where only the roots of the words are present, and some words change ranking because of the grouping that occurs in the words with the same root.

TF-IDF

TF-IDF was implemented using the module *TfidfVectorizer* from the *Sci-kit Learn* library. For testing, both the maximum document frequency and max features parameters are varied.

RDF

Within the development of this project, a RDF document was created by interpreting the meaning of triples. This document contains information about a word's synonyms, hyperonyms and other attributes obtained from "Triplos relacionais 10 recursos" from the *Onto.PT* website (Oliveira, 2014), which also has information about the confidence level (a value that represents the number of sources that supported that connection between the words, being the maximum value 10) of each triple. This information would then be extracted through SPARQL queries, which are used to extract specific values from a RDF document. Some words in the document also contain information about their polarity

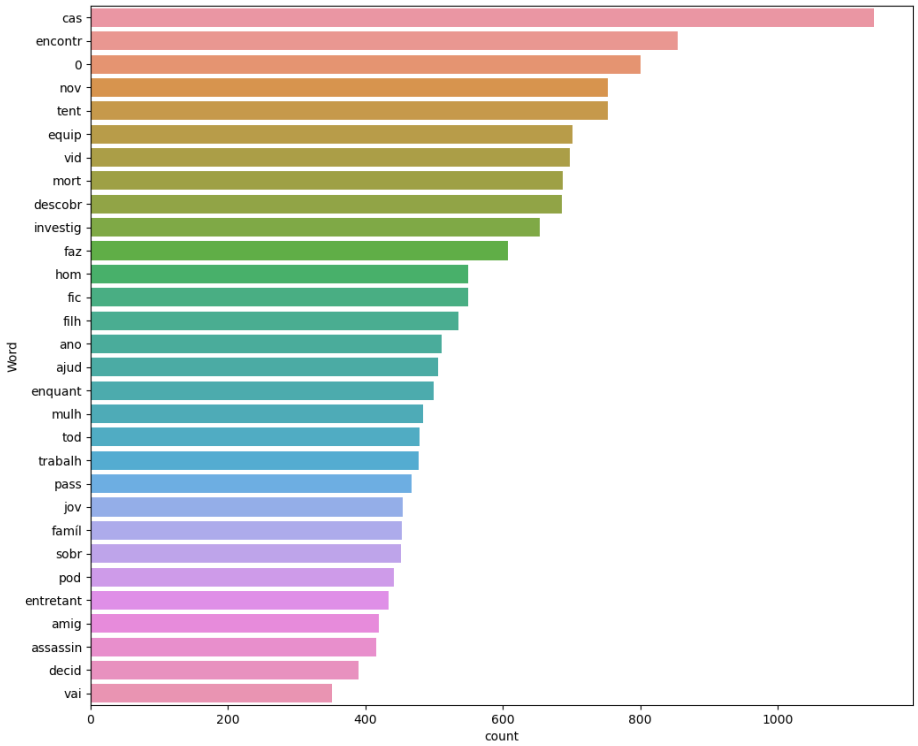

Figure 4.3: Words frequency after stemming

(which is the same as sentimental value), from the *SentiLex-flex-PT* dataset[6] (Silva et al., 2012), which contains various inflected forms of each lemma. This document was then made available in a GitHub repository[7], for other people that may need it.

Synonyms and Hyperonyms

The extraction of synonyms and hyperonyms can be useful for associating words with the same meaning by replacing every synonym and hyponym with the same word, which can be a synonym or hyperonym.

To replace the synonyms and hyponyms, SPARQL queries were performed to select the word with more confidence value. There is a dictionary that stores the correspondences that were made previously, so that a word is always replaced by the same word (which is the main purpose of this approach), so if a word is a value in the dictionary it will be replaced by its key.

There is also a variable called *MIN_COMMON_SYNS* that determines the number of synonyms that a word and a dictionary key's values should have in common before replacing the word by that key and adding it to the corresponding values.

When there are no synonyms available for a word but there are hyperonyms, the word

[6] `https://github.com/sillasgonzaga/lexiconPT/blob/master/data-raw/SentiLex-flex-PT02.txt`
[7] `https://github.com/joao-f-almeida/rdf-with-Onto.PT-and-SentiLex-data/`

is replaced by its hyperonym with the most confidence.

Sentiment Analysis

Sentiment analysis was explored to conclude if the sentimental value of the words in a synopsis could help the classifier categorizing its content.

This technique is used to create new features, with the number of words having positive, negative, and neutral polarity and the sum of those values for each synopsis as a fourth feature. The values are also accessed through SPARQL queries.

Classifiers

The main classifier used was a One vs Rest classifier with the Logistic Regression estimator from the *Sci-kit Learn* package. For the expected genre values we used the *MultilabelBinarizer* module, also from *Sci-kit Learn*, which creates a binary array that represents the genres present for each synopsis.

First, we fit the classifier model with the training dataset and then we obtain the predicted values with probabilities for the testing dataset. Then, using the threshold that is given as a parameter, we will select as predictions the genres that have a probability higher than the threshold value.

Later on, an SVM classifier in a One Vs Rest scenario and a KNN classifier were also implemented with the *Sci-kit Learn* package, to see how they would perform on this same dataset and if any of them would be able to outperform the one with Logistic Regression.

Testing

For testing everything the relevant parameters in a manageable amount of time, the first step was deciding which variables would be worth testing. The chosen variables for the standard preprocessing were the *max_df* and *max_features* parameters of the TF-IDF vectorizer, the use of stop-word removal and stemming or not and finally the *threshold* for the classifier's decision. The *max_df* stands for maximum document frequency and corresponds to the maximum number of documents (in this case synopses) that may contain a word for it to be considered a feature and is expressed between 0 and 1. Its usage allows us to discard words that are present in too many synopses and consequently lose their relevance to the classification process. The *max_features* parameter is related to the number of features that will be returned. For example, if we have more words than the number specified in this variable, some of the less relevant words will be cut from the feature space. The threshold value works as a decision boundary for the classifier. It is a value between 0 and 1 and if, for example, we have 0.4, it means that the classifier will consider as predictions the genres that have a probability of over 0.4 of being correct. We have access to the probability values using the *predict_proba* function that is available in our One Vs All classifier. Also, the stop-word removal and stemming were set to True or False so that we could visualize their impact on the final results.

For the synonym and hyponym substitution, the only variable that we decided to test was *MIN_COMMON_SYNS*.

Then, using preprocessing with consideration for the sentimental value of the words,

only the number of additional features was varied. This number is relative to the count of positive values, negative values, neutral values, and the sum of values per synopsis. We thought it would be worth testing to see if the sum of sentimental values per synopsis would add any value as a feature.

4.3.3 Experimentation

For the experimentation, the dataset was divided into 70% for training and 30% for testing, and the metric used to evaluate the results was the F1-Score.

Procedure

The main point of this study was to determine if the usage of a knowledge database with information related to the synonyms, hyperonyms, and sentimental value of a word would improve the efficiency of the preprocessing, so the first tests aimed at obtaining the results for the standard methods and the best values for the variables referred in Section 4.3.2. We developed a simple script where, for the *max_ df*, every value between 0.1 and 1 was tested with a 0.1 spacing in between, and then, the best value and adjacent best value were tested a second time, as well as all the values between them with a 0.01 spacing. The best value would then be fixed for the *max_ df* and the script would advance for the *max_features* variable where the values between 1000 and 10000 were tested with a 1000 spacing and then with 100 spacing in a similar process to what had been done before to *max_ df*. Then, the threshold was tested with the same procedure and values used for *max_ df*. Finally, the stop-word removal and stemming were varied between True and False and the best value for each variable was saved for the tests that would be performed later on. The mean and standard deviation of the f1-score was obtained in 15 runs of every one of these tests.

For the synonym and hyponym substitution, the values used for *MIN_ COMMON_ SYNS* were 1, 2, 3, and 4 and the values presented were the results of 3 runs made for each value. The reason for the fewer runs made here is because the high amount of SPARQL queries made during each run resulted in time-consuming runs.

Sentimental value analysis results were also obtained from 3 runs because of the same reason and the values used were 3 and 4 features, as was mentioned before.

After that, it was also interesting to visualize the results from both synonym and hyponym substitution and sentimental value analysis working together so we also made 3 runs for this, using the best values from all the previous tests.

To finish the tests, out of curiosity to observe how a different classifier would have an impact on the results, both an SVM classifier and a k-Nearest Neighbors classifier were implemented and tested. For the SVM classifier, tweaking the parameters wasn't appearing to improve the results so the default values were used, only with the addition of considering probabilities so that the threshold could then be tested the same way it was for the Logistic Regression classifier. For the KNN classifier, the default values (considering the 5 nearest neighbors) were also used, since changing the number of neighbors to consider also didn't appear to improve the results significantly.

Results

We started by experimenting different parameters for the *max_ df* and *max_features* parameters of *TF-IDF*, and choose the two consecutive higher values. Then we performed tests with smaller intervals between those values, to select the best parameter to be fixed during the next tests.

max_df	**F1-score**
0.1	0.758 ± 0.005
0.2	0.756 ± 0.007
0.3	0.758 ± 0.004
0.4	**0.762 ± 0.005**
0.5	**0.759 ± 0.006**
0.6	0.757 ± 0.004
0.7	0.761 ± 0.007
0.8	0.755 ± 0.005
0.9	0.757 ± 0.007
1.0	0.447 ± 0.010

Table 4.5: Results obtained for different values of Max_Df

The best results were obtained in the range between 0.4 and 0.5, being 0.5 the value chosen by the algorithm (because it got better results when tested with 0.01 spacing), with a 0.76 F1-Score. We think that this value is close to what should be expected because with *max_ df*=0.5 the words that are in more than half of the synopsis are being cut out and no genre is present in more than half of the total synopsis either, so a word that is available in more than 50% of the synopsis would probably have reduced value as a feature and would introduce confusion to the classifier because it would for sure be related to more than one genre.

max_feat	**F1-score**
1000	0.728 ± 0.004
2000	0.747 ± 0.006
3000	0.757 ± 0.006
4000	0.758 ± 0.006
5000	0.755 ± 0.008
6000	**0.760 ± 0.006**
7000	**0.758 ± 0.005**
8000	0.757 ± 0.005
9000	0.757 ± 0.004
10000	0.757 ± 0.006

Table 4.6: Results obtained for different values of Max Features

The *max_features* parameter corresponds to the maximum amount of words (features) to be considered. The value is relative to the total number of words so it is also dependent on the efficiency of stop-word removal. We performed tests with values in the range of 1000 to 10000, shown in Table 4.6, and realized that with 1000 features, the result was a lot worse than the others, so below that value, the performance should decrease. This means that below 1000, features with significant discriminative values were being discarded. The best result obtained was between 6000 and 7000, and tests with an interval of 100 were performed to show that the best result was using 6200 features. It is important to note

that the total of unique words in all of the datasets is 21,557, so we can consider that only about 29% of the words were considered relevant.

threshold	**F1-score**
0.1	0.560 ± 0.004
0.2	0.670 ± 0.004
0.3	**0.743 ± 0.003**
0.4	**0.756 ± 0.005**
0.5	0.715 ± 0.006
0.6	0.600 ± 0.007
0.7	0.415 ± 0.009
0.8	0.213 ± 0.009
0.9	0.060 ± 0.006
1.0	0.000 ± 0.000

Table 4.7: Results obtained for different values of Threshold

The threshold value is used in the classifier and is represented as a probability to choose the genres as predictions for the classification. Only genres with a certain probability, above the threshold, are assigned. This parameter is the one that affects the results the most because it is necessary to choose a value that can establish the balance between choosing fewer or more genres than what would be necessary. If the threshold is zero, all genres are chosen, and on the other hand, if the threshold is one, none of the genres will be chosen, because the probability never reaches that value.

In table 4.7 we can observe that the best results were obtained in the range of 0.3 and 0.4. After performing tests between that range with an interval of 0.01, 0.38 obtained the best results, so it was the value chosen for the following tests.

stop_words	**F1-score**	**stemming**	**F1-score**
True	**0.762 ± 0.007**	True	0.759 ± 0.006
False	0.762 ± 0.006	**False**	**0.761 ± 0.006**

Table 4.8: Results obtained for the usage of stop-word remmoval and stemming

In Table 4.8, we can see that neither the stop-word removal nor the stemming had a major impact on the results obtained. Part of it is because the function used to clean the text already does a good job but we would expect the difference between the results to be higher, given that these techniques are two of the most important pieces of the text preprocessing pipeline.

Common	**F1-score**
1	0.752 ± 0.004
2	0.757 ± 0.005
3	0.759 ± 0.004
4	**0.760 ± 0.003**

Table 4.9: Synonyms results with different minimum common synonyms

To replace the words with their synonyms and hyperonyms, the minimum number of common synonyms/hyponyms necessary between two words, for them to be retrieved directly from the dictionary was varied. In Table 4.9, we can observe that including the extra step of word substitution in the preprocessing does not have a big impact on the final results. We can deduce that using a lower number, like 1, makes the words

repeat themselves more because the algorithm tries to keep the dictionary reduced by not searching for more words if correspondence is already available and that has pros and cons since even though the words are repeated more frequently, the context can be lost because if correspondence is already available, the algorithm will not search for one that has a higher level of confidence. That explains why the results are better for a higher level of common synonyms/hyponyms required.

Features	**F1-score**
Positives, Neutrals, Negatives	0.753 ± 0.004
Positives, Neutrals, Negatives, Count	**0.754 ± 0.006**

Table 4.10: Results obtained for using Sentimental Value with 3 and 4 features

Using the 4 features showed a little bit of improvement in the results when comparing with using only 3, as we can see in 4.10. In general, though, the incorporation of the sentimental value of words as features did not seem to have a positive impact to make the preprocessing any better.

Test	**F1-score**
Synonyms + Sentiment Analysis	0.750 ± 0.005

Table 4.11: Synonyms and Sentiment Analysis Result

The result with Synonyms and Sentiment Analysis was also a little bit lower than some values that we were able to obtain previously. At this point, it would be expected because neither the synonyms nor the sentiment analysis provided the positive impact it would take for this test to have the potential of being the one with the best results.

Threshold	**F1-score**
0.1	0.690 ± 0.004
0.2	0.748 ± 0.004
0.3	0.771 ± 0.006
0.4	**0.776 ± 0.005**
0.5	**0.773 ± 0.003**
0.6	0.756 ± 0.003
0.7	0.714 ± 0.005
0.8	0.650 ± 0.004
0.9	0.538 ± 0.006
1.0	0.000 ± 0.000

Table 4.12: Results of different threshold values in SVM classifier

The results obtained on both Table 4.12 and Table 4.13 show us that by changing the classifier we can considerably change the results obtained. It was noted that the SVM classifier was able to produce better results than the Logistic Regression with the same parameters besides the threshold, even though it was tested out of curiosity. One interesting fact however is that the KNN classifier doesn't obtain a result of zero f1-score when the threshold is 1. That is because if the 5 neighbors considered all have the same genre, then it will be assumed with 100% probability by the classified that the genre in question is the correct prediction.

The SVM classifier was also tested with Synonym/Hyponym substitution and Sentiment Analysis simultaneously, but the results were worse than those obtained with the Logistic Regression estimator, having an F1-Score of around 0.63.

Threshold	F1-score
0.1	0.606 ± 0.029
0.2	0.602 ± 0.029
0.3	**0.739 ± 0.010**
0.4	**0.717 ± 0.006**
0.5	0.694 ± 0.025
0.6	0.661 ± 0.081
0.7	0.470 ± 0.126
0.8	0.406 ± 0.211
0.9	0.072 ± 0.089
1.0	0.111 ± 0.090

Table 4.13: Results of different threshold values in KNN classifier

4.3.4 Observations

During the development of this investigation, a knowledge database was created with information on relations between words in Portuguese (such as synonyms, hyponyms, and other relations) and the sentimental polarity of each word.

To help classify movies' synopsis with their genres, three different approaches were implemented, to test the way they would have an impact on results. The knowledge database was used to replace words with their synonyms and hyperonyms in the synopsis and to create new features based on the sentimental valence of the words. Those features are the number of positives, negatives, and neutral words and the total sum of those values of the words in the synopsis.

The results showed no improvements in using any of the approaches. To not discard this as an option, future investigation should be done by applying these methods to other datasets, and try to complete the knowledge database, once there are some words without relations or polarity assigned. We think the results could be better if the methods were applied to a problem using, for example, the English language, where more stop-word repositories, better stemmers, and more knowledge databases are available for usage and so there is more potential for improvement.

As it was said before and was seen in the latest tests, the results could have been better if another, more complex classifier was used, but in this case, the aim of the project was entirely on testing the two before-mentioned methods as extra preprocessing steps.

In the context of this internship, the results obtained in this prototype helped to gather some conclusions about the decisions to be made in the preprocessing, like what are the parameters that influence the final results the most and what are the techniques that should or not be incorporated. Using knowledge databases in a way similar to what was experimented is, therefore, discarded, since it did not show relevant improvements to simple preprocessing.

4.4 Additional Analysis

After implementing and documenting the project in the Knowledge-Based Systems discipline's context, a more in-depth analysis was performed to extract more indicators that may help in the continuation of this project. This in-depth analysis started by extracting

more metrics from the best tests performed previously to figure out where were the biggest difficulties in the classification process and then some concrete examples of synopses, their actual genre, and predicted genre were displayed, as well as the probabilities attributed to each class.

In figure 4.4 we can observe the precision and recall separately and one value that stands out is a recall of only 0.55 for class *"ação"*, which also lowers the F1-Score for the same class. By observing figure 4.5 we can see that the cause for that was the model only obtaining 252 positives when there were 455 to be considered. In figure 4.6 we can add that *"ação"* was mistaken with drama several times by the classifier, probably because the words used in both are sometimes similar.

In the confusion matrix present in figure 4.6, the concrete values are not displayed as it is not completely accurate to determine with the class was predicted and true given the use of a one-hot encoded matrix, because there is no concrete correspondence unless it is a true positive, but the colors obtained are approximate and give an idea of the categories among which the classifier was most confused.

Another interesting observation that can be done from these figures is that the recall value goes up for classes with more instances (we can see the number of instances for each class on the "support" column in figure 4.4). This may indicate that it would be interesting to perform class balancing in future tests to see if it has any impact on the recall values obtained since then all classes would have the same number of instances, although in this case, the discrepancy was big because the testing dataset is randomly selected since the number of instances of the original dataset did not vary as much.

	precision	recall	f1-score	support
ação	0.72	0.55	0.63	455
comédia	0.80	0.77	0.78	615
crime	0.82	0.81	0.81	674
drama	0.68	0.95	0.79	881
micro avg	0.74	0.80	0.77	2625
macro avg	0.75	0.77	0.75	2625
weighted avg	0.75	0.80	0.77	2625
samples avg	0.76	0.82	0.77	2625

Accuracy: 0.55

Figure 4.4: Additional metrics extracted using the previously optimized parameters for the SVM classifier

The printed examples were also interesting to visualize since they gave the opportunity of comparing the synopsis of a movie to the probability of each genre individually and see if the error was by a long shot or not.

Some of those examples are, considering a threshold of 0.38 for the probability values:

Tables 4.14, 4.16 and 4.17 are good examples of movies/series for which the genre classification would be completely correct. In table 4.15 we have an example where the correct genre ("comédia") had a probability that got very close to the threshold but did not make it and another probability ("drama") that was too high and resulted in a bad

Movie	Two Broke Girls T5
Synopsis	Max fica perplexa e curiosa quando o rapaz com quem anda a sair recentemente expressa hesitação em ter relações. E ainda, Sophie e Oleg têm dificuldade em engravidar. Classificação etária: M/12.
Predicted genre	Comédia
Probabilities	Ação: 0.1299, Comédia: 0.6290, Crime: 0.1504, Drama: 0.2474
Actual genre	Comédia

Table 4.14: First example and respective probabilities

Movie	Carbonaro Effect Special T6
Synopsis	Michael Carbonaro, um mágico experiente, realiza ilusões e truques de magia em pessoas inocentes em situações quotidianas.
Predicted genre	Drama
Probabilities	Ação: 0.1937, Comédia: 0.3564, Crime: 0.2508, Drama: 0.4208
Actual genre	Comédia

Table 4.15: Second example and respective probabilities

Movie	Cartas para Julieta
Synopsis	Uma jovem americana viaja até à cidade de Verona (de onde é originária Julieta Capuleto, da conhecida história de Romeu e Julieta) e decide aderir a um grupo de voluntários que respondem a cartas endereçadas a Julieta, dando conselhos amorosos aos seus remetentes.
Predicted genre	Comédia, Drama
Probabilities	Ação: 0.1912, Comédia: 0.4209, Crime: 0.1331, Drama: 0.5076
Actual genre	Comédia, Drama

Table 4.16: Third example and respective probabilities

Movie	Annie (2014)
Synopsis	Um remake do filme que encantou os mais novos na década de 80, que conta a história de Annie, uma jovem órfã, que mora com a sua maldosa mãe de acolhimento. Mas tudo muda quando um arrogante magnata a acolhe em sua casa, numa disfarçada ação de campanha.
Predicted genre	Comédia, Drama
Probabilities	Ação: 0.1446, Comédia: 0.4017, Crime: 0.1796, Drama: 0.6368
Actual genre	Comédia, Drama

Table 4.17: Fourth example and respective probabilities

Movie	Porky's
Synopsis	Uma das maiores comédias do seu tempo, que gerou duas sequelas imediatas e inspirou um número incalculável de comédias adolescentes simples, mais recentemente incluindo American Pie - A Primeira Vez (1999).
Predicted genre	Comédia, Drama
Probabilities	Ação: 0.0990, Comédia: 0.4744, Crime: 0.1799, Drama: 0.4814
Actual genre	Comédia

Table 4.18: Fifth example and respective probabilities

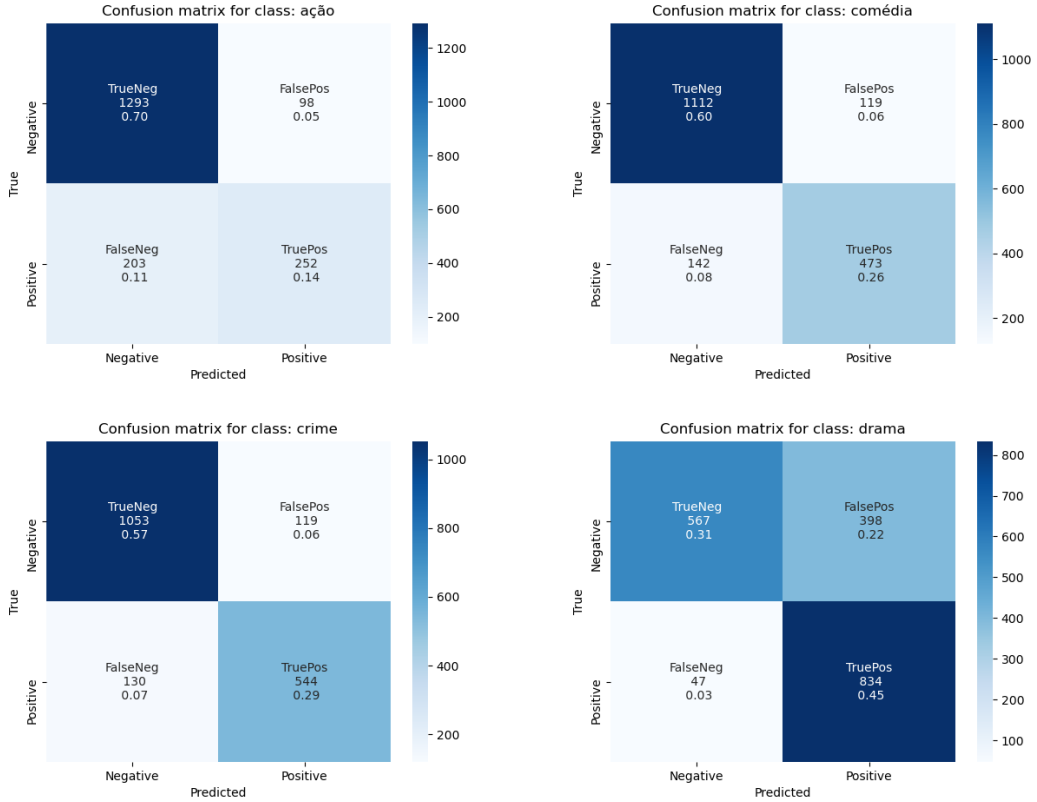

Figure 4.5: Confusion matrices obtained for each class considered

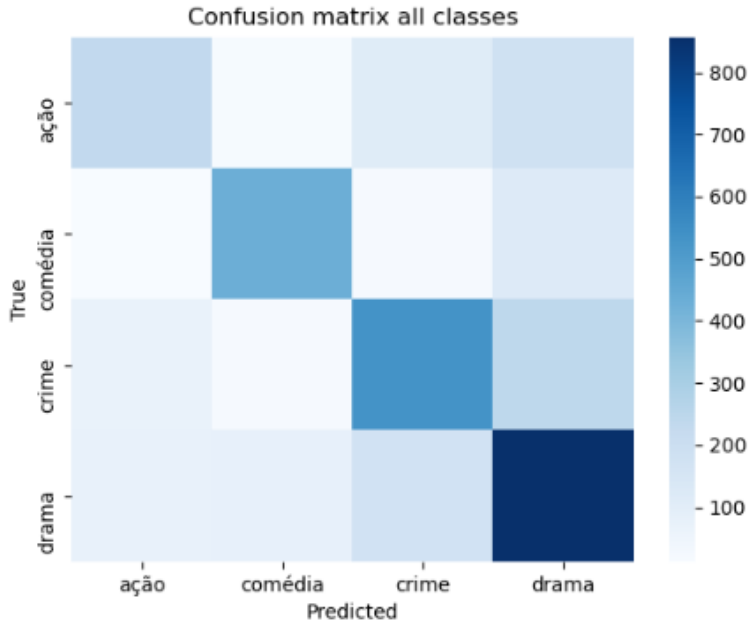

Figure 4.6: Confusion matrix for all classes

classification. Finally, in table 4.18 we have an example that sometimes happens in these multiclass problems, which is having an error in a portion of the predictions. In this case, "drama" was above the threshold and counted as a prediction when ideally it should not.

This page is intentionally left blank.

Chapter 5

TV programs classification

In this chapter, we will address the main focus of the second iteration, which was to develop a viable classifier for sports programs, that could be directly used in the company's systems for filling the gaps in the database entries related to that type of program. This chapter will describe in detail the workflow followed in the second iteration, the decisions made and all the steps taken for developing a viable classifier to be used on such a large scale.

5.1 Context

To begin the second iteration, we started by choosing the main focus for the remaining time. This was a very cautious process because the choice would influence the workflow of the whole iteration and we wanted this project to have the highest possible impact when used directly in the MEO systems while maintaining an interesting role in the research aspect.

For making sure the decision we were about to make was the right one, several meetings took place and a detailed analysis on what would have the most impact, directly and indirectly, was performed.

Although the most types of programs to be chosen had a certain percentage of classified programs, we decided it would be interesting to pick a more challenging route and chose the sports programs, since there were no programs at all in the sports channels with a classification, even though the cataloging process was already defined, as shown in table 4.4. This would also have a very significant impact both for labeling the sports programs and helping gather business opportunities, since the major sports channels in Portugal are usually Premium, and knowing a user's interests can help with making the right recommendation when trying to reach a deal.

Figure 5.1 shows the state of play at the beginning of the second iteration, we were at a point where we already had the knowledge provided by the preliminary experiments and wanted to reach a final product that could then be incorporated into the MEO ecosystem. This figure will be completed as we read through this chapter, to provide a better understanding of the whole course of action.

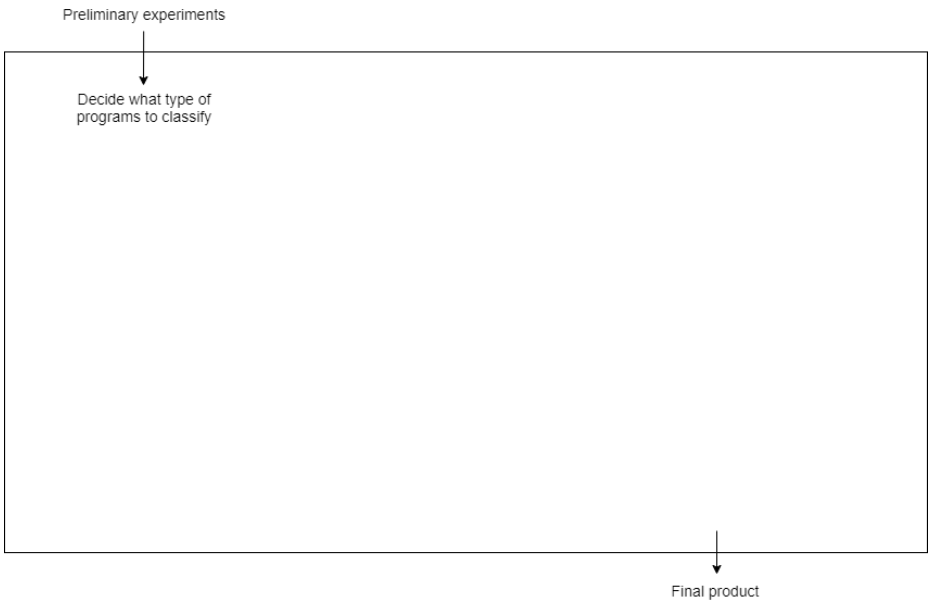

Figure 5.1: Decision making phase on the second iteration course of action

5.2 Dataset

In this project, we take into consideration the data corresponding to all the sports programs broadcasted from August 2020 to May 2021. This dataset contains 302,995 entries. From those entries, we select 75% (227,246 synopses) for training and testing and 25% (75,749 synopses) for validation.

For the train and test dataset, we remove the duplicates given their program key and name/synopsis pair, to avoid considering the same program more than once while training or testing the classifier. By removing the duplicates, we go from 227,246 entries to **28,036 unique entries**. From those, by key-word association (explained in the following section), we can directly classify **24,159 entries**, which corresponds to about 86% of the programs obtaining a label by key-word association.

As expected, some labels have much more occurrences than others and that can be easily visualized in table 5.1. The "Futebol" label has much more occurrences than any other and some other labels like "Desportos inverno" or "Dança" have none or almost none. This is an issue that we will discuss further, in section 5.4.6.

Futebol	51.29%	Combate	1.48%	Radical	0.33%
Motorizado	13.11%	Snooker	1.35%	Jogos olímpicos	0.24%
Basquetebol	11.44%	Andebol	1.19%	Futebol americano	0.23%
Golfe	2.81%	Voleibol	1.02%	Atletismo	0.14%
Futsal	2.70%	Surf	0.87%	Triatlo	0.11%
Ténis	2.55%	Rugby	0.84%	Hipismo	0.06%
Ciclismo	2.45%	Biatlo	0.72%	Vela	0.06%
Esqui	2.28%	Snowboard	0.44%	Dança	0.01%
Hóquei	1.96%	Skateboard	0.34%	Desportos inverno	0.00%

Table 5.1: Label occurences in the dataset

On the other hand, for the validation dataset, we want the data to be classified to be

as close as possible to the entries that will be taken directly from the database. Therefore, there is no removal of duplicates since in a real application there will be some sports programs that appear more than once (for example a daily news program), and all the 75749 synopses are considered.

5.3 Approach

Given the fact that, by the start of this iteration, we had no classified data, the approach for this problem had to start by defining how we were going to obtain the labeled data to use in both training and testing.

5.3.1 Classifications by key-word association

The first obstacle when choosing the sports programs as our main focus was to think about how we could get enough classified sports programs to support the training, testing, and validation of a classifier in a viable way.

As defined in table 4.4, the cataloging process related to the sports programs would be "based on words or key-expressions, univocal to the universe of the modality/sport that appears in the titles of sports channels and sporting events that appear on generalist channels". That line of thought inspired the idea of creating a dictionary associating certain key-words with the corresponding sport, rather than classifying a large number of samples by hand, since it would take less time and also, in this particular context, there are competitions and terms whose name is univocal to one sport only.

Therefore, the training script starts by classifying a set of unclassified sports programs through key-word associations that are defined in a Python dictionary. Those programs are then submitted to the usual classification process (preprocessing, feature extraction, and classification).

In figure 5.2 we can observe part of the associations that are made. This Python dictionary is a mapping of keys/sports, which correspond to the labels, to values/key-words. If any of those key-words are present in a synopsis, then the program will be classified as the corresponding sport since the values of each sport are univocal and distinct.

In this figure, we also notice that we can have sequences of words as values. That is possible because we started taking N-Grams (2.2.1) into consideration when extracting Term Frequency - Inverse Document Frequency (TF-IDF) features. This is possible directly through a parameter when calling the TF-IDF Vectorizer, by defining the N-Gram range we want to take into consideration, for example, if we define the tuple (1,3), we will have 1 word, 2 words, and 3 words tokens.

5.3.2 Implementation

Since the training of the classifier would be highly dependent on the associations made beforehand, the dictionary where those associations are defined would be constantly subject to improvements, like insertion of new genres, dictionary enrichment, and corrections based on output analysis.

```
'Futebol':['bundesliga', 'laliga', 'la liga', 'primeira liga', 'ucl',
           'uefa champions league', 'premier league', 'ligue 1',
           'serie a', 'supertaça cândido de oliveira', 'taça de portugal',
           'segunda liga', 'liga italiana', 'superliga turca', 'youth league',
           'liga das nações','copa do brasil', 'campeonato paulista', 'mundial 2022',
           'fut. fem','jogo de preparação', 'liga revelação', 'liga dos campeões',
           'juvenis','copa 90','liga europa','jupiler','eredivisie','libertadores',
           'champions league','taça da alemanha','liga belga',
           'taça de itália','superliga chinesa','taça da liga argentina',
           'liga holandesa','scottish premiership','liga polaca','mls',
           'copa sul-americana','liga inglesa','liga mexicana','taça do rei', 'superliga suiça',
           'taça de inglaterra','taça da liga','k-league'],
'Motorizado':['automobilismo','gp','motorizados','motocross','superbike',
              'motociclismo', 'fórmula e', 'fórmula 1', 'nascar', 'wec', 'automotive',
              'indycar','f2','félix da costa','miguel oliveira','rally','carros turismo',
              'wtcr','motores','bezerke','karting','wrc','f1','auto foco','volante','automóvel',
              'speedweek','autódromo'],
'Radical':['radical','bmx','x games','radicais'],
'Combate':['bellator','mma','ufc','boxe','kickbox','wrestling','muay thai',
           'judo','artes marciais'],
'Ténis':['atp world tour','roland garros'],
'Basquetebol':['nba','fiba','basketball','euroliga','eurocup','playoff playback',
               'marquee matchup','baloncesto','dunk contest'],
```

Figure 5.2: Some examples of the final version of the associations dictionary

Insertion of new genres

While adding key-words to their corresponding sports, it was easy to notice that some programs, even though related to a particular sport, would be left out because that sport was not considered as a valid label for classification. Consequently, the labels set would grow gradually as new sports worth including were found. This was a continuous task throughout the whole project because as new synopsis were verified, new sports would be worth considering for classifying a given program. These new labels were obtained either because a sport was not classified before or because it was grouped with other sports when it was worth being considered a different one, given the size of their community or the importance they would have while considered in separate ("Skateboard" for example, which was before considered as "Radical"). Every new label was added either as a proposal with approval of the MEO team or as a suggestion by them, since this implies certain changes to the set of possible sub-genres shown before, in table 4.4.

As defined in table 4.4, the first set of labels was formed by:

{"Futebol", "Motorizado", "Radical", "Combate", "Ténis", "Basquetebol", "Futsal", "Golfe" and "Atletismo"}

And after the insertion of new genres, this set would become:

{"Futebol", "Motorizado", "Radical", "Combate", "Ténis", "Basquetebol", "Futsal", "Golfe", "Atletismo", "Futebol americano", "Snooker", "Rugby", "Ciclismo", "Andebol", "Voleibol", "Hóquei", "Surf", "Esqui", "Biatlo", "Triatlo", "Jogos Olímpicos", "Dança", "Desportos inverno", "Hipismo", "Skateboard", "Snowboard" and "Vela"}

Even though this decision would increase the number of classified synopses that could be used for training, testing, and validating the classifier, it would highly increase the challenge of correctly classifying the programs since more categories introduce more uncertainty for the classifier and it becomes harder to have a univocal correspondence between sports and their key-words.

Dictionary enrichment

In addition to the dictionary key set, the value set has also changed during the process, mostly by the addition of key-words. This was a slow process because the dataset had to be constantly validated to check if the key-word association was working like planned since the dictionary was also constantly changing.

After creating the dictionary by adding the obvious words, there were still lots of programs that were not being classified, and so, there were some techniques that were used to obtain new key-words for each sport. Among those techniques, the spotlight goes to Latent Dirichlet Allocation (LDA) and the use of Word Clouds.

Latent Dirichlet Allocation

Latent Dirichlet Allocation (Blei et al., 2001) is a generative probabilistic model for collections of discrete data (usually text corpora) which can model several topics that contain sets of words that are considered related. This topic modeling approach can find natural groups of items and can help to discover hidden themes in the collection of synopses and organize/summarize them.

In table 5.2, we can observe some examples of topics that are correctly generated. These were created using 27 topics (number of sports considered) and we chose to show the 10 top words for each topic. Picking this number of topics, because the dataset is unbalanced, resulted either in existing more than one topic fully related to one of the sports with more samples or to having sports that are less likely to appear unexpectedly on some topics. Either way, the results produced by this parameter were interesting since they give more emphasis to the most important sports and allow observing those who, even with fewer samples, manage to appear on some topics. In topics 1, 6, and 8, for example, all the words are related and correctly grouped and in some other topics like 7 and 10, we can even observe the grouping of teams from the same league. In topic 7 there is a reference to Barcelona, Atlético, Real Sociedad, Villareal, Levante, and Granada, all from LaLiga, and in topic 10, the Premier League teams Manchester United, Manchester City, Liverpool, West Ham, Tottenham, Leicester, and Leeds are also referenced.

Topic 0	mundo, esqui, alpino, manga, slalom, masculino, feminino, basquetebol, livre, saltos
Topic 1	automobilismo, corrida, series, fia, gp, nascar, superbike, cup, motociclismo, mundial
Topic 2	holandesa, futebol, neste, americano, nfl, ajax, draft, psv, 0000, face
Topic 3	nações, real, madrid, smartbank, la, inglaterra, rali, ralis, laliga, mundo
Topic 4	primeira, superliga, turca, benfica, sc, boavista, revelação, especial, rio, ave
Topic 5	voleibol, fc, porto, basquetebol, benfica, hóquei, fórmula, análise, andebol, 0000
Topic 6	nba, basquetebol, principal, mundo, season, regular, 0000, la, finals, lakers
Topic 7	laliga, la, santander, barcelona, futebol, atlético, sociedad, villarreal, levante, granada
Topic 8	motogp, corrida, motociclismo, gp, pilotos, velocidade, máxima, livres, treinos, português
Topic 9	italiana, juventus, encontros, inter, campeões, ligações, palco, ac, youth, milan
Topic 10	league, premier, united, man, city, liverpool, west, tottenham, leicester, leeds
...	...

Table 5.2: Some topics generated using Latent Dirichlet Allocation

Despite not all topics being perfectly organized, running this algorithm and showing a higher number of top words helped gathering new terms for the association dictionary. We ran the algorithm several times because since it is not deterministic, the results vary and so do the conclusions we may take from them.

Word Clouds

Another method that was very useful for gathering key-words was the generation of Word Clouds.

A Word Cloud is, just as the name implies, a set of words with associated weights, usually represented by an image composed by that set of words, in which the ones with more relevance are in a bigger font than the ones less relevant.

Like in LDA, this method is applied after preprocessing so that the words displayed are more likely to not be stop-words and be more susceptible to represent only one of the 27 sports that are considered.

When looking at the Word Clouds containing words from all genres, since the main objective of its implementation was to gather an overview and suggestions of key-words for each sport, we decided that it would be very useful to obtain an image for each one of the labels, so that the potential key-words would be considered one at a time. This also allowed us to look for potential genres that were being misclassified, if there were words in the image that did not belong to the label that was being analyzed.

In figure 5.3, we have an example of the Word Cloud obtained for the "Futebol" label, which is the one with more occurrences in the train and test dataset, as referred to in section 5.2. By observing the words we can rapidly tell a few that are candidates to be key-words for the "Futebol" label, like "bundesliga", "laliga" and "brasileirão", for example, which are national championships whose name is only related to football. Also, in the earlier stages, observing the clouds helped to identify mistakes, by spotting outliers and tracking their origin.

Figure 5.3: Word Cloud obtained for "Futebol"

Moreover, in figure 5.4 we can also observe the Word Clouds obtained for the two upcoming labels with the most occurrences, which are "Basquetebol" and "Motorizado". These images are interesting because they provide a very brief and simple overview of all entries in the dataset, one label at a time, and help a lot with extracting key-words for each of them because the words with the most occurrences are bigger than the others. For example, in the "Basquetebol" Word Cloud, we can see that "nba" has an even bigger font than "basquetebol", which is the label itself, and since "nba" is univocal to this label, it is a very powerful key-word when classifying with associations, because it will allow a lot of programs to be classified.

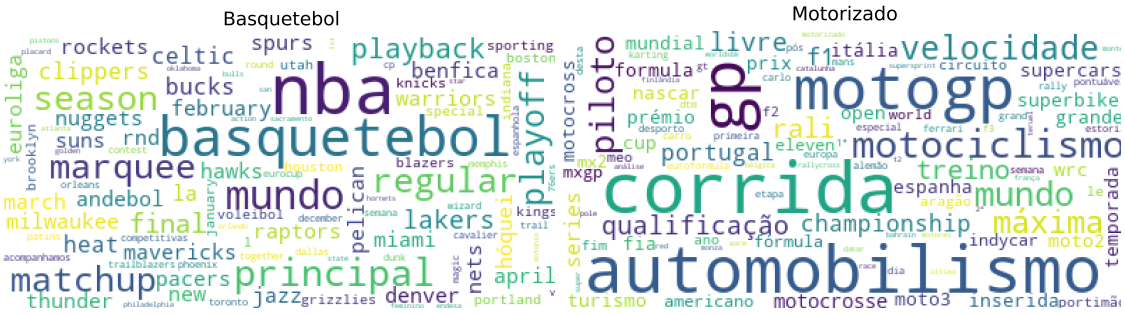

Figure 5.4: Word Clouds obtained for "Basquetebol" and "Motorizado"

Corrections based on output analysis

Due to the classification by word associations being made automatically, it was very important to validate the dataset and make sure that it was suitable for training a classifier that is going to be used on a very large scale.

Accordingly, the labels applied had to be revised to make sure they were being placed correctly. This method was simplified by dropping the duplicated synopsis and analyzing through Microsoft Excel pivot tables, which allow the user to obtain interesting metrics easily, by filtering the data with columns values and creating custom tables with the desired rows and columns, for example.

For helping with correcting possible mistakes and have some terms of comparison for our word association method, a dataset with some programs classified by hand was provided by MEO, after the associations' dictionary had been revised a few times. With this data, it was then made possible to compare the data obtained by word association to the examples classified by hand and then analyze the cases in which the labels did not match, individually.

The final version of the key-word association process can be understood more easily by observing the scheme presented in figure 5.5.

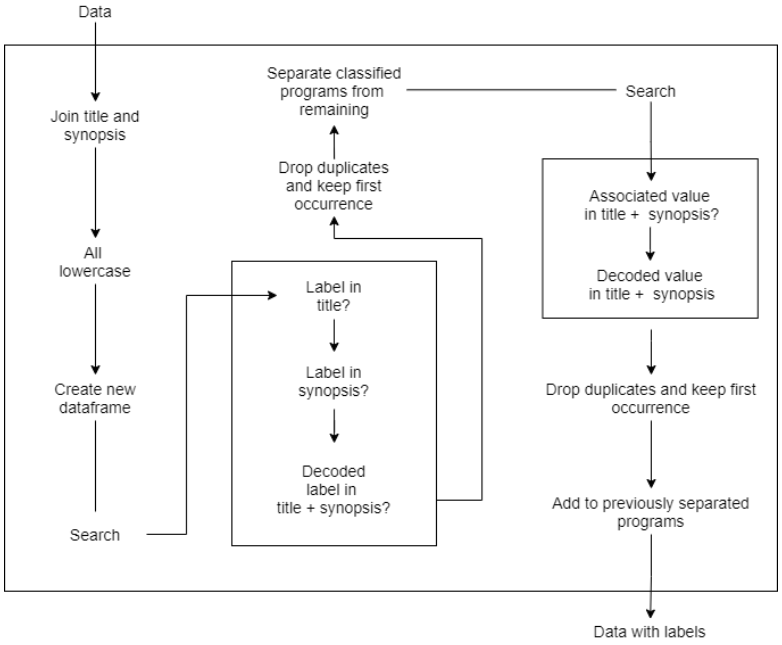

Figure 5.5: Key-word association process pipeline

Insertion of new stop-words

In the context of sports programs, besides the usual stop-words like text connectors, for example, some other words need to be taken into account, since they apply for all labels and appear in a large amount of synopsis. Some of these words are, for example, "transmissão" or "direto".

For aiding with the process of adding new stop-words manually to the file considered beforehand, in 4.3.2, we grouped the words with the most total TF-IDF value in an Excel file (it is important not to confuse with value in a single document, since then we would be removing the most valuable words to the classification), that would also help when understanding why certain programs were being considered in a way different from what would be expected, by identifying words present in the synopsis with a high ranking in the file mentioned.

In table 5.3, there is a representation of 50 stop-words that were added using the before mentioned method. Some of them were added because of the context we are working with since they refer to more than one sport and may lead to a more difficult classification.

liga	direto	transmissão	contar	jogo
encontro	taça	assista	campeonato	camp
resumo	nacional	jornada	melhores	dedicado
espaço	acompanhe	espetacular	antevisão	torneio
momentos	cada	onde	prova	copa
emoções	maior	competição	cobertura	confere
disputados	provas	ronda	jogos	faz
alta	campeonatos	dadas	curvas	game
programa	golos	melhor	magazine	todos
desportiva	protagonistas	preparação	nenhum	lance

Table 5.3: Added stop-words

Club identification

Besides performing the classification of the programs by their sport, it was suggested that it would be useful to implement a simple mechanism that identifies the clubs that are present in a certain program and flags them in the output file.

The implemented mechanism consists of key-word detection, where if the club's name or any associated name is found, then the club is flagged as present on that program. Besides the associated names, there's also a Python dictionary containing the exclusions for when the name of a club is included in the name of another club. For example, for this task, we considered the three clubs with the most supporters in Portugal, which are Benfica, Porto, and Sporting, and "Porto" is included in "Leões de Porto Salvo" which is a well-known futsal club and even "Sporting" is included in "Sporting de Braga" which is another well-known football club.

This task provides data that can be of high value for business opportunities. As an example, if a client watches mostly Benfica matches but does not subscribe to the "Benfica TV" channel, there is a good opportunity in recommending the channel to this client. As this is also included in the final version of our classifier, its output will be addressed in section 5.5.

5.3.3 Bidirectional Encoder Representations from Transformers (BERT) approach

Since the research made about BERT, it was defined that it would be one of the main points of focus for the second iteration because it also produces interesting results as a feature extractor, and having good features is as important as having a good classifier.

When it comes to implementing transformer architectures, one of the platforms that needs to be taken into account is Hugging Face, given its well-structured documentation, simplicity of use, and variety of supported models.

Hugging Face

Hugging Face[1] is an organization which provides Natural Language Processing (NLP) technologies. It has a growing community, in particular around the Transformers[2] library, which already supports 63 different models, among which is Google's BERT.

This package allows to easily implement well-known transformer architectures for a wide variety of NLP tasks like Fill-Mask, Question Answering, Summarization, Sentence Similarity, Feature Extraction, among many others.

The website allows us to chose the task we want to perform and provides a very intuitive interface for choosing the most used model given a language, python library used (PyTorch, TensorFlow, among others), datasets used for training and licenses.

Zero-shot classification

Among the NLP tasks covered by Hugging Face, one that quickly grabbed our attention was the "Zero-Shot Classification". This pipeline allows us to select a model and define a group of labels we want to consider, feed it with text segments we want to classify and it returns the labels ordered by probability and the score of each label (given the sum of all scores is 1).

Ideally, this would be the best-case scenario for the task we want to perform since it requires no additional information and no preprocessing at all, just the synopsis and the labels.

Nevertheless, both models we tested did not achieve good results right from the first tests performed. Those models were facebook's BART and BERTimbau from NeuralMind (Souza et al., 2020).

When testing the BART model, since it was trained in the Multi-NLI dataset, which is a collection of sentence pairs annotated with textual entailment (directional relation between text fragments) that contains samples in English only, the results are bad since both the synopsis and the labels are in Portuguese and correctly translating them would result in an added complexity to our problem. As we can observe in figure 5.6, there are examples where the classification is correct, like in the first one even though the score is not too high, but also examples where a simple synopsis is labeled incorrectly, as the second. We only present examples and not results on the whole dataset since this method was only an exploratory experiment and was considered not worth investing too much time into

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

[2] `https://huggingface.co/transformers/`

since the model was trained for the English language and our dataset was in Portuguese.

```
Sequence: Liga BWIN - Sporting vs Benfica
Best label: Futebol
Score: 0.21762388944625854
```

```
Sequence: NBA - LA Lakers vs OKC Thunder
Best label: Outro
Score: 0.12486924976110458
```

Figure 5.6: Examples of classifications obtained using the BART model

For the BERTimbau model, since it was not trained for Zero Shot Classification, the results obtained seem to be random. In figure 5.7, we can see that a wrong label was chosen with a low score for both examples, and by observing the remaining scores we note that they are obtained randomly.

```
Sequence: Liga BWIN - Sporting vs Benfica
Best label: Skateboard
Score: 0.06206781789660454
```

```
Sequence: NBA - LA Lakers vs OKC Thunder
Best label: Skateboard
Score: 0.0765073373913765
```

Figure 5.7: Examples of classifications obtained using 'bert-base-portuguese-cased'

These were the only two models that were tested because there are not any models trained with the Portuguese language for this pipeline and given that our two choices were the default model for the task (one of the most used) and the most used Portuguese model.

Feature extraction with BERTimbau

Given the results obtained while testing the Zero-Shot Classification were far from ideal, for the next step we decided to use the 'feature-extraction' pipeline, which computes the features for a given text so that we could compare the state-of-the-art technology with the hitherto used TF-IDF when it comes to feature extraction.

The model used for this task was, once again, the 'bert-base-portuguese-cased' from NeuralMind also known as BERTimbau, which is a pre-trained BERT model for Brazilian Portuguese that achieves state-of-the-art performances in Named Entity Recognition, Sentence Textual Similarity, and Recognizing Textual Entailment.

Since this ended up being our final approach using BERT and the one compared to the previous approach, the results with our dataset are presented and discussed in section 5.4.2.

5.4 Tests and results

As the classification process has several different phases, we decided to test them all separately. The metric we chose for comparing the results was the F1-Score, and we present the mean and standard deviation obtained in 15 runs made for each test, and the subsequent tests are always performed under the best variables obtained. Since the results here were much closer than in section 4.3.3, we also decided to use 4 decimal digits instead of the previous 3. The testing process started by testing all the parameters related to preprocessing, then the feature extraction techniques and parameters associated, and finally some different classifiers. Here we also present some results obtained when testing the classifier on a dataset with programs from generalist channels, using over-sampling techniques on the sports dataset, and also by not considering the labels with fewer occurrences for training

our classifier. The classifier used until the classifier's test was always the Logistic Regression since it obtained good results and its running speed allows to make more repetitions and test more values.

5.4.1 Preprocessing

We started by testing three parameters, two directly related to the preprocessing phase of the classification, which is the usage of stop-word removal and application of stemming, and one related to the dataset fields used (that can be the title, synopsis, or both). First, we test the stop-word removal being applied or not, and then we select the option that obtained the best result and test the application of stemming in the dataset.

stop_words	**F1-score**
True	**0.9880 ± 0.0016**
False	0.9876 ± 0.0011

Table 5.4: Results obtained for usage of stop-word removal

stemming	**F1-score**
True	0.9873 ± 0.0011
False	**0.9881 ± 0.0009**

Table 5.5: Results obtained for usage of stemming

By observing tables 5.4 and 5.5, we can perceive that the best results were obtained using stop-word removal and not performing stemming. We can also not that the results are very high, mainly because we already started by fixing parameters by their best value obtained in previous tests, and also, unlike in the movies dataset used in section 4.3.3, this is not a multi-label program, since every program is only associated with one label, which makes the classification process easier.

Fields considered	**F1-score**
Synopsis	0.9880 ± 0.0013
Title + Synopsis	**0.9889 ± 0.0015**
Title	0.9884 ± 0.0009

Table 5.6: Results obtained for considering title, synopsis or both

Testing the fields considered for input to the classifier, the option that provided the best results was to use both the title and synopsis at the same time, which could be expected because this way the feature extraction technique and the classifier can have access to more information about the program and that leads to a better classification.

5.4.2 Feature extraction

In this phase, the parameters and techniques related to the feature extraction phase were tested. We started by testing the three parameters related to the TF-IDF algorithm that can be changed in the program, which are the Max-DF (maximum document frequency), which defines the percentage of synopsis in which a word can be for it to be considered as a feature, the Max-features, that indicates the maximum number of features to be considered, and the N-Gram range, that relates to how many words can be considered as

a feature, for example, if we have an N-Gram range of (1,3), the features can be single words, 2-grams and 3-grams.

Also, as in Max-DF and Max-features, we are working with numeric values, we start by testing the values with larger spacing and then smaller spacing. Therefore, for the Max-DF variable, the range of the first tests is from 0.1 to 1 with 0.1 spacing and then between the two best consecutive values with 0.01 spacing. For the Max-features the tests are performed similarly, from 1000 to 10000 with 1000 spacing in the first group of tests and then with 100 spacing between the two best consecutive values, like in the tests performed in section 4.3.3.

max_df	**F1-score**
0.1	0.9857 ± 0.0013
0.2	0.9866 ± 0.0010
0.3	0.9866 ± 0.0008
0.4	0.9864 ± 0.0009
0.5	**0.9875 ± 0.0010**
0.6	**0.9871 ± 0.0010**
0.7	0.9863 ± 0.0012
0.8	0.9868 ± 0.0011
0.9	0.9864 ± 0.0012
1.0	0.5093 ± 0.0006

Table 5.7: Results obtained for different values of Max_Df

The tests performed for the Max-DF variable (table 5.7) indicated the best value to be 0.53 with an F1-Score of 0.9876. This value was close to what was the best in the preliminary experiments and it is not a surprise. After all, this indicates that it is best to leave out the words that appear in more than 53% of the synopsis, which have less value for the classification because of being present in a lot of examples, which are not all from the same label because the label "Futebol" is associated with 51% of the programs.

max_feat	**F1-score**
1000	0.9862 ± 0.0010
2000	**0.9873 ± 0.0008**
3000	**0.9880 ± 0.0014**
4000	0.9866 ± 0.0011
5000	0.9858 ± 0.0009
6000	0.9854 ± 0.0013
7000	0.9854 ± 0.0012
8000	0.9852 ± 0.0012
9000	0.9845 ± 0.0011
10000	0.9845 ± 0.0011

Table 5.8: Results obtained for different values of Max Features

For the Max-features variable (table 5.8), the best value was 2200 with an F1-Score of 0.9883. Even though this dataset had a lot of samples, this value can be justified by the type of programs that are present, since, in the sports programs, the synopsis is usually very generic, being most of the time just the name of the competition and the teams/athletes that are competing, with little extra information.

With relation to the N-Gram range (table 5.9), the best result was obtained when using a range of (1,2), which can be explained by most of the competitions' names being only

n_grams	**F1-score**
(1,1)	0.9865 ± 0.0010
(1,2)	**0.9887 ± 0.0013**
(1,3)	0.9874 ± 0.0009
(1,4)	0.9883 ± 0.0010
(1,5)	0.9873 ± 0.0012

Table 5.9: Results obtained for different N-Gram sizes

one or two words, like for example "Roland Garros", "Liga Placard", "Premier League", "Bundesliga", "Ligue 1", "Primeira Liga", among others. Another observation we can make is that using N-Grams was always favorable, since using a range of (1,1) would be the worst choice for this parameter.

Feature extraction technique	**F1-score**
TF-IDF	**0.9887 ± 0.0013**
BERT	0.9667 ± 0.0054

Table 5.10: Results obtained for different feature extraction techniques

In the feature extraction technique, we tested the previously implemented TF-IDF with the best values from the previous tests in this section against BERT using the 'bert-base-portuguese-cased' model. Even though both algorithms performed very well, the TF-IDF algorithm performed better, considering both F1-Score and the time taken to run the script. Since we were curious about this result, we also checked some example outputs obtained while using BERT from classification, and noticed that even though the results were good, there were some cases where the prediction would be completely off and the probability associated was very high. This was worse than what was obtained using TF-IDF since the results obtained, when wrong, usually had a low probability associated and were more interpretable. One important note is that, given the computational power available, the dataset used for training using BERT was limited to 5000 samples so that we could run 15 repetitions in a manageable amount of time and without causing any overheating issues.

5.4.3 Classifiers

For the classifiers, even though the Logistic Regression used until now was obtaining good results and was fast enough to be applied in a large database, running about 200 iterations per second in the system used, we wanted to check if there was another classifier that would be worth implementing considering both the results obtained and time taken. So, we used the lazy_predict package that allows us to test 26 different classifiers on the same dataset easily, while also providing insightful metrics for comparing them.

The system used for performing these tests was an MSI GE73VR-7RF Raider laptop with the following specs:

- CPU - Intel Core i7 - 7700HQ
- GPU - GeForce GTX 1070 8GB GDDR5
- RAM - 16GB DDR4 (2400MHz)

Also, since the objective of this experiment was to compare various algorithms, the dataset for both the TF-IDF and BERT was restricted to the programs displayed between

February and March 2021, since we were using a laptop and both the feature extraction using BERT and some of the classifiers tested would take too long to run in these conditions.

When it comes to results, by observing table 5.11, it is interesting to notice that the Logistic Regression algorithm, apart from being the one with the highest F1-Score, is also one of the techniques with the least time taken to perform the classification, both while using TF-IDF or BERT for the feature extraction. The Random Forest and Extra Trees Classifiers also obtained interesting results when it comes to both F1-Score and time taken, but not as good as the Logistic Regression.

	TF-IDF		BERT	
	F1-Score	Time taken (s)	F1-Score	Time taken (s)
Logistic Regression	0,9827	10,304	0,9732	6,340
XGBClassifier	0,9803	371,865	0,9474	182,449
RandomForestClassifier	0,9791	10,132	0,9151	32,667
ExtraTreesClassifier	0,9751	21,645	0,9156	7,055
PassiveAggressiveClassifier	0,9735	28,880	0,9709	9,850
LinearSVC	0,9731	1325,216	0,9721	74,043
DecisionTreeClassifier	0,9723	10,907	0,8114	15,235
BaggingClassifier	0,9712	30,021	0,8785	107,078
LGBMClassifier	0,9700	21,830	0,9407	433,981
RidgeClassifierCV	0,9670	15,099	0,9687	3,800
Perceptron	0,9640	17,351	0,9639	6,065
CalibratedClassifierCV	0,9622	5174,286	0,9691	342,204
RidgeClassifier	0,9611	4,027	0,9663	1,212
ExtraTreeClassifier	0,9577	2,251	0,7591	0,829
GaussianNB	0,9427	6,684	0,8832	2,474
LinearDiscriminantAnalysis	0,9299	61,813	0,9663	4,276
NearestCentroid	0,9200	2,167	0,8106	0,871
SGDClassifier	0,9185	15,669	0,9272	7,247
SVC	0,9129	594,402	0,9620	42,807
KNeighborsClassifier	0,8397	57,332	0,9426	27,272
QuadraticDiscriminantAnalysis	0,7235	10,500	0,4387	3,483
BernoulliNB	0,7194	2,391	0,8444	1,164
AdaBoostClassifier	0,4746	36,100	0,3340	70,375
LabelSpreading	0,3268	4,995	0,2605	5,948
LabelPropagation	0,3268	4,862	0,2605	5,329
DummyClassifier	0,2884	1,904	0,2884	0,733

Table 5.11: Results obtained for testing different classifiers given the F1-Score and time taken, in seconds

5.4.4 Using the best parameters from previous tests

By using the best parameters obtained in the before mentioned tests presented in this section, we managed to achieve interesting results.

Figure 5.8 shows the classification report obtained, where we can check the precision, recall, F1-Score, and support (number of samples available in the testing dataset) for each label. The observation of this report shows that in general, the F1-Score is high for most classes, sometimes even reaching 1. In the labels with the least amount of samples, as we have a very high number of samples to consider, the F1-Score tends to be very low, sometimes reaching 0, but it almost doesn't affect the general F1-Score because it only happens in labels with less than 4 samples for testing. However, in sections 5.13 and 5.4.7 we will discuss two possible solutions to deal with this type of problem.

	precision	recall	f1-score	support
Andebol	1.00	0.99	0.99	83
Atletismo	1.00	0.20	0.33	10
Basquetebol	1.00	1.00	1.00	810
Biatlo	1.00	1.00	1.00	52
Ciclismo	1.00	0.99	0.99	176
Combate	1.00	0.90	0.95	104
Dança	0.00	0.00	0.00	1
Esqui	1.00	0.99	1.00	165
Futebol	0.98	1.00	0.99	3615
Futebol americano	1.00	0.75	0.86	16
Futsal	1.00	1.00	1.00	192
Golfe	1.00	0.98	0.99	203
Hipismo	0.00	0.00	0.00	4
Hóquei	1.00	1.00	1.00	138
Jogos olímpicos	1.00	0.88	0.93	16
Motorizado	0.99	0.98	0.98	947
Radical	0.80	0.70	0.74	23
Rugby	1.00	0.95	0.97	61
Skateboard	0.95	0.79	0.86	24
Snooker	1.00	1.00	1.00	97
Snowboard	0.97	0.94	0.95	32
Surf	1.00	0.98	0.99	63
Triatlo	1.00	0.86	0.92	7
Ténis	1.00	0.98	0.99	185
Vela	1.00	0.50	0.67	4
Voleibol	1.00	0.99	0.99	70
micro avg	0.99	0.99	0.99	7098
macro avg	0.91	0.82	0.85	7098
weighted avg	0.99	0.99	0.99	7098
samples avg	0.99	0.99	0.99	7098

Figure 5.8: Classification report

Furthermore, in figure 5.9, a confusion matrix containing all the labels is presented. By looking at it we can easily spot a diagonal line with darker blue tones. This was the wanted result since this line shows the correct predictions, and also, out of this line, even though some squares are darker than others, they do not represent any major classification mistake since almost all of them are a very light blue, which indicated a low number of misclassifications.

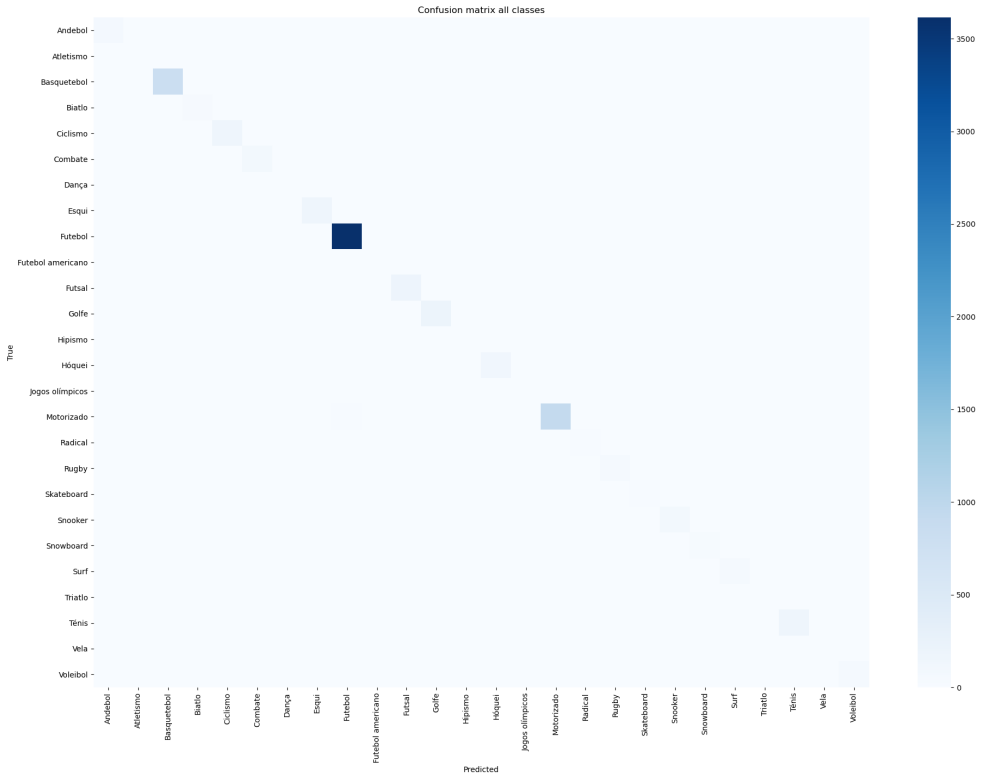

Figure 5.9: Confusion matrix for all classes

5.4.5 General programs exercise

One exercise that was also made for testing our sports programs classifier was to submit general programs synopsis and check the probabilities attributed by the classifier to each sport. It was interesting to observe that usually when the program is displaying a sport it results in a high probability for that sport and when it is related to any other type of program, it usually results in a low probability for all the sports. This is another possibility of application for this classifier and the tests made also helped to check if more words needed to be added to the stop-word list or other improvements that had to be made.

Figure 5.10 shows the percentage of programs that remain unclassified given the threshold defined, which works as a minimum probability that needs to be achieved by a label for it to be attributed so that a sports label is only assigned to a sports program. This graph is useful if we have information about the percentage of sports programs in the generalist channels, for example, if we know that there are 20% sports programs, the right threshold value will be around 0.55, even though there are still going to be misclassifications.

In table 5.12, there are illustrated 3 examples on which the classifier performed well, under a 0.5 threshold. For the first example, since the program corresponded to a sport, the label "Voleibol" was assigned with a very high probability. On the other two, since they are two programs that are not directly related to sports, no label had a probability over 0.5 so there were no labels attributed and the programs remain unclassified (by this classifier, could then be classified by another).

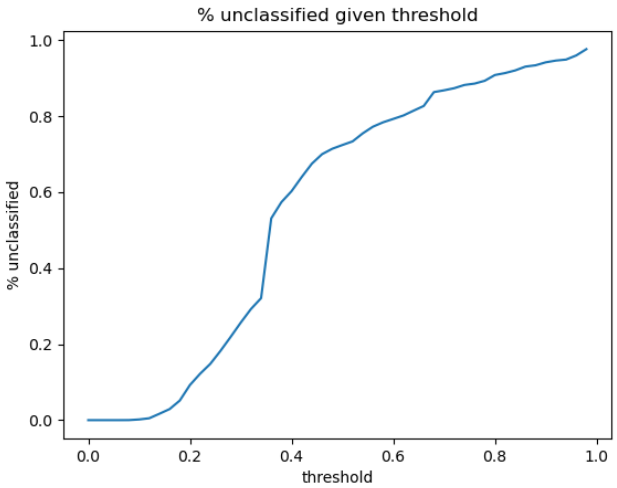

Figure 5.10: Percentage of unclassified programs given threshold

	Example 1	Example 2	Example 3
Title	Voleibol: AJM Fcporto x Vilacondense	Jornal da tarde	Somos Portugal
Synopsis	Espaço dedicado a transmissões desportivas	As notícias que marcam a atualidade nacional e internacional	Um programa que leva Portugal a todo o país, através das suas festas e romarias, das tradições, da gastronomia, da música e da boa disposição
Label	Voleibol	———	———
Probability	0.9501	———	———

Table 5.12: Some examples of outputs obtained with programs shown in generalist channels

5.4.6 Over-sampling

Given the distribution of the labels, shown in table 5.1, an approach we considered to be worth implementing and testing was to perform over-sampling so that all the labels would have the same number of occurrences as "Futebol".

The results in table 5.13 showed that using an over-sampling technique is beneficial to the results, and figure 5.11 also shows a higher F1-Score than before for the least populated classes. However, during the validation phase we noted that, even though the accuracy was higher, the errors obtained would be more relevant to the final classification, since it would misclassify more programs with more audience. For example, using over-sampling, since the classifier, during the training phase, received the same samples of every label, it may classify "Futebol" as "Dança", while before it would be more likely to do the opposite and classify as "Futebol" when it should be another label, which would not have as much impact since "Futebol" is the most watched sport in Portugal, and so, its good labeling is a priority.

Over-sampling	**F1-score**
None	0.9887 ± 0.0013
RandomSampling	0.9949 ± 0.0011
SMOTE	$\mathbf{0.9950 \pm 0.0007}$

Table 5.13: Results obtained for using Over-sampling techniques

	precision	recall	f1-score	support
Andebol	1.00	1.00	1.00	83
Atletismo	1.00	1.00	1.00	10
Basquetebol	1.00	1.00	1.00	810
Biatlo	1.00	1.00	1.00	52
Ciclismo	1.00	0.99	0.99	176
Combate	0.95	0.95	0.95	104
Dança	0.14	1.00	0.25	1
Esqui	1.00	0.99	1.00	165
Futebol	1.00	1.00	1.00	3615
Futebol americano	0.94	1.00	0.97	16
Futsal	1.00	1.00	1.00	192
Golfe	1.00	0.98	0.99	203
Hipismo	0.80	1.00	0.89	4
Hóquei	1.00	1.00	1.00	138
Jogos olímpicos	1.00	0.94	0.97	16
Motorizado	1.00	0.99	0.99	947
Radical	0.96	1.00	0.98	23
Rugby	1.00	0.98	0.99	61
Skateboard	1.00	1.00	1.00	24
Snooker	1.00	1.00	1.00	97
Snowboard	0.91	1.00	0.96	32
Surf	0.98	0.97	0.98	63
Triatlo	0.78	1.00	0.88	7
Ténis	1.00	1.00	1.00	185
Vela	1.00	1.00	1.00	4
Voleibol	0.99	1.00	0.99	70
micro avg	1.00	1.00	1.00	7098
macro avg	0.94	0.99	0.95	7098
weighted avg	1.00	1.00	1.00	7098
samples avg	1.00	1.00	1.00	7098

Figure 5.11: Classification report with over-sampling

5.4.7 Discarding the labels with the least impact for training

One of the issues of the dataset we are working with, as shown in table 5.1, is that beyond being unbalanced, some labels are practically non-existent in the training dataset (or non-existent at all), like "Desportos inverno" or "Dança". Consequently, a potential solution that was implemented was to create a parameter that allows the user to set a value so that labels that have a lower number of samples are removed from the training dataset. For example, if we set the value as 10, the programs with the labels "Dança" and "Desportos inverno" would not be considered for training purposes.

By observation of the table 5.14, we can remark that this was an interesting approach and it does provide better results if the right value is chosen, which in this case was 10. This was implemented in the final classifier once it has a good impact in prioritizing a good classification in the labels with the most share, to the detriment of the less impactful ones.

5.5 Validation and outputs

When it comes to classifying programs on a large scale, even though the F1-Score is a reliable metric, it is also important to test the classifier in a real scenario and observe the outputs. Since the classifier is going to be applied to all the sports programs in the

Minimum samples	**F1-score**
0	0.9887 ± 0.0013
5	0.9882 ± 0.0012
10	**0.9890 ± 0.0011**
25	0.9889 ± 0.0009
50	0.9885 ± 0.0010
100	0.9887 ± 0.0010
250	0.9885 ± 0.0012
500	0.9882 ± 0.0013
1000	0.9887 ± 0.0014

Table 5.14: Results obtained for different values of minimum samples

MEO database, and the results will influence what is going to be presented to all the customers, our priorities were to make a classifier that runs fast (because there are a lot of programs that need to be classified), optimize as much as we can so that there are close to no classification mistakes, and when there are, we want them to be in programs with fewer viewers, so it does not have that much impact.

For that, given the test results presented in section 5.4, we have chosen TF-IDF as our feature extraction technique since the results were good, it is simple, runs fast, and the results are also more easily interpretable. We have also chosen to not perform over-sampling because by observing the results individually, we noticed that it would make the classifier miss more classifications in the most viewed labels, like "Futebol" and "Basquetebol", to the detriment of correctly classifying in labels that do not have as high as an impact like "Dança" and "Vela".

Also it was important for the output to be as clear and informative as possible. Accordingly, on the output file that results from running our classifier, there are the following columns for each program:

- **CODOP_PROG_TV** - unique identifier of the program
- **NAME** - title of the program
- **SYNOPSIS** - synopsis of the program
- **TARGET** - target label, if defined
- **TFIDF_PREDICTIONS** - most likely prediction
- **TFIDF_PROBABILITIES** - probability of the most likely prediction
- **TFIDF_SECOND_PREDICTION** - second most likely prediction
- **TFIDF_SECOND_PROBABILITY** - probability of the second most likely prediction
- **ASSOCIATION** - label obtained by key-word association, if available
- **TFIDF_FINAL_CLASSIFICATION** - associated label if available, most likely prediction otherwise
- **CLUBS** - clubs found in the program (among Benfica, Porto and Sporting)
- **FLAG_BENFICA** - 1 if Benfica is found in the program, 0 otherwise

- **FLAG_PORTO** - 1 if Porto is found in the program, 0 otherwise
- **FLAG_SPORTING** - 1 if Sporting is found in the program, 0 otherwise
- **TFIDF_PREDICTION_EQUALS_ASSOCIATION** - 1 if most likely prediction equals the associated label, 0 otherwise
- **TFIDF_FINAL_EQUALS_ASSOCIATION** - 1 if the value in "TFIDF_FINAL_CLASSIFICATIC equals associated label, 0 otherwise
- **HAS_TARGET** - 1 if there is a target defined, 0 otherwise
- **TFIDF_FINAL_EQUALS_TARGET** - 1 if the value in "TFIDF_FINAL_CLASSIFICATION" equals target label, 0 otherwise
- **PROBA_BIGGER_THAN_THRESHOLD** - 1 if probability is higher than the threshold defined for a program to be classified by the classifier, 0 otherwise

Apart from the classification itself, which was the main objective of the whole project, we can also get some extra information like the probability of the prediction, which can be useful if we want the predicted label to be attributed only if the probability is above a certain threshold, that is defined as a parameter, to ensure that the classifications are correct, at the cost of not having a classification for all the programs. We decided that it was also important to include the second prediction and its probability to check, in wrong classifications, if the second classification would be the correct one. Also, the "TFIDF_FINAL_CLASSIFICATION" column was included to offer an alternative in the classification, where the program is classified by association and the classifier only predicts the label when there is no association obtained. The club detection is also very important for recommending potential business opportunities and also have a more complete picture of every program, since, in Portugal, football is the sport with the most fans and the clubs considered represent the vast majority of the fans.

Given the importance that the key-word association has to this project, during the various validation phases that took place, several improvements were made to the association mechanism. For example, since there were synopsis that included the name of more than one sport, we decided to submit the title first for searching for key-words, and only if there were no matches we would search on the synopsis as well. Also, first, the key-words considered are the name of the sport itself and only then the key-words associated with each sport are searched. Another improvement was to make a custom python function to search the key-words since words like "revelação" in "liga revelação" (the under-23 football championship in Portugal) were being classified as "Vela" because it was included in the word, even though the full word did not match and all the built-in functions tested had some flaws, so a regex-based approach had to be implemented for the key-word association to run flawlessly.

By observing the two example outputs shown in table 5.15, we can have a better understanding of how the output information works. Some interesting observations are that both classifications are correct and, for example 1, the second prediction would be "Motorizado", which is interesting since Carlos Sainz, whose name is present in the title of the program, is a Formula 1 driver. In the second example, "Basquetebol" was the second most likely prediction, but the probability was very close to zero. Also, in the second example, we can see how the club classification works and how it is important to have it separated from the program title or synopsis.

	Example 1	Example 2
CODOP_PROG_TV	14971308	15058052
NAME	Copa 90: Paulo Dybala x Carlos Sainz	Benfica x Sporting - Liga Revelação
SYNOPSIS	Copa 90: Paulo Dybala x Carlos Sainz Série que procura o melhor jogador do FIFA, entre personalidades do mundo do desporto desde o futebol até à Fórmula 1. Rio Ferdinand, ex-internacional inglês é o principal apresentador.	Benfica x Sporting - Liga Revalação Transmissão do jogo a contar para a Liga Revelação.
TARGET	Indefinido	Indefinido
TFIDF_PREDICTIONS	Futebol	Futebol
TFIDF_PROBABILITIES	0.918974043	0.966959979
TFIDF_SECOND_PREDICTION	Motorizado	Basquetebol
TFIDF_SECOND_PROBABILITY	0.032311324	0.009555176
ASSOCIATION	Futebol	Futebol
TFIDF_FINAL_CLASSIFICATION	Futebol	Futebol
CLUBS		Benfica,Sporting
FLAG_BENFICA	0	1
FLAG_PORTO	0	0
FLAG_SPORTING	0	1
TFIDF_PREDICTION_EQUALS_ASSOCIATION	1	1
TFIDF_FINAL_EQUALS_ASSOCIATION	1	1
HAS_TARGET	0	0
TFIDF_FINAL_EQUALS_TARGET	0	0
PROBA_BIGGER_THAN_THRESHOLD	1	1

Table 5.15: Example output for two programs

The flag in the column "TFIDF_PREDICTION_EQUALS_ASSOCIATION" was really important so that we could easily calculate the number/percentage of predictions made by the classifier that matched the associations made, which represents the accuracy of the classifier. When it comes to accuracy, we started with values around 70% and after all the improvements were made, in the final version we achieved 97.60% accuracy in the 44963 programs of the validation dataset that could be labeled by association.

As we can observe in figure 5.12, we were now at a phase where only the incorporation in the MEO ecosystem was left for us to achieve the wanted final product, which was a sports classifier working directly with the database entries.

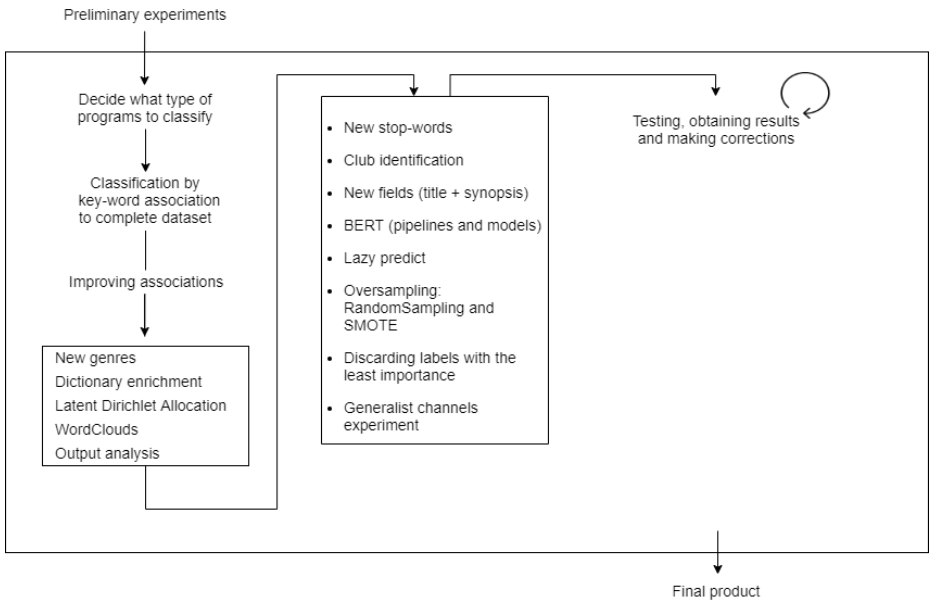

Figure 5.12: Testing phase position on the course of action

5.6 Revisiting the movies dataset

After all the improvements made to what was presented in the first iteration, even though it was not in the same context, we thought it would be interesting to apply some of the main changes to the movies classifier, mainly BERT, and test to see if the results would improve in that context as well.

Despite our expectations of obtaining better results by using BERT for feature extraction, since in the movies dataset the text is more linear, instead of being so organized by topics, like in the sports dataset, the results remained very close to what had been obtained before. Even though theoretically, BERT performs better in more linear text, given that the model can have access to the context of the words, it produces features in a way that the probabilities obtained by the classifier are closer to the others. Given that, with the movies dataset, we are working with a multi-label problem, this is far from ideal since it becomes more difficult to define a threshold capable of efficiently dividing the correct labels from the incorrect ones, even after performing tests with every threshold with a 1% radius.

Consequently, one of the possible solutions that could be performed would be to fine-tune BERT to perform the classification of these programs. Even though we still managed to implement a fine-tuning approach for BERT in another dataset, it was decided, together with MEO, that it would not be worth applying it to the movies dataset since it was not a priority and its impact in this area would not justify the computational resources needed since it would not be possible to do a proper fine-tuning on the laptop used so far and the priority of classifying the movies was, at the moment, not as high as performing other tasks in the sports classifier, like preparing the integration process.

This page is intentionally left blank.

Chapter 6

Integration in the MEO systems

After all the required elements were properly implemented and tested, it was time to start the phase that really gives value to this project, which is its integration into MEO's systems, in order to classify the programs directly into the database.

Thus, during the final stages of the project, there was the need to clean and organize the code (for example, reading the data had to be in a separate function so that it could be easily changed for reading the database directly) and prepare packages with the code, a requirements file with the python libraries needed and their versions and a readME file explaining how to run the program and the expected inputs, outputs and how to change the parameters. In every delivery, the scripts were conveniently commented and organized so that the integration process could be easier.

To enable a process that could work both in the present and the future, we decided to first test the classifier packages, to check if the classifier was doing its function as expected and make the changes needed in time. These packages had the code and information needed to run an already trained classifier. Then, we started working on the training package, which makes available the possibility to train a classifier using new data, since the programs and competition names may change anytime and the classifier would not be able to deal with them in the best of ways.

Also, after every version was delivered there was a meeting where the MEO team was assisted and every bit of code was explained so that they could understand what was really going on and also for helping to suggest improvements that could be inserted in the upcoming versions. Moreover, in between the meetings, there have been updates by message about some smaller corrections or doubts that might exist.

Tables 6.1 and 6.2 illustrate the main functions of each package, their expected input, output and description, for a better understanding of the whole process and every function. It is important to note that the functions *set_stop_words*, *infer_tags*, *stem_text*, *remove_stopwords*, *clean_text_detailed* and *string_found* are also used in the training package, but are not represented in that table to avoid duplicated information. These tables were also delivered to the MEO team in charge for helping with code maintenance, or if some change is to be made by someone else.

When it comes to package versions, there were four different versions delivered for the classifier package, with some minor changes provided in sub-versions. Some of the main changes made from the first version to the fourth were the addition of Bidirectional Encoder Representations from Transformers (BERT) for feature extraction, club classification, several flags on the output file, changes on the key-word dictionary and improvements to the

Function name	Input	Output	Description
string_found	string1, string2	True/False	Receives 2 strings and checks if the first is included in the second, using regex
infer_tags	vectorizer, classifier, multilabel binarizer, synopse, title, stop-words	prediction, probability, association	Performs the whole classification process for 1 synopsis at a time, returning the information associated with the classification obtained
set_stop_words	—	stop-words set	Returns a set of stop-words composed by the english and Portuguese sets of the NLTK library, as well as a text file with words added by hand
stem_text	list of words	stemmed list of words	Performs the stemming on a received list of words, using the RSLPStemmer (Removedor de Sufixos da Língua Portuguesa)
remove_stopwords	text, stop-words	text without stop-words	Checks if every word on the given text is in the stop-words set and removes them if they are
clean_text_detailed	text	list of words	Cleans the text by putting everything in lowercase, tokenizing, removing punctuation and replacing every number by zeroes (for example: 2021 ->0000)
vectorize	list of words for feature extraction, vectorizer	features	Uses the vectorizer to obtain features from the input text
get_clube	synopse	club, benfica_flag, porto_flag, sporting_flag	Checks for the presence of the clubs considered, with respect to the dictionary of combinations of words to not be considered (for example: Leões de Porto Salvo is not related to FC Porto)
unite	list of words	text	Simply joins the input words to form a sentence
read_data	—	data	Reads the data from an excel file into a pandas DataFrame
main	data, classifier, vectorizer, genres expected	—	Takes care of the whole process, starts by joining the title and synopsis if required by the parameters, binarizes the genres expected, gets the stop-words set and performs the whole classification process, by calling the infer_tags function for each of the synopses and organizing the outputs for them being displayed in the output excel file, with all the information presented before in this chapter
if __name__ == '__main__'	—	—	Reads the data, imports the previously trained classifier and vectorizer, as well as the expected genres, and calls the main function

Table 6.1: Information about the main functions on the classifier package

Function name	Input	Output	Description
directclassify	data	classified data	Classifies that unlabeled data as it is described in figure 5.5
filter_above_threshold	classified data	classified data with number of samples aboce threshold	Checks if every label has a number of samples above the threshold defined in the parameters. If not, then the programs with the labels in question are removed from the training and testing datasets
process_genres	genres	processed genres	Gets the genres obtained directly from the dataset and produces a list containing them (in this case it is always 1 genre only), for example: "Futebol" ->["Futebol"]
printmostrelevant	vectorizer, input texts	—	Generates an excel file with the ranking of the words that have the most total TF-IDF value
getPipeline	—	pipeline	Imports the BERT pipeline to be used for feature extraction
getFeatures	plot, pipeline	features	Gets the BERT features applying the pipeline created before to individual plots
trainClassifier	train features, test features, train outputs	classifier, predictions	Creates and trains the classifier with the given features and outputs for training, and then makes predictions using the testing dataset
main	key-word classified data	F1-Score	Takes care of the whole training process, divides the dataset, applies the preprocessing methods, extracts the features, performs over-sampling if needed (defined in the parameters and trains the classifier. Also obtains a classification report after the testing is done and the F1-Score. At the end, the classifier and the vectorizer (if TF-IDF is used) are saved
if __name__ == '__main__'	—	—	Checks if the dictionary has any repetitions, reads the data, classifies the unlabeled data based on key-words, filters the dataset to eliminate the labels with least occurrence if needed and calls the main function, printing the results

Table 6.2: Information about the main functions exclusive to the training package

code, on the correction of minor mistakes, adding verifications to check if all the requirements are valid and improving performance. The first version of the classifier package was delivered on April 26th and had been improved since then.

The training package had two versions associated, mainly distinguished by the addition

of oversampling and the possibility of having a threshold to filter training inputs on the second version.

6.1 Parameters

Along with the script files, there was, in each package, a parameters file where several variables could be changed more easily for each run, without having to change the main code files.

For the classifier package, the main parameters were:

- **FILE_NAME** - name of the file with the programs to be classified
- **STOP_WORDS** - if stop-word removal is to be performed (True/False)
- **STEMMING** - if stemming is to be performed (True/False)
- **INCLUDE_TITLE** - if we want to use title, synopsis, or both (0 - synopsis, 1 - both, 2 - title)
- **PROBA_THRESH** - minimum probability value for a program to be classified

These parameters are well commented so that the person running the scripts can change the configuration of the program based on their intentions. It is important to note that some parameters like *STOP_WORDS*, *STEMMING*, among others, should be the same for the classifier and for the training package to provide the classifier with the conditions in which it was trained and tested.

Moreover, for the training package, there were more variables to allow the training to be as adaptive as possible to the needs of the system the classifier is going to be applied to. Those parameters are:

- **FILE_NAME** - name of the file with the programs to be classified
- **STOP_WORDS** - if stop-word removal is to be performed (True/False)
- **STEMMING** - if stemming is to be performed (True/False)
- **INCLUDE_TITLE** - if we want to use title, synopsis, or both (0 - synopsis, 1 - both, 2 - title)
- **MAX_DF** - maximum percentage of documents in which a word can be for it to be considered as a feature
- **MAX_FEATURES** - maximum number of features to be considered by Term Frequency - Inverse Document Frequency (TF-IDF)
- **CLASSIFIER** - classifier to be trained, can be logistic regression, K-Nearest Neighbors (KNN) classifier or Support Vector Machine (SVM)
- **EMBEDDING** - feature extraction method, can be TF-IDF or BERT
- **NR_RUNS** - number of runs, is useful to obtain a representative mean and standard deviation for the F1-Score

- **N_GRAMS** - the minimum and maximum number of words to be considered together as a feature
- **ASSOCIATIONS** - to consider the key-words dictionary or the label names only as key-words (True/False)
- **SAVE_CLASSIFIER** - if we want to save the trained classifier, vectorizer, and list of expected genres (True/False)
- **SAMPLING** - over-sampling technique to be used, can be "NONE", "RANDOM" for RandomSampling, and "SMOTE"
- **EXCLUDE_MIN_SAMPLES** - if we want to exclude, for training and testing, the programs whose label doesn't appear more than a certain threshold (True/False)
- **MINIMUM_SAMPLES** - threshold of minimum samples to be considered if the upper parameter is activated

It is important to refer that all the training parameters with a direct impact on the testing results were tested and all the results are presented in the previous chapter.

6.2 Integration impact and results

Due to the fact that the final product has already been integrated into the MEO ecosystem, to work directly over the database entries, it is possible to visualize its direct impact in some entries.

Table 6.3 shows 3 examples of programs where both the *TFIDF_FINAL_CLASSIFICATION* and *CLUBS* values were obtained directly from our classifier.

TFIDF_FINAL_CLASSIFICATION	CLUBS	PROGRAM_NAME	SYNOPSIS
Futebol	Benfica	Benfica x PSV - Liga Dos Campeões (Direto)	Assista ao jogo da UEFA Champions League entre Benfica x PSV
Futebol	Benfica	Spartak Moscovo x Benfica - Liga dos Campeões (Direto)	Transmissão do encontro a contar para a Liga dos Campeões
Futebol	Benfica	Benfica x Arouca - Primeira Liga	Transmissão do jogo da Primeira Liga

Table 6.3: Examples of programs in the database after applying the classifier

As referred to before, this automatic classification of programs is useful and directly related to other areas like improvement of Recommendation Systems and gathering business opportunities by obtaining a more accurate profile of individual customers.

In figure 6.1, we can observe, in the upper table, that 33% of the clients watched football programs that were until then considered undefined. At this point, not all programs were undefined since, during the development of this project, a more simple key-word based approach was applied for directly classifying some of the entries. This information gain allows providing better recommendations for those customers and better their experience as clients.

In the lower part of the figure, it shows that from all the customers that watched programs classified as football, 44% watched programs related to the club SL Benfica. These details allow, for example, for the company to explore the possibility of selling the services of "Benfica TV", which is Benfica's dedicated channel, to the percentage of those clients that do not already subscribe to it. These examples also work in a similar fashion for the other two biggest clubs in Portugal, which are FC Porto and Sporting CP, but

since "Benfica TV" is the only channel whose services are sold separately, we thought those would be the best results to present in this report.

GENERO_CANAL	TFIDF_FINAL_CLASSIFICATION	GENERO_PROG_FUTEBOL	GENERO_PROG_INDEFINIDO	PERC_CLI
Desporto	Futebol	0	1	33%
Desporto	Futebol	1	0	3%
Desporto	Futebol	1	1	64%

GENERO_CANAL	TFIDF_FINAL_CLASSIFICATION	CLUBS	PERC_CLI
Desporto	Futebol	Benfica	44%

Figure 6.1: Direct impact of the product when applied directly in the databases

Finally, after the integration phase was completed, we had our desired final product. Figure 6.2 shows the full course of action taken for developing this classifier with an adequate level of detail, making it easier to understand the whole process.

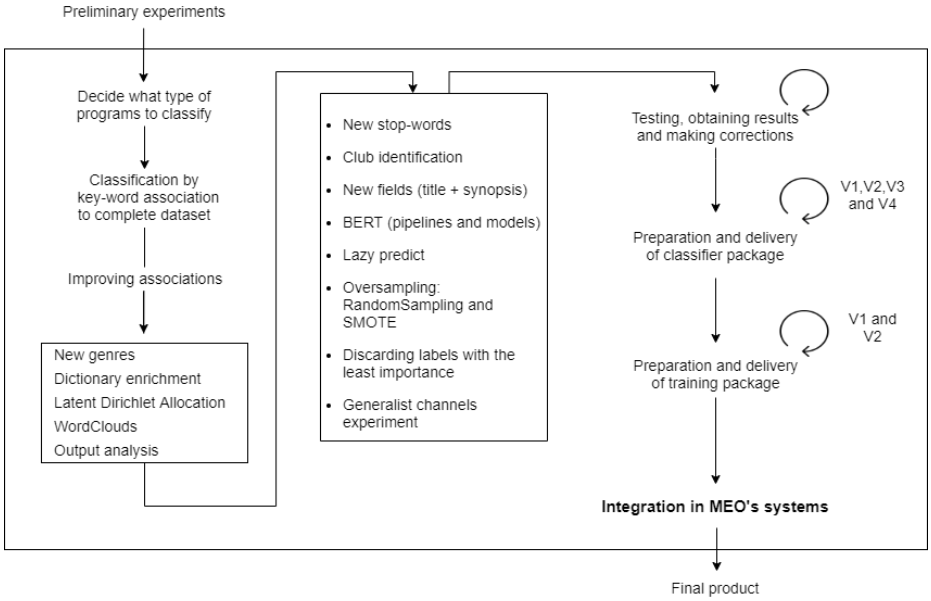

Figure 6.2: Integration phase position on the course of action

This page is intentionally left blank.

Chapter 7

Conclusion

In the scope of this internship, we have developed and integrated a model capable of automatically classifying sports programs taking into account the available information about them, while also obtaining and providing insightful metrics to support the decision process.

In this document, we started by contextualizing the theoretical concepts that are related to the project and overviewing some of the related experiments and their methodology. Later, the work performed in the first iteration is presented in detail. Afterward, the methodology used to approach the problem in the second iteration is tackled, from the decision-making process to the validation. Finally, an overview of the integration process is presented.

We were able to achieve significant gains of information in the MEO programs database, with regard to the recommendation systems, with the classification of previously unlabeled programs, and the user profiling, with both the classification of unlabeled programs and the identification of sports clubs in a synopsis. In particular, we were able to identify 33% more clients that watched football ("Futebol", in Portuguese), since its genre was unidentified at the start, and was then obtained through our classifier. Also, from all the clients that watched programs classified as "Futebol", we can now understand that 44% of them watched programs directly related to the club SL Benfica. Along with the classifier's development, some relevant metrics and experiments were also provided and every decision was made with the approval of the MEO team's members. In the end, the gains obtained allow the company to attain a better individual engagement and unravel possible business opportunities, and the clients to have a better user experience.

The investigation performed in the context of this Masters's Thesis also gave me important knowledge and experience in the areas of Natural Language Processing (NLP) and text classification.

For the future, relative to the sports classifier, for making it easier to add new keywords or retraining the model on the side of the MEO team, there have been various meetings to explain the code and deeply how the final product works. Also, there are still some aspects that remain to be tested, like for example to fine-tune Bidirectional Encoder Representations from Transformers (BERT) for this specific task or to test other, more complex, classifiers. When it comes to other types of programs, code was made available for covering different types of classification like multi-label classification on labeled data developed in the scope of the movies experiment, and also for single-label classification on labeled data developed in the experiments made for the sports classifier, on samples

labeled by hand. Even though these examples did not yet produce the necessary results to be integrated into the ecosystem, they are a solid basis from which more complex and robust classifiers can be created.

References

Amin, M. Z. and Nadeem, N. (2019). Convolutional neural network: Text classification model for open domain question answering system.

An, J. and Chen, Y.-P. P. (2005). Keyword extraction for text categorization. volume 2005, pages 556 – 561.

Blei, D., Ng, A., and Jordan, M. (2001). Latent dirichlet allocation. volume 3, pages 601–608.

Boser, B., Guyon, I., and Vapnik, V. (1996). A training algorithm for optimal margin classifier. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 5.

Breiman, L. (1997). Arcing the edge. Technical report.

Bučar, J. and Povh, J. (2013). A knn based algorithm for text categorization. *Proceedings of the 12th International Symposium on Operational Research in Slovenia, SOR 2013*, pages 367–372.

Cavnar, W. and Trenkle, J. (2001). N-gram-based text categorization. *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Chi, S., Qiu, X., Xu, Y., and Huang, X. (2019). *How to Fine-Tune BERT for Text Classification?*, pages 194–206.

Christanti, V., Rudy, R., and Naga, D. S. (2018). Fast and accurate spelling correction using trie and damerau-levenshtein distance bigram. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 16(2):827.

Dalal, M. and Zaveri, M. (2011). Automatic text classification: A technical review. *International Journal of Computer Applications*, 28.

Dan, M. and Severin, B. (2005). Classification process in a text document recommender system. *Annals of Dunarea de Jos*, 2005.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Du, R., Safavi-Naini, R., and Susilo, W. (2003). Web filtering using text classification. pages 325 – 330.

Dwivedi, S. K. and Arya, C. (2016). Automatic text classification in information retrieval. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies - ICTCS '16*. ACM Press.

Ertugrul, A. M. and Karagoz, P. (2018). Movie genre classification from plot summaries using bidirectional lstm.

Ferilli, S., Carolis, B., Esposito, F., and Redavid, D. (2015). Sentiment analysis as a text categorization task: A study on feature and algorithm selection for italian language.

Frank, E. and Bouckaert, R. (2006). Naive bayes for text classification with unbalanced classes.

Friedman, J. (2002). Stochastic gradient boosting. *Computational Statistics Data Analysis*, 38:367–378.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Glazkova, A. (2020). A comparison of synthetic oversampling methods for multi-class text classification.

González-Carvajal, S. and Garrido-Merchán, E. C. (2020). Comparing bert against traditional machine learning text classification.

Guarino, N., Oberle, D., and Staab, S. (2009). *What Is an Ontology?*, pages 1–17.

Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. (2006). Using knn model for automatic text categorization. *Soft Computing*, 10.

Gupta, G. and Malhotra, S. (2015). Article: Text document tokenization for word frequency count using rapid miner. *IJCA Proceedings on International Conference on Advancements in Engineering and Technology*, ICAET 2015(12):24–26. Full text available.

Hand, D. (1998). Data mining: Statistics and more? *The American Statistician*, 52:112–118.

Ho, K. W. (2011). Movies' genres classification by synopsis.

Hoang, Q. (2018). Predicting movie genres based on plot summaries.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Iosifova, O., Iosifov, I., Rolik, O., and Sokolov, V. (2020). Techniques comparison for natural language processing.

Joshi, P. (2019). Predicting Movie Genres using NLP – An Awesome Introduction to Multi-Label Classification. `https://www.analyticsvidhya.com/blog/2019/04/predicting-movie-genres-nlp-multi-label-classification/`. [Online; accessed 21-September-2020].

Joyce, J. (2019). Bayes' Theorem. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2019 edition.

Kamruzzaman, S. M., Haider, F., and Hasan, A. R. (2010). Text classification using data mining. *CoRR*, abs/1009.4987.

Karamizadeh, S., Abdullah, S. M., Halimi, M., Shayan, J., and j. Rajabi, M. (2014). Advantage and drawback of support vector machine functionality. In *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, pages 63–65.

Lai, S., Xu, L., Liu, K., and Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2267–2273. AAAI Press.

Li, B., Yu, S., and Lu, Q. (2003). An improved k-nearest neighbor algorithm for text categorization.

Liu, Z., Wang, C., and Wang, F. (2019). An improved knn text classification method. *International Journal of Computational Science and Engineering*, 20:397.

Lorensuhewa, A., Pham, B., and Geva, S. (2002). Keyword-based text matching approach for design style recognition.

Mangolin, R., Miranda Pereira, R., Jr, A., Silla, J., Feltrim, V. D., Bertolini, D., and Costa, Y. (2020). A multimodal approach for multi-label movie genre classification.

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

Narkhede, S. (2018). Understanding Confusion Matrix. `https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62`. [Online; accessed 10-January-2021].

Oliveira, H. G. (2014). The creation of onto.PT: A wordnet-like lexical ontology for portuguese. In *Lecture Notes in Computer Science*, pages 161–169. Springer International Publishing.

Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference.

Peng, J., Lee, K., and Ingersoll, G. (2002). An introduction to logistic regression analysis and reporting. *Journal of Educational Research - J EDUC RES*, 96:3–14.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. volume 14, pages 1532–1543.

Rambola, R., Varshney, P., and Vishwakarma, P. (2018). Data mining techniques for fraud detection in banking sector. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, pages 1–5.

Rennie, J., Shih, L., Teevan, J., and Karger, D. (2003). Tackling the poor assumptions of naive bayes text classifiers. *Proceedings of the Twentieth International Conference on Machine Learning*, 41.

Saif, H., Fernández, M., He, Y., and Alani, H. (2014). On stopwords, filtering and data sparsity for sentiment analysis of twitter. In *LREC 2014, Ninth International Conference on Language Resources and Evaluation. Proceedings.*, pages 810–817.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2018). Bidirectional attention flow for machine comprehension.

Shang, W., Huang, H., Zhu, H., Lin, Y., Qu, Y., and Dong, H. (2006). An adaptive fuzzy knn text classifier. volume 3993, pages 216–223.

Silva, M. J., Carvalho, P., and Sarmento, L. (2012). Building a sentiment lexicon for social judgement mining. In *Lecture Notes in Computer Science*, pages 218–228. Springer Berlin Heidelberg.

Souza, F., Nogueira, R., and Lotufo, R. (2020). BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Tokunaga, T. and Iwayama, M. (1994). Text categorization based on weighted inverse document frequency.

Ur-Rahman, N. and Harding, J. (2012a). Textual data mining for industrial knowledge management and text classification: A business oriented approach. *Expert Syst. Appl.*, 39:4729–4739.

Ur-Rahman, N. and Harding, J. (2012b). Textual data mining for industrial knowledge management and text classification: A business oriented approach. *Expert Syst. Appl.*, 39:4729–4739.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Zhang, H. and Li, D. (2007). Naïve bayes text classifier. pages 708 – 708.