B. ANEXO: ENGLISH SUMMARY

Introduction

Given the enormous quantity of data being generated by the industry and society every day on the Internet, data-driven analysis has been a rising trend in the last two decades. From *business intelligence*, to *hate-speech* or *fake news* detection, Machine Learning techniques are a key tool necessary for processing and obtaining relevant insights from these continuously growing amounts of data.

First of all, as documents originate from the human language, and machines are not really capable of processing directly speech; *Natural Language Processing (NLP)* field aids computers to extract valuable attributes and represent human language in such a way that analysis can be carried on them. Classifying documents on their topical categories is a relevant task achieved through the combination of *Machine Learning* and *NLP*; however, in order for these techniques to work as expected, a generous amount of labeled documents from each category is usually needed.

Nonetheless, gathering labeled data for document classification is often hard or even impossible for many researchers or businesses. For this reason, the main goal of this work is to explore the use of category definitions as training data for the document classification task. These definitions would be automatically extracted from their *Wikipedia* page in such a way an autonomous and dynamic system for training data extraction precedes the classification task.

In order to follow this main motivation, a set of milestones have been defined, in such a way that a complete end to end classification system is attained.

First, two *web crawlers* will be designed and implemented in order to gather training and evaluation data from both *Wikipedia* (definitions and articles) and *arXiv* (scientific articles) online repositories. Then, the obtained documents will be processed so that are represented in a clear and formatted structure suitable for the classifier's labor. Afterwards, a battery of experiments based on hyperparameter search and model evaluation will be performed for a set of implemented models for the task of document classification based on topic definitions. Finally, the results obtained in these tests will be compared to those executed in two different but rather common situations: training over a subset of articles per class, and training over a randomly chosen article per class.

Bearing these objectives in mind, this project thesis has been structured as follows:

- **State of the art:** this chapter will serve as a contextualization and theoretical basis for this work, founded on a scientific literature search on the algorithms and text processing techniques to be applied.
- **Data gathering and processing:** narrates the procedure for obtaining the datasets by means of the implemented web crawlers as well as the text preprocessing and vectorization techniques applied on the extracted documents.
- **Design and implementation of classifier models**: describes the design and development criteria for the classification models considered as well as the final structure of the dataset which will be fed to them.
- **Experimentation on different document classification scenarios**: exploring the performance of different models and parameters when classifying documents based on: i) topic definitions, ii) subset of articles, iii) single article per class.
- **Conclusions and future work**: Based on the experiments that had been run, the main key points will be drawn from the experimentation and the goals initially established; ending with a list of future lines of work to be taken into account for continuing this project.

State of the art

First, the document classification task can be stated in the following way: given a set D made up by documents d_i, $i \in \{0, 1, ..., D-1\}$ belonging to a category or topic $t \in \{0, 1, ..., n_{\text{topics}}\}$ so that n_{topics} is the total number of unique classes in the set; it is desired to classify an unlabeled document d_j by a model estimating the category which maximizes $p(t|d_j, D)$. It should be noted that the set of words present in all documents is grouped in a vocabulary of dimension V unique words; having each document a variable extension in number of words.

Besides the general problem of document classification aimed to be solved, this work is characterised by some important aspects. On the one hand, the supervised learning approach in which a set of labeled documents is used to train the models for later inference on unseen samples; with the special case of *one-shot* learning [14], where there is a single labeled document for each category during training (the definitions of each class). On the other hand, the exploration of different document vectorization techniques, in such a way that the main characteristics can be abstracted from them and represented in a lower dimensional space suitable for comparison.

The fact that definitions are used as a training set instead of articles in the supervised learning fashion where the training set belongs to the same statistical distribution as the test data; enhances a potential risk as definitions are not strictly a document more from the categories which they define, but rather a document which is supposedly broader and contains the general aspects describing a given topic. This may or may not enhance the classification performance of the models, and thus has been worth considered exploring.

For these reasons, this work has made a literacy review on both the usage of definitions of categories as training sets as well as document representation techniques. It has been shown previously in literature the use of *Wikipedia* as a support of the supervised learning task, either by directly using the articles for the training sets and generating *graphs* [8] which model the category ontologies; or by pre-tuning the kernel of a *Support Vector Machine* [7] used as classification model, among other approaches [9] [10].

Regarding the document representation techniques, in order for the classifiers not to suffer from the *"Curse of dimensionality"* [24] problem that arises when analyzing high dimensional data (i.e with a high number of attributes), different vectorizations have been explored aiming to represent the documents in a lower-dimensional space which ideally is capable of keeping the semantic relationships of the words and sentences in this new and denser space, suitable for computational models.

The first representational form explored was the *Bag of Words (BoW)* model, which simply represents the corpus of a document as a dictionary with unique words as keys and their frequency (*term frequency, tf*) in the document as values. Moreover, the *term frequency - inverse document frequency (tf-idf)* model improves *BoW* by also taking into account the *idf* via a logarithmic transformation, giving a weight for each word depending on their discriminative power: high for words appearing frequently in a set of common documents, and low for those commonly appearing in all of the documents or appearing once or twice in the whole corpus [19].

However, these two models neither take into account word ordering nor considers semantic relationships between words, as all of them are separated by the same distance in their new dimensional space. As a consequence, this work also explored the use of *word embeddings* given the performance boosts they have shown on many scientific publications [20][21]. These *embeddings* are word vectors obtained through the so called *Neural Network Language Models (NNLM)* [22] or also by applying dimensionality reduction techniques to word co-occurrence matrices [23], among other methods.

The most prominent implementation of a *NNLM* for obtaining *word embeddings* was the *word2vec* model published by *Mikolov et. al* [25]. This model uses a shallow neural network consisting on a single hidden layer which architecture (structure of the layers and nodes) depends on the underlying algorithm selected, as they show two alternatives: *Continuous BoW*, and *Continuous Skip-gram*. As a result of either one of the chosen algorithms; a vectorial representation of the word vocabulary will be learnt so that the new dimensional space is able to capture semantic relationships between the words; i.e placing close words which meanings are similar, and even having dimensions representing e.g gender, or singular/plural noun relationships.

Briefly, the goal of these algorithms is to obtain my means of a learning algorithm a matrix of *embeddings* $\mathbf{W}$ from which columns the vectorial representation of words will be taken. In the *CBoW* architecture, training will be performed given a context of C words surrounding a central word, which won't be shown to the model and will try to estimate. This will be an iterative procedure involving sliding windows of contexts through the entire corpus until the model is able to correctly estimate the central word from its relevant context. On the other hand; the *Continuous Skip-gram* model does exactly the opposite, given a central word it will try to estimate the surrounding context words.

Moreover, given that the main objective of this project involves the representation of documents (as collection of words) prior to the classification task, another algorithm has been considered for its application on the *Maximum Similarity Classifier* distance-based inference also developed in this work. Introduced by *Q. Le and Mikolov* [32], *doc2vec* is based on the *word2vec* concept but instead aims to represent whole documents or paragraphs as dense vectors in a lower dimensional space capturing the same semantic attributes as those intended in the *word2vec* model.

In a similar design criteria that in *word2vec*, the *doc2vec* model has two different algorithms which can achieve the same goal, but in distinct approaches. The *PV-Distributed Memory (PV-DM)* scheme concatenates a paragraph or document to a sequence context words sampled from that document and will learn by predicting the center word through an iterative process a matrix **D**, from which the *document embeddings* can be extracted from. On the other hand, the *PV-Distributed BoW (PV-DBoW)* algorithm of *doc2vec* will learn the document matrix **D** by means of probabilistic sampling from an input document a list of context words.

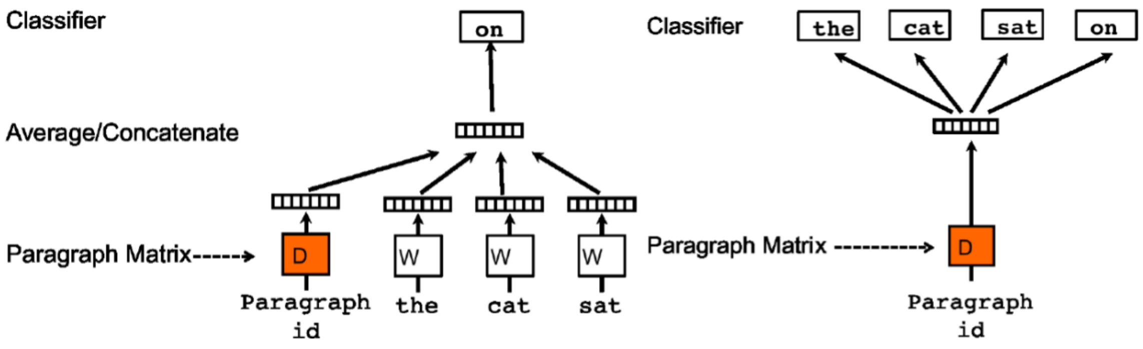

Fig. B.1. PV-DM (left) and PV-DBoW (right) architecture schemes. Figure from [32]

Data gathering and processing

In order to obtain datasets suited for the document classification task based on category definitions, two *web crawlers* have been implemented in this project alongside with text processing functions. The code written in *Python* mentioned in the following chapters can be found in functions and modules under the library present at the *GitHub* repository [41] of the project, freely accessible and open source.

First, for obtaining both the category definitions as well as articles belonging to that given list of categories, a module called `wiki-parser` has been developed using the `Wikipedia API` [33]. The goal of this module is to extract those documents from the web and process them so that a structure of topic definitions together with a set of articles belonging to each of the topics is obtained in clean text. Furthermore, a *multi-threading* version of the *crawler* was developed in order to concurrently extract all the category articles and reduce the overall time required for the library to gather the data.

Secondly, in order to have a secondary dataset more complex in nature and challenging for classifiers, a *web crawler* of the *arXiv* [34] scientific papers repository was developed. This `arxiv-parser` is able to perform through the `Requests` *Python* library a set of queries[18] to the *arXiv* search engine and process with the `BeautifulSoup` [40] framework the *html* pages resulting from the `GET` requests. As a result, a dataset consisting of several papers sorted by categories (i.e the main scientific fields of the *arXiv* repository) is obtained. Note that, for the task at hand, this project has considered the articles as their publication title together with their extended abstracts.

After designing and developing the data gathering tools; a set of dataset processing functions were implemented in the `doc_utils` module in order for the classifiers to use the extracted documents. Two types of functions were developed: the text "cleaning" functions, and the text vectorization functions. First, the text cleaning methods perform different levels of text processing: *tokenization*, where strings of text are split in lists of individual words; *lemmatization*, where the words are converted to their root form (i.e singular forms, present verb tenses...); punctuation signs removal, and finally *stop-words* removal. Note that it will be possible during model testing to select different granularity of these text preprocessing techniques.

Once the documents have been processed, the vectorization functions are in charge of restructuring the lists of processed tokens so that the models can fit the data as they would normally do. Each model has a different "prepare data" function, depending on each *framework* needs. In the end of this whole process, a structured *train* and *test* processed data splits is achieved, ready to be fed into the classification algorithms, as the figure B.2 depicts.

Design and implementation of classifier models

A set of models have been considered in order to test the relevance of topic definitions for document classification. These models have been divided in two groups; the baseline models, which use the classical vectorization procedures of *BoW* and *tf-idf*; and a se-

[18]Can also specify the amount of papers per category, range of publication years, relevance, etc.

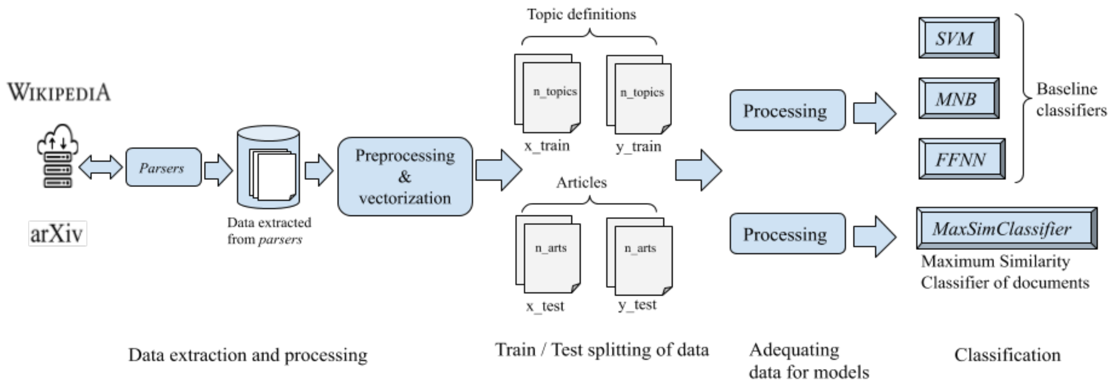
Wikipedia
arXiv
Parsers
Data extracted from parsers
Preprocessing & vectorization
Topic definitions
n_topics
n_topics
x_train
y_train
Articles
n_arts
n_arts
x_test
y_test
Processing
Processing
SVM
MNB
FFNN
Baseline classifiers
MaxSimClassifier
Maximum Similarity Classifier of documents
Data extraction and processing
Train / Test splitting of data
Adequating data for models
Classification

Fig. B.2. General schematic of the full process: from data extraction to document classification.

cond group consisting on a single classifier custom-designed for this project so that uses document *embeddings*, named *Maximum Similarity Classifier (MSC)*.

First; regarding to the baseline models, this work has explored the following:

- *Feed Forward Neural Networks (FFNN)*: using the *Keras* `Sequential` model declaration [44], a *FFNN* classifier scheme was designed. In the scheme, it was left to choose the model architecture; composed by the number of layers, number of nodes per layer, activation functions and optimizers for the learning algorithm.

- *Support Vector Machines (SVM)*: by means of the *Scikit-learn* library, a *SVM* classifier [51] was also considered, together with a *Radial Basis Function (RBF)* kernel transformation. An ensemble of $\frac{n_{topics}\cdot(n_{topics}-1)}{2}$ one-to-one binary SVM classifiers was applied to deal with this multi-class classification task. Different values for the model parameters C and γ were taken into account for the model testing.

- *Multinomial Naive Bayes (MNB)*: this classifier is named “naive” as it holds both the assumptions [57] of conditional independence between words of the corpus as well as their positional independence; thus making it a suitable classifier for the *BoW* and *tf-idf* text vectorizations. In the same approach as with the *SVM* classifier, the *Scikit-learn* implementation of the algorithm [52] was chosen as a starting point. The main tunable parameter of the model, α, was taken into account for testing.

Note that the reasoning of choosing the *Scikit-learn* implementation of *SVM* and *MNB* algorithms was driven by the ease of use of them together with the text preprocessing steps which can be introduced with ease in the `Pipeline` function of the models.

Finally, regarding to the usage of document *embeddings*:

- *Maximum Similarity Classifier (MSC)*: developed for this project, is a distance-based classifier which uses document *embeddings* learnt by the *doc2vec Gensim's* implementation [53]. The classifier was implemented following the design specifications of *Scikit-learn* modules, so that those who are familiar with their interfaces can quickly use this model. In that way, first the classifier can be initialized with a given set of hyperparameters such as its the vector size of the *embeddings* or the type of *doc2vec* algorithm to use, among others. Then, the `fit` method learns the document embeddings for the input dataset. Finally, with the `predict` or `score` methods the classifier can perform inference on the test data and compute the *cosine similarity* distance metric between the topic definitions learnt during the model training and the test articles fed into the model. Whereas the `predict` method outputs for each category the most similar articles ranked by its similarity; the `score` method returns either the *Top-1* or *Top-2* accuracy rates given the true labels of the test articles. A `pseudo-labelling` [15] method was also implemented in the classifier in order to explore its applicability on the project.

Experimentation on different document classification scenarios

After implementing the aforementioned classifiers, a set of tests were run inside a *Google Colaboratory* [38] environment. These experiments can be found as *Python .ipynb notebooks* in the `Notebooks` folder of the project repository. Following a search-and-evaluation approach, the tests were organized as follows:

1. Selecting the dataset and classifier to start the test with.
2. Defining the list of vectorization and preprocessing techniques to be applied.
3. Defining a list with ranges of model hyperparameters to consider for the classifier.
4. *Grid-search* of best hyperparameter and preprocessing/vectorization combinations according to the score metric.
5. Evaluation of the model's best combination of parameters with its Top-1 or Top-2 accuracy scores and *confusion matrix* plot to analyze results.

After executing the classification tests over the *FFNN*, *SVM*, *MNB* and *MSC* classifiers using topic definitions as training set; two other situations were also explored and tested:

- Classification based on training over a single, randomly chosen article per category; and testing over the rest of articles excluding that one.
- Classification based on training and testing over a subset of articles of each category, in a 25-75 split (25 % of the articles belonging as training data, and 75 % of the articles as test data).

The final results of the model's best combinations for each situation and datasets are shown in table B.1. Note that in order to account for the high variance resulting from the randomized extraction of the chosen article to train in the single-article situation; the results marked with (*) were averaged for a series of 30 experiment trials.

Dataset - Top-1 acc. rate	***FFNN***	***SVM***	***MNB***	***MSC***
Wikipedia - Clf. articles 25 %75 %	0.7310	0.7430	**0.7670**	0.6988
*Wikipedia - Clf. by article per cat.**	0.2013	**0.3381**	0.3308	0.3173
Wikipedia - Clf. by cat. definition	0.4050	0.7357	**0.7397**	0.6101
arXiv - Clf. articles 25 %75 %	0.7050	0.7101	**0.7300**	0.6950
*arXiv - Clf. by article per cat.**	0.1680	0.2476	**0.2538**	0.1957
arXiv - Clf. by cat. definition	0.2350	0.4188	**0.4287**	0.4212
Dataset - Top-2 acc. rate	***FFNN***	***SVM***	***MNB***	***MSC***
Wikipedia - Clf. articles 25 %75 %	0.8875	**0.8920**	0.8840	0.8435
*Wikipedia - Clf. by article per cat.**	0.3442	**0.4987**	0.4936	0.4777
Wikipedia - Clf. by cat. definition	0.6402	**0.8814**	0.8793	0.7668
arXiv - Clf. articles 25 %75 %	**0.8999**	0.8750	0.8700	0.8400
*arXiv - Clf. by article per cat.**	0.2814	0.4073	**0.4137**	0.3310
arXiv - Clf. by cat. definition	0.4099	0.6063	**0.6225**	0.6215

TABLA B.1. OVERALL RESULTS FOR DIFFERENT DOCUMENT CLASSIFICATION SCENARIOS.

Conclusions and future work

After observing the performance of the classifiers on the different situations and datasets; as well as the *grid-search* conducted for each, a set of conclusions were drawn.

First, the complexity of the different datasets have challenged all the classifiers during the tests, as a strong bias towards the conflicting categories was observed through the confusion matrices of their results. These conflicting categories where mainly those which perhaps were broader in the sense that could encompass others; e.g. *Statistics* category in the case of the *arXiv* scientific fields; or *Mechanical Engineering* in the case of the *Wikipedia* engineering topics dataset. It should be noted too the relevance of the category definition length in number of words; as the broader and lengthy the definition, the more prone to for the classifiers to generalize the prediction of articles towards it.

Second, the role of text preprocessing has proven crucial for the correct performance of the classifiers. During the *grid-search* procedures, all of the classifiers evaluated had better results when the complex preprocessing techniques were applied (*tokenization* + *lemmatization* + *stop-words* filtering). In the case of the *SVM* and *MNB*, *tf-idf* transformation obtained also better results than the *BoW* vectorization. On the other side, the use of document *embeddings* in the *MSC* classifier proved that for the datasets under analysis, the *PV-DBoW* algorithm performed better than the *PV-DM*, although both document *embedding* techniques fell behind the *SVM* and *MNB* classifiers, which in general obtained the best results on nearly all situations.

Finally, when comparing the performance on classification based on the three different scenarios under analysis, it was shown that classifying documents based solely on their category definitions as training data was more efficient and yielded better results than the classification using an article sampled from each category. However, it was not a surprise that the classical scenario of 25 %-75 % train/test splits performed better in all of the classifiers and dataset, given the fact that having more sample articles per class during training typically translates into better validation results.

Bearing these results in mind, the motivation and goal of this project has thus been satisfied by concluding that, in the absence of labeled documents belonging to the different categories for training; the use of category definitions as training data is a relevant choice when there is no possibility of training over a subset of articles.

On the whole; given the bounded extension of a bachelor thesis and the broad range of possibilities this work opened during its research and development, some lines of **future work** have been noticed as reasonable to be explored as an extension:

- Exploring other datasets with different nature; e.g. classification of narrative literature works based on their topic definitions.
- Performing experiments with other kind of document vectorization techniques; for example, *"word embedding clouds"* to represent the documents, or using the *Topic Movers Distance* [62] as distance-based metric.
- Adding *Topic Modelling* attributes extracted from *Latent Dirichlet Analysis (LDA)* technique to the datasets.
- Exploring sample strength from the definitions sets. This would be done in order to determine how many labeled samples would yield similar performance to the usage of definitions during classification.
- Applying class-balancing techniques in order to deal or mitigate the bias towards the conflicting classes observed during classification.
- Implementing a *Label Propagation* graph algorithm which would try to improve the failed *pseudo-labelling* approach which did not yield significant results during experimentation on the *MSC*.
- Deployment of the *web crawlers* as independent web applications or microservices, having a dynamic interface which would ease the querying and retrieval of data, directly yielding the datasets in a *.json* file for ease of manipulation with no interaction with the raw *Python* project library at all.