CE807 Text Analytics: Assignment 2 (Includes Tasks 1 and 3)

Student ID: 1802490

ABSTRACT

Multi-label text classification finds its applicability in many domains, where automated article tagging is certainly one of them. However, it is not uncommon to discover that many text documents of interest are not publicly available, while many machine learning approaches assume the contrary. Fortunately, the related metadata, such as titles, are often readily accessible. According to recent research, it is certainly possible to classify documents using only their titles almost as accurately as with the access to the full-texts. However, it seems to be the case that standard decision trees did not receive much attention in the context of this problem. Thus, this work attempts to close this gap by evaluating vanilla decision tree models and simple tree-based ensembles in the task of extreme multi-label learning using publicly available datasets of document titles. The results obtained suggest these methods are not powerful enough to deal with extreme multi-labeling given their prediction quality, training time and excessive memory use.

1 INTRODUCTION

The importance of text analytics in both business and science has grown significantly over the years, perhaps mostly due to ever increasing consumption of text through the internet. One of the main branches of this field is text classification, a task commonly encountered in order to handle web search queries or classify text documents. For instance, it is often desirable to assign a set of appropriate labels or tags to a new article. As manual tagging is time-consuming and increasingly unfeasible due to the tremendous amounts of labels in practical applications, it is clear that automated procedures are necessary.

Depending on the specificity of the problem at hand and considered number of labels, the classification itself can be binary or multi-level, where the former deals with only two classes (class and no-class), while the other handles more categories. The latter further differs depending on how many labels are expected to be assigned to one data instant. Multi-class classification anticipates each data record to be a member of one and only one class, whereas multi-label classification, also called multi-label learning, allows each instance to have a set of labels. Furthermore, due to unique challenges posed by the problems including thousands of labels, they are often referred to as eXtreme Multi-label Learning (XML). Many of today's practical problems fall into this extreme category, such as tagging new Wikipedia articles, where more than a million labels are available.

Another problem of text classification comes from the fact that a full content of considered documents is not always openly available, a problem commonly present in scientific journals. Thus, a question arises whether such articles can be categorised using publicly available metadata, like titles. This alternative approach would solve not only the accessibility issues, but also the storage overhead as article's metadata usually weigh only a fraction of its full-text equivalent in terms of memory. As it has been shown in [6, 13], text documents can be indeed tagged using titles as effectively as when having access to their full content. Although both studies thoroughly test various learning methods, including both classic and modern techniques, they seem to miss one of the most popular group of Machine Learning (ML) methods. Decision Trees (DTs).

Therefore, motivated by the identified gap and the fact that DTs are known to perform quite well within the XML domain [8, 10, 16–18], the goal of this work is to investigate how tree-based ML methods perform in the title-based text classification task. Furthermore, due to the nature of decision trees, it is also natural to check how simple ensembles of trees perform in this task. Finally, due to the properties of the datasets incorporated in this project, the problem being solved can be categorised as the extreme multi-label learning.

This document is structured as follows. Firstly, some preliminary knowledge about decision trees and ensembles is presented in Section 2, critical to further understanding of methods incorporated in this study. Section 3 contextualises this work within existing literature of multi-label learning. Then, Section 4 discusses the methodology undertaken in this work, including the description of datasets, utilised methods and evaluation metrics. Next, the experimental design and obtained results are demonstrated as part of Section 5, followed by the discussion of various aspects of this work in Section 6. Finally, Section 7 concludes the report.

2 BACKGROUND

Before discussing further details of the existing XML methods and those proposed for this specific project, it is perhaps helpful to briefly go through the core ideas and ML algorithms that are of major interest. Thus, this section presents the main principles behind decision trees as well as the combination of many models, also known as the ensembles. A comprehensive account of both approaches is provided in [2].

2.1 Decision Trees

Decision Trees are one of the most common Machine Learning techniques, often being chosen as an initial method to explore the problem at hand as DTs are lightweight and relatively simple to incorporate. They are suitable for both classification and regression tasks, including text classification [1]. Such trees are represented as a hierarchical structure, created during the process called tree induction, and consist of decision nodes, edges and leaf nodes. An example of a simple DT is depicted by Figure 1. These structures aim to partition the data by given attributes and they corresponding values to eventually land in a specific class as the outcome. The most popular node splitting criteria are the Gini index and information gain.

One of the advantages of decision trees is certainly their transparency as the resulting tree can be visualised and further investigated. In addition, the hierarchical structure can be converted into human-readable IF-THEN rules. Furthermore, DTs can shed some light on the feature importances within the data as the splits located closer to the root are considered more informative. Trees

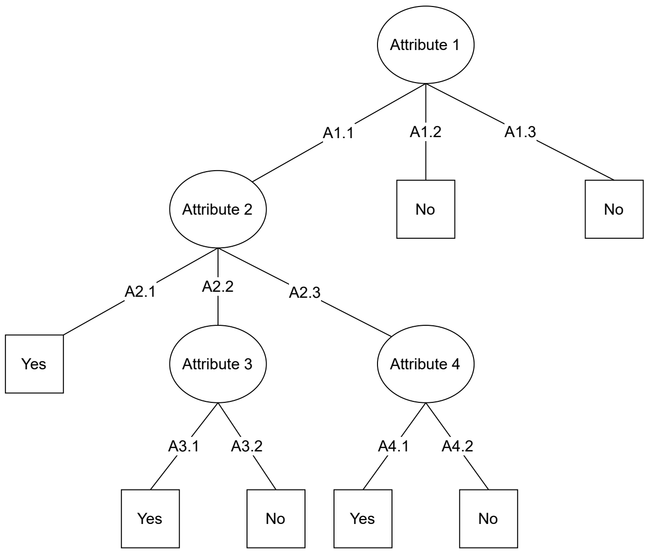

Figure 1: An example of a simple decision tree with four attributes and two classes.

also do not require any data normalisation and often perform much better on categorical data.

On the other hand, decision trees are known to struggle with high-dimensional inputs, often leading to overfitting, though there are prunning techniques to counteract this phenomena. It is also difficult to parallelise the process of tree induction, making it problematic if datasets are considerably large. Finally, although DTs handle categorical data naturally well, the opposite happens with continuous ones.

2.2 Ensembles

The idea behind ensembles is to combine multiple models into one complex method that hopefully outperforms those in separation. Depending on a concrete mixing procedure, ensembles can consist of completely different ML methods, are able to reduce overfitting, or even deliver high-performing models based on relatively simple and weak algorithms. Perhaps the most common technique to combine many learners' outputs it *voting*, where all models are trained on the entire dataset and their predictions are aggregated into a single value through a simple function, such as median, sum and so forth. *Stacking* approach further extends this idea by replacing the simple aggregate function with another meta learner, where its inputs are the outputs from the models of the previous layer. It is possible to take this idea even further and design many layers of learners connected to each other. There are also different variants of ensembles with regards to how they sample the original dataset. For instance, *bagging* is the approach where each learner draws samples randomly with replacement. As a consequence, some data instants may occur more than once or not at all for selected models, often resulting in reduced overfitting. It is also worth noting that the learning process of such an ensemble can be performed in parallel as the inner learners do not depend on each other. This very feature is contrary to the other popular technique, *boosting*, where the training of each model occurs sequentially. This is because each consecutive model is trained on the samples that the previous model struggled to classify, resulting in increasingly specialised models throughout the training.

3 RELATED WORK (TASK 1)

This section discusses the state-of-the-art learning techniques in the context of multi-label classification and its extreme sub-domain. The latter is discussed in detail as it is directly connected to this study and the methods incorporated. In addition, the latest works on title-based multi-label learning are presented, which attempt to solve the same type of problem as this project. For this reason, these studies can be considered a departure point of this work.

3.1 Multi-label Learning

The standard multi-label problems usually involve a moderate amount of labels per dataset, often less than 100. For a comprehensive review on multi-label learning techniques, consult [24], although it does not discuss the extreme cases.

Perhaps one of the most interesting and relatively recent studies applied Convolutional Neural Networks (CNNs) to separate text characters instead of words, introducing the Char-CNN [26]. One of the features of the model is that it does not require any prior information about the language structure and hence is suitable for many languages. The procedure incorporates a standard back-propagation algorithm to optimise the model and a 1-D convolution with temporal max-pooling, which is the equivalent of the 1-D max-pooling used in computer vision. This enabled the model to be deeper than 6 layers. The Char-CNN consumes a sequence of characters encoded as one-hot vectors of a fixed-size alphabet (70 in this case). In addition, this technique utilises the English thesaurus for the purpose of data augmentation. The study compares the newly developed method to many traditional as well as deep learning algorithms. Moreover, the authors introduce eight large scale multi-label datasets of varying size and complexity (127,000 - 4 million samples; 2-14 labels). The Char-CNN reports competitive results to existing methods, where it performs considerably better on larger datasets. The idea of character-based CNNs was further explored as part of [5], which introduced a VD-CNN method. Inspired by models applied to the computer vision domain, the authors tried more advanced deep learning structures and tested various depths on the neural network (9, 17, 29 and 49). According to the evaluation performed on the same datasets as [26], the setting with 29 layers delivered the best accuracy. It is also worth noticing that VD-CNN used only raw inputs to train the model, meaning no pre-processing or data augmentation was involved.

A completely different approach was taken in [21], where the authors employed graph neural networks and created a method called TextGCN. Moreover, it learns embeddings for words and documents, and builds one big text graph out of them based on their frequencies. Then, the method turns the problem of text classification into a node classification. Interestingly, the authors pointed out that the employed neural network performed the best when having only two hidden layers. Deeper solutions were found to be inferior in terms of performance. According to the experiments conducted on relatively small datasets (7,400 - 18,848 documents and 2-52 classes) the proposed technique outperformed other existing deep learning and graph-based approaches. The authors hypothesise that it could be due to the TextGCN being able to capture word-document and word-word relations. Despite the significant performance gain, the developed method has a major drawback of being transductive, meaning the access to unlabeled test data is necessary at the time of the graph construction, which happens during the training time. For this reason the method's practical application is very limited as it needs to re-build the graph everytime new unseen samples arrive.

3.2 Extreme Multi-label Learning

The extreme case differs from the standard multi-label learning by dealing with extremely large numbers of labels, often thousands of them, where each data point is assigned a subset of them. In addition, there is often a problem of sparsity, where some labels are present in only a few data examples, often called tail labels. Over the years, different approaches have been explored that try to exploit various aspects of XML.

3.2.1 One-Vs-All. Also known as One-Vs-Rest. The general idea is to train a separate classifier for each label, resulting in N classifiers given an N-class problem. Such solutions usually deliver high accuracy results, but are time-consuming in terms of both training and prediction, especially if N is really large (thousands or hundreds of thousands).

One of the well-established XML methods is Primal-Dual Sparse (PD-Sparse) [23], which tackles both multi-class and multi-label classification problems. It is based on a Fully-Corrective Block-Coordinate Frank-Wolfe (FC-BCFW) algorithm that utilises both primal and dual sparsity. On a high level, the method employs separate linear classifiers per each label and incorporates a negative sampling technique to improve training time and prediction accuracy. This sampling strategy is set to use the examples misclassified by a given classifier in previous training iterations or if the loss exceeds a defined maximum margin. The reported results show competitive accuracy under sub-linear training time. Unfortunately, the prediction time remained linear. The method was further extened into a parallelised version (PPD-Sparse) [22], in order to distribute the training process over more CPUs.

Another method, a Distributed Sparse Machines for Extreme Multi-label Classification (DiSMEC) [3], revisits the classic One-Vs-All approach and attempts to address the common problems of training and prediction complexity as well as model size. The training time is improved by the double layer of parallelisation, where all dataset labels are split into batches and distributed to different computation nodes. Then, on each such node the binary classifiers are trained per each label, where each model is trained on its separate CPU thread. This technique enables DiSMEC to be trained in hours for relatively big datasets. However, such a training strategy requires an access to clusters of multi-threaded computers. For instance, the study mentions the use of 1,000 CPUs, which definitely limits the applicability of the proposed solution. With regards to the model size, it is optimised by removing the weights of the classifiers that are close to zero, within [-0.01, 0.01] interval to be precise. As it turns out, as much as 96% of the weights can fall within that interval and therefore drastically reducing the model's size if such weights are discarded. Finally, the fast prediction time happens as a natural consequence of both distributed classifiers and a lower amount of the weights. According to the reported results DiSMEC beats SLEEC [4], FastXML [16], PD-Sparse [23] and other state-of-the-art methods on most of the datasets used.

Further advancement of One-Vs-All approaches was proposed as part of [15], which introduced a method called Parabel. It builds an ensemble of up to three trees out of dataset labels, where in each tree the labels considered to be similar enough are placed in the same leaf node. Parabel also incorporates a negative sampling technique by selecting positive examples of the other labels located in the same leaf node. This ensures only the critical and most confusing negative examples are chosen. As all the leaves contain One-Vs-Rest classifiers, the overall prediction result is the probability of the labels in the leaf matching the new data example, or the average in the case of the ensemble. As a consequence of these functionalities, the authors successfully scaled the method to a dataset consisting of 7 million data points, delivering the accuracy close to DiSMEC [3] but with much shorter training time and smaller model size. Parabel was further parallelised into PParabel [12] to speed up the training even more.

During the times of ever growing datasets and their complexity, the developed methods so far quickly become unfeasible to use. To counteract this issue, a Scalable LInear extreme ClassifiErs (Slice) technique was proposed [7], able to scale to 100 million labels and 240 million training examples. Furthermore, it offers logarithmic training and prediction times with regards to the number of labels, thanks to the negative sampling strategy. As in all One-Vs-Rest methods, Slice trains a separate linear classifier per each label. Moreover, due to negative sampling, the classifiers are fed with only the most confusing samples. To cut the prediction time, the method considers only the most probable labels during evaluation rather than checking all of them. Apart from scaling to enormous data sizes, Slice also beats state-of-the-art methods on the usual XML benchmarks and offers shorter training time.

3.2.2 Embeddings. One of the problems of XML is the sparsity and high-dimensionality of inputs and outputs. This type of methods attempt to map feature and label spaces (or sometimes both) to lower dimension dense representations consisting of latent variables, also called embeddings.

One of the classic embedding-based methods is SLEEC [4], which consists of two major steps: learning embeddings and k-nearest neighbour (kNN) classification. First, it compresses the labels to lower dimensions in order to capture the inter-label correlations and the distance within the closes neighbours. At prediction, it clusters the training data and learns embeddings for each group. Then, it performs kNN search to classify a new data point. To make the solution more stable and accurate, an ensemble of many models is constructed, where each of them operates on different data clusters. Further studies attempted to improve SLEEC and address its major drawbacks, such as training and prediction time. For instance, the work done as part of [19] introduced the AnnexML, which is claimed to be 58 times faster than SLEEC with regards to prediction time, while matching or even surpassing state-of-the-art accuracy. AnnexML utilises the kNN graph algorithm to build the label graph, which is then mapped into the embedding space.

At prediction time, the embedding space is then explored to find suitable labels.

Another method, Robust Extreme Multi-label Learning (REML) [20], focuses on the problem of tail labels, that is, the ones assigned to very few data points. The study proposes an optimisation method that explicitly takes into account such labels. In addition, the main optimisation problem is divided into smaller sub-tasks, each of which can be solved separately in parallel, effectively speeding up the training time. However, as pointed out by the authors, increasing the number of sub-problems also decreases overall accuracy of the solution. Thus, this feature can be seen as the means of prioritising either the training time or overall accuracy, depending on the importance under given circumstances. According to the results, REML matches the performance of SLEEC [4], FastXML [16] and other classic XML methods on small and large datasets.

3.2.3 *Trees.*

Tree-based methods attempt to take the advantage of the fact that their prediction time is sub-linear, or even logarithmic if the tree is balanced. In addition, it is quite common to build ensembles of trees in order to deliver even stronger and more robust models.

The FastXML [16] is a well-established tree-based method within the XML community. It learns a hierarchy over the feature space and optimises the nDCG-based objective at each node for better partitions, as opposed to the usual Gini index or clustering error employed in previous methods utilising decision trees. The overall goal is to build an ensemble of such trees in order to increase the robustness of the final solution. According to the experiments, it reliably induces balanced trees, delivering logarithmic prediction time as a consequence. The method was extended into PfastreXML [8] by introducing a propensity scored objective function and including classifiers sensitive to tail labels.

The idea of decision trees for XML problems was further explored in [10], which proposes a random forest-based technique called ReWeighting Forest (ReWF). The procedure assigns higher weights to the data points that are more difficult to classify. Furthermore, it discards the individual trees that are of lower quality to reduce the overall size of the solution and improve accuracy. The authors report superior performance when compared to FastXML [16] and PfastreXML [8].

3.2.4 *Deep Learning.*

Encouraged by the Deep Learning (DL) revolution in computer vision and language modelling, the deep neural networks are also being applied to text classification problems. One of their biggest advantages is the ability to handle vast amounts of input features, often without any pre-processing. This certainly makes them attractive in terms of real applications, though, on the other hand, they often have substantial computational needs.

One of the most interesting text-based DL methods introduced relatively recently is FastText [9], mostly due to its simplicity. It averages the word embeddings to represent the input documents and ignores words order. It also utilises a simple linear softmax classifier. As a consequence of this basic principles, it is very fast to train and still delivers good results. However, it is worth noting that FastText was designed primarily to deal with multi-class classification tasks, though it is applicable to multi-label problems to some extent as well.

The official first attempt to use DL explicitly for XML problems was done in [11], which introduced the XML-CNN method, based on the existing CNN text classifiers that aim to solve multi-class classification tasks. XML-CNN learns word embeddings, adopts dynamic max pooling and optimises the binary cross-entropy loss function. Unlike previous CNN-based models for text classification, XML-CNN introduces a hidden fully connected layer between the pooling and output layers. According to the results, this helps to learn better representations and increases the accuracy. The method scales to a dataset with over 2 million examples and 500 thousand labels. Although it outperforms state-of-the-art methods, or is at least the second best one, it is fair to say that XML-CNN is relatively expensive to train, even assuming the access to GPUs.

Further efforts to apply DL techniques to the XML realm is presented in [25]. The introduced algorithm, Deep embedding for XML classification (DXML), builds the label graph during the training stage, where two graph nodes are connected if the two labels occur together in any sample. Then, the graph is transformed into a low-dimensional embedding. Similarly, the input features are also mapped into a low-dimensional representation, through a deep neural network. At the next stage, both feature and label embeddings are mapped into a common embedding space. At prediction, the method gets the input to the low-dimensional representation and applies the kNN search to find similar data points, resulting in the average of labels of found samples. Similarly to SLEEC [4], it also clusters the inputs first, but it does so in the low-dimensional embedding space to improve stability and avoid the need to build ensembles. Overall, the reported accuracy is close to other state-of-the-art methods, though, as usually with DL procedures, training is relatively expensive.

3.3 Title-based classification

Most of the studies and their proposed methods assume access to the full content of the text documents of interest, which is not always the case in practice. However, even in those restricted access situations, it is still possible to get other information about the documents, such as abstracts, titles and so forth. Perhaps this limited amount of information is sufficient to perform the usual text classification tasks as with a full access to articles' content.

The first study exploring this research direction investigated the feasibility of using titles instead of full-texts to automatically annotate the documents [6]. The authors thoroughly compared different vectorisation and classification techniques. They investigated both concept and term frequency methods as well as TF-IDF and BM25 representations. The classifiers tried comprised many classical approaches, such as Naive Bayes, linear methods, neighbourhood-based procedures and neural networks, just to name a few. Some of them adjusted as One-Vs-All techniques, others organised as simple stacking ensembles. To assess the feasibility of using the titles only, the study compared all the variants of proposed solutions on both titles and full-texts, where the sample-averaged F-score measure is reported as it is argued to be the most suitable one given the importance of each sample's accuracy. In terms of pre-processing, the study suggests that the combination of concept and term frequencies is superior to any other setting. Furthermore, the classification results show that titles can be almost as effective as full-texts in the task of article tagging, where a classic Multi-Layer Perceptron (MLP) model performed the best.

Encouraged by the promising results obtained by [6], this line of work was further continued in [13], where the authors explored deep learning methods for the task of title-based article tagging. More precisely, they investigated MLPs, Recurrent Neural Networks (RNNs) and CNNs. In addition, the study introduced two new multi-label datasets, which can be considered an XML problem given the amount of labels and data points involved. According to the reported results, in some settings the title-based solution can even surpass the performance of the full-text one. The other interesting finding was that on average the MLP-based methods performed better than CNNs and RNNs.

4 METHODOLOGY

This section discusses the major aspects of this project's methodology. The goal is to explore standard tree-based models in the task of semantic indexing by using only documents' titles. Thus, two publicly available title-based datasets, described in Section 4.1, are being utilised. However, before feeding the data into a model, they are first pre-processed into an acceptable representation, so a model can consume them. This is described in Section 4.2. Furthermore, the details of incorporated learning methods are demonstrated in section 4.3. As with any method-based research, it is imperative to properly evaluate the quality of proposed procedures. The exact details of evaluation metrics involved in this study are presented in Section 4.4. Overall, the high-level overview of the design and involved components are depicted in Figure 2.

4.1 Data

The two datasets used for evaluation, namely EconBiz and PubMed, are publicly available through the Kaggle website[1]. Both of them are organised in the same way, that is, each data point consists of a title and a list of corresponding labels. It is worth noticing that the number of labels per each data entry can vary. The main properties of the two datasets are shown by Table 1. In addition, both datasets were introduced by [13], although the study used full-texts as well, which are not part of the public dataset.

	EconBiz	**PubMed**
Samples	1,064,634	12,834,026
Labels	5,661	27,773
Samples per label	819.1	5,852.3
Labels per sample	4.4	12.6
Words per sample	6.88	9.6

Table 1: The properties of the two datasets used in the project.

EconBiz is a carefully prepared English dataset that comes from the EconBiz[2] digital library, provided by ZBW - Leibniz Information Centre for Economics[3]. The entries are annotated with STW labels.

The PubMed dataset originates from the training data that were part of a BioASQ challenge[4]. The data are in English and the labels are in the format of MeSH terms.

4.2 Pre-processing

This study takes the same pre-processing strategy as in [13]. The titles are first converted into unigram counts. Then, the 25,000 most frequent entries are translated into TF-IDF representation, which in turn is finally fed into a classifier. In addition, the labels are binarised, meaning they are turned into a sparse matrix of one-hot encoded vectors of all categories. In general, three distinct pre-processing stages can be distinguished, presented in Figure 2 as *Count Vectorizer*, *TF-IDF* and *MultiLabel Binarizer*, the implementations of which were provided by [14].

Count Vectorizer. It takes the titles in a raw text format and converts them into unigrams. In addition, the occurrence of the tokens is being counted, resulting in a matrix of unigrams and their number of occurrences. Furthermore, as part of the unigram conversion a default set of stop words for Englush is removed from the titles. Finally, only the most frequent 25,000 terms are considered to build the vocabulary.

TF-IDF. The resulting count matrix from the *Count Vectorizer* is fed into the *TfidfTransformer* in order to transform it into a normalised Term-Frequency Inverse Document-Frequency (TF-IDF) representation. This technique is useful because highly frequent terms could otherwise dominate the data, while in practice the ones rarely occurring are also of great interest.

MultiLabel Binarizer. This functionality simply transforms the nested lists of labels to a binary matrix of size $samples \times labels$ as this is the preferred format for multi-label problems.

4.3 Methods

Encouraged by the successes of tree-based solutions in XML problems [8, 10, 16–18] and the fact that previous studies focused on title-based text classification did not investigated that direction [6, 13], this work attempts to employ standard decision trees and a few simple ensembles of them. More precisely, the goal is to train and evaluate the following three classifiers: Decision Tree, Random Forest and Extremely Randomised Trees, further referred to as DT, RF and XTS respectively. Similarly to pre-processing, the implementations are provided via [14].

Decision Tree. This is a standard decision tree classifier that is based on binary splits and gini criterion to measure the quality of the splits. In order to prevent the tree from overfitting, the following three parameters are being tuned and investigated: *max_depth*, *min_samples_split* and *max_features*. The *max_depth* one simply allows the tree to be grown to a certain depth; otherwise a fully-grown tree is constructed. *min_samples_split* controls the minimum number of data examples necessary to make a split at a given decision node. *max_features* is the maximum number of features that are used when splitting the nodes.

Random Forest. This is a classic bagging ensemble of decision trees, where each base estimator is trained on a slightly different dataset due to it being randomly sampled with replacement from the original one. The output of the method is the average prediction

[1]https://www.kaggle.com/hsrobo/titlebased-semantic-subject-indexing
[2]https://www.econbiz.de/
[3]http://www.zbw.eu/de/
[4]http://www.bioasq.org/

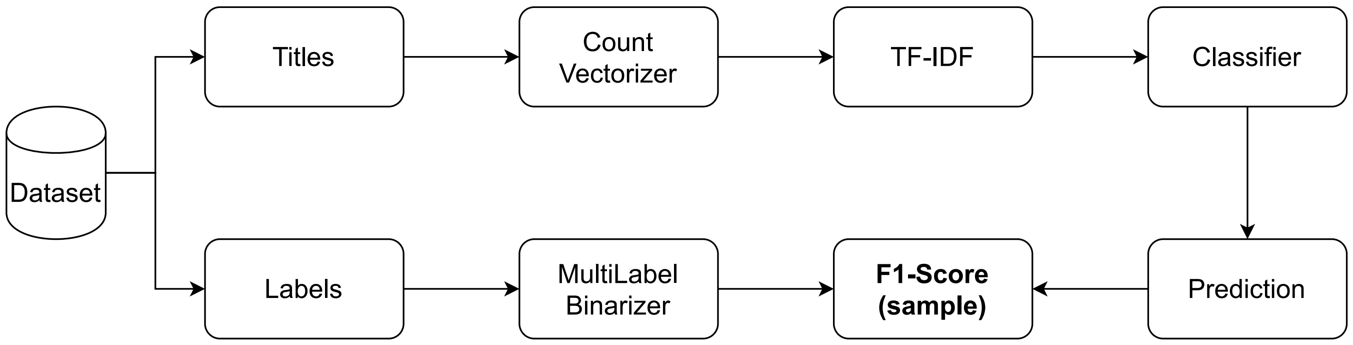

Figure 2: The pipeline of the proposed method. The titles are preprocessed into 25,000 most common unigrams and represented as TF-IDF terms. Such data is then fed into a classifier that returns predictions, which are in turn compared with binarised labels to produce sample-based F1-score metric.

of the all inner estimators. This strategy is specifically suitable for the classifiers that are prone to overfitting, which is definitely the case with the vanilla DTs. Thus, this method is expected to perform better than pure DT in case it exhibits high variance. Apart from the data sampling technique, the other layer of randomness occurs at node splitting as a random subset of features can be involved in the process. In terms of the parameters being investigated for this model, the same three apply as for the decision tree with an addition of the number of inner trees being controlled by the *n_estimators* parameter.

Extremely Randomised Trees. XTS approach is very similar to RF with the exception that it takes the randomness one step further. More concretely, when searching for a subset of features as part of the node splitting, the thresholds are random as well for each considered feature. As a result, this technique reduces model's variance even further. With regards to the model's parameters, the same set applies as with the random forest.

4.4 Evaluation

The evaluation measure employed in this project, sample-based F_1 score, is the same as in [13]. Apart from the convenience of comparing the same metrics between the studies, it is also argued in [6] that this specific metric fits the best to an eventual application, where the best performance per document is of interest. The sample-based F_1 score formula is shown in Equation 1, where precision is the fraction of returned results that are actually true, whereas recall is the proportion of true instances that were identified. The metric is calculated for each sample and then averaged over all data points.

$$F_1 score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{1}$$

It is also worth pointing out that the metric is calculated as part of either a 5-fold cross-validation or the evaluation on a held-out test data, which is always a 10% of the entire dataset. Furthermore, apart from the F_1 measure, both training and prediction times of the tested models are investigated as time complexity is considered an important aspect of XML.

5 EXPERIMENTS

All the experiments were run on a high performance computing cluster with many computing nodes available, each of which comprised of multiple CPUs. In fact, the multi-threading aspect was only applicable to the ensemble models as each inner estimator can be trained in parallel. In such cases, the number of requested CPUs always matched the amount of estimators. Vanilla DTs were in general run within a single-core environment, though whenever they were part of a K-fold cross-validation, each fold was running on a separate CPU thread. In addition, all the scripts necessary to replicate the experimental runs are provided with the code supplied for easier reproducibility.

Overall, two types of experiments were conducted. Firstly, the standard decision tree was fine-tuned on a fraction of the smaller dataset. Once optimised, the best parameters found were then used for DT and ensemble training on larger chunks of the data. Both types of experiments investigated the F_1 measure, training and prediction time.

5.1 Hyperparameter tuning

The objective of this experiment was to find the optimal parameters of the DT classifier. To speed up the process, only 10% of the EconBiz dataset was involved in these tests. Before each training session, the aforementioned fraction of the data were further split into 90% training and 10% testing, though the latter was not incorporated in this particular test. The remaining 90% was utilised to run 5-fold cross-validations using 5 CPU cores.

As discussed in Section 4.3, the three parameters of interest were max_features, max_depth and min_samples_split. The goal was to find the combination of the parameters for which the model returned the highest F_1 measure, taking into account the training and prediction time as well. All the 12 configurations of the test runs and the results obtained are provided in Table 2.

According to the results (Table 2), setting the min_sample_split to either 10 or 20, while leaving the other two parameters to defaults, gave the highest F_1 score (0.23). In fact, the configuration where the aforementioned parameter was set to 20 can be actually considered a better choice as it offers a slightly shorter training time than

m_f	m_d	m_s	F1	Fit [min]	Pred [s]
sqrt	None	2	0.19	9.50 ± 0.22	37.25 ± 1.96
None	None	2	0.22	161.78 ± 2.20	23.75 ± 0.22
None	None	10	0.23	185.99 ± 3.29	28.35 ± 0.54
None	None	20	0.23	163.66 ± 1.63	32.00 ± 0.38
None	None	50	0.22	159.88 ± 1.37	33.06 ± 1.98
None	None	100	0.20	306.80 ± 7.45	73.06 ± 19.04
None	10	20	0.04	8.82 ± 0.50	28.70 ± 2.86
None	50	20	0.11	47.26 ± 1.49	36.14 ± 4.36
None	100	20	0.15	52.99 ± 0.64	31.02 ± 1.77
None	200	20	0.19	81.18 ± 0.51	31.69 ± 0.60
None	500	20	0.22	107.02 ± 1.06	23.45 ± 0.23
None	1000	20	0.22	131.34 ± 0.92	31.96 ± 1.70

Table 2: The results obtained from 5-fold cross-validation of the Decision Tree model tested against 10% of the EconBiz dataset. Columns: m_f - max_features; m_d - max_depth; m_s - min_sample_split; F1 - sample-based F1 score; Fit - training time in minutes; Pred - prediction time in seconds.

the other. Thus, this winning setting was further employed in the second part of the experiments.

5.2 Scaling up

As soon as the optimal model parameters were identified as part of the first experiment, it was possible to evaluate all the selected methods on larger subsets of the data. The selected models were tested against four different sizes of the EconBiz dataset, namely 10, 20, 50 and 100 percent of the samples. Unfortunately, due to the overwhelming size of the PubMed dataset, only a small portion of it (1%) was involved in the experiments. At the beginning of each experimental run, the data were divided into 90% training and 10% testing. With regards to the ensemble models, each of them was tested with 5 and 10 inner estimators, running within 5 and 10 CPUs environments respectively. The overall goal of this experiment type was to investigate the quality of predictions of selected methods and their feasibility to scale to large datasets. The results obtained from various runs are presnted in Table 3.

6 DISCUSSION

The experimental results of both types of test are discussed in this section, grouped into separate subsections for convenience.

6.1 Tuning

The cross-validation runs explored a range of values for the three parameters of interest (see Table 2), in the hope of finding the best performing setup. The model clearly performed worse when the maximum amount of features considered was limited to a square root of the number of all the features. This is perhaps not surprising as the feature space was already cut to the most frequent 25,000 terms during the pre-processing stage. Similarly, tweaking the allowed tree depth also resulted in worse performance. Although the depth limits of 500 and 1,000 resulted in almost the best F measure, their training times were only a bit faster than the best performing setups, diminishing any practical gains of the limited depth cases.

Model	Metrics	EconBiz				PubMed
		10%	20%	50%	100%	1%
DT	F1 score	0.24	0.28	0.32	0.29	0.23
	Fit [h]	6.44	9.45	29.74	94.79	42.41
	Pred [s]	12.08	17.48	78.36	301.55	49.02
RF(5)	F1 score	0.22	0.21	x	T	0.23
	Fit [h]	2.51	2.87			17.56
	Pred [s]	16.77	35.62			128.74
RF(10)	F1 score	0.26	x	x	x	x
	Fit [h]	8.47				
	Pred [s]	67.69				
XTS(5)	F1 score	0.25	T	x	T	T
	Fit [h]	1.75				
	Pred [s]	22.48				
XTS(10)	F1 score	0.24	x	x	x	x
	Fit [h]	3.31				
	Pred [s]	58.88				

Table 3: The results of experimental runs involving various models and datasets. F1 score refers to the sample-based metric. Fit and Pred denote training (hours) and prediction (seconds) time respectively. DT - Decision Tree; RF - Random Forest; XTS - Extremely Randomised Trees. Given model(n), n refers to the number of estimators used to build the ensemble. Table cells marked with x denote the setting was not run, while T means the training was terminated by the server.

The minimum sampling split factor was found to improve the score from the default setting of 2 resulting in 0.22 to the setting of 10 and 20 offering the score of 0.23.

Most of the combinations delivered scores ranging from 0.19 to 0.23. Only a small fraction of them went below that level, though understandably due to very strict depth limits. On one hand, it can be said that the performance of the DT was very stable regardless of the chosen parameters. However, it is also fair to admit that the score of 0.23 is not particularly high given the T1 EconBiz results of [13] ranging from 0.357 to 0.391, though their testing protocol was slightly different.

In terms of the time complexity, the best performing settings usually surpassed two hours of training, sometimes even approaching three. The solutions with seriously limited tree depth naturally were faster to train. A unique setting that balanced the score and training time particularly well was the one with even more limited feature space. That setup took only about 10 minutes to train.

6.2 Testing

The second stage of the experiments attempted to scale the selected methods to larger dataset sizes and evaluate their performance (see Table 3). All five models were successfully trained on a 10% of the EconBiz data. The best scoring method (0.26) was a random forest with 10 estimators, though it was the slowest to train (over 8 hours). The weakest model (0.22) was also based on the random forest approach but with 5 estimators. Perhaps the best balance of score and training time was delivered by XTS with 5 inner trees, which got a 0.25 score under less than 2 hours of training. Moving forward to a 20% of the EconBiz data, only two models were successfully trained, vanilla DT and RF(5). XTS(5) was terminated due to memory issues. The standard tree performed definitely better in terms of the score (0.28) compared to the forest (0.21). On the other hand, DT was much slower to train (almost 9.5 hours vs less than 3). Further increases in training size worked only for the decision tree. After training on 50% of the data, its score increased to 0.32. Unfortunately, at the cost of almost 30 hours of training. A full-scale training delivered a slightly worse performance of 0.29, which took almost 4 days. Similar training sessions on the full data performed for RF(5) and XTS(5) were terminated by the server due to memory problems.

Training on the PubMed data was even more challenging as it not only consisted of more samples but also 5 times more labels than the EconBiz dataset. This can be observed by examining the training times. As a consequence, only two models were successfully trained on a 1% of the PubMed data, DT and RF(5). Both models obtained the same score of 0.23, where the ensemble was much faster (17.5 vs 42.5 hours), though definitely not quick. The other ensemble, XTS with 5 estimators, was terminated by the server during training.

Overall, the standard DT method was the only one being able to scale to the full EconBiz dataset, and complete a training session on 1% of the PubMed data. While its score was improving when using increasingly large data sizes, the training time also rose sharply as well unfortunately. Random forest with 5 estimators can be classified as a second-best method due to it being able to scale to 20% of the EconBiz data and 1% of the PubMed one. Although the scores obtained by RF(5) were worse than DT, its training times were significantly faster. All other models, namely RF(10), XTS(5) and XTS(10) were able to complete training on only 10% of the smaller dataset.

Training time certainly varied from method to method. The standard DT was overall the slowest, though perhaps unsurprisingly as it does not take advantages of multi-threaded environments. The benefits of parallelisation can be observed in almost all instances of the ensembles training. On the other hand, a single decision tree was definitely much more memory efficient, enabling it to scale to the full EconBiz data, whereas the ensembles clearly had issues with excessive memory consumption, resulting in terminated training sessions.

With regards to other studies that incorporated the same datasets, [13] obtained certainly better F_1 scores for both of them. The lowest score achieved by the authors on the EconBiz data was a 0.357 (T1), whereas the best performance of this work was a score of 0.32 (50%). The difference on the PubMed data is even bigger at the worst score from the aforementioned study was at 0.419 (T8), compared to 0.23 (1%) from this project.

7 CONCLUSION

In summary, this study explored the feasibility of utilising standard decision trees and simple tree ensembles in the task of title-based extreme multi-label text classification. The following are the concluding remarks and a few suggestions for a possible future work.

XML problems are extremely difficult. The extreme multi-label learning problems are truly unique and greatly differ from the classic machine learning tasks. Firstly, the input features are high-dimensional and highly sparse, eliminating many methods not handling this particular aspect very well. Secondly, many modern datasets within XML consist of millions or even hundreds of millions of samples, pushing the learning methods to their capacity limits. Thirdly, the amount of labels to deal with are often enormous. And as many classic ML methods are rather designed for multi-class classification problems, they usually require advanced modifications to handle multi-label scenarios, not to mention the extreme cases, which demand further optimisations. Finally, as a consequence of this aspects of the XML area, further problems arise, such as model training lasting for days or immense model sizes. Therefore, it is clear that XML demands highly sophisticated and often distributed solutions to effectively solve those challenging problems.

Standard DTs perform poorly in XML. At this point it is clear that classic decision trees designed to rather handle multi-class problems are not readily applicable to XML tasks. They indeed can solve the problem to some degree as the F_1 score of 0.32 is far from random, but nevertheless it is not a state-of-the-art performance. This is not to say that tree-based methods are not suitable for XML as there are clearly successful applications of them [8, 10, 16–18]. However, these approaches incorporate advanced partitioning procedures, optimise specific loss functions and take advantage of parallel computing as much as possible. As demonstrated by this study, even simple bagging methods do not improve much over the poor performance of the standard decision trees.

Future work. Even though pure decision trees proved to perform poorly in XML, there is still a chance for ensembles to deliver some new interesting solutions. For instance, both studies investigating document titles as part of [6, 13] reported strong MLP models with only one hidden layer. Such a shallow model is perhaps simple enough to build an ensemble of MLPs and train it under sensible amount of time. A slightly different approach could involve all the four methods tested in [13] to build a complex ensemble model out of them. However, given the deep learning models can be expensive to train even in separation, combining more of them into a single learning procedure may seriously limit the practicality of such a method.

REFERENCES

[1] Charu C. Aggarwal and ChengXiang Zhai. 2012. A Survey of Text Classification Algorithms. In *Mining Text Data*, Charu C. Aggarwal and ChengXiang Zhai (Eds.). Springer US, Boston, MA, 163–222. https://doi.org/10.1007/978-1-4614-3223-4_6

[2] Ethem Alpaydin. 2014. *Introduction to Machine Learning*. MIT Press.

[3] Rohit Babbar and Bernhard Schölkopf. 2017. DiSMEC: Distributed Sparse Machines for Extreme Multi-Label Classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17)*. Association for Computing Machinery, Cambridge, United Kingdom, 721–729. https://doi.org/10.1145/3018661.3018741

[4] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse Local Embeddings for Extreme Multi-Label Classification. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 730–738.

[5] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very Deep Convolutional Networks for Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, 1107–1116.

[6] Lukas Galke, Florian Mai, Alan Schelten, Dennis Brunsch, and Ansgar Scherp. 2017. Using Titles vs. Full-Text as Source for Automated Semantic Document Annotation. In *Proceedings of the Knowledge Capture Conference (K-CAP 2017)*. Association for Computing Machinery, Austin, TX, USA, 1–4. https://doi.org/10.1145/3148011.3148039

[7] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. 2019. Slice: Scalable Linear Extreme Classifiers Trained on 100 Million Labels for Related Searches. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19)*. Association for Computing Machinery, Melbourne VIC, Australia, 528–536. https://doi.org/10.1145/3289600.3290979

[8] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme Multi-Label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, San Francisco, California, USA, 935–944. https://doi.org/10.1145/2939672.2939756

[9] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, 427–431.

[10] Zhun-Zheng Lin and Bi-Ru Dai. 2017. Reweighting Forest for Extreme Multi-Label Classification. In *Big Data Analytics and Knowledge Discovery (Lecture Notes in Computer Science)*, Ladjel Bellatreche and Sharma Chakravarthy (Eds.). Springer International Publishing, Cham, 286–299. https://doi.org/10.1007/978-3-319-64283-3_21

[11] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep Learning for Extreme Multi-Label Text Classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. Association for Computing Machinery, Shinjuku, Tokyo, Japan, 115–124. https://doi.org/10.1145/3077136.3080834

[12] Jiaqi Lu, Jun Zheng, and Wenxin Hu. 2019. PParabel: Parallel Partitioned Label Trees for Extreme Classification. In *Network and Parallel Computing (Lecture Notes in Computer Science)*, Xiaoxin Tang, Quan Chen, Pradip Bose, Weiming Zheng, and Jean-Luc Gaudiot (Eds.). Springer International Publishing, Cham, 82–92. https://doi.org/10.1007/978-3-030-30709-7_7

[13] Florian Mai, Lukas Galke, and Ansgar Scherp. 2018. Using Deep Learning for Title-Based Semantic Subject Indexing to Reach Competitive Performance to Full-Text. In *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries (JCDL '18)*. Association for Computing Machinery, Fort Worth, Texas, USA, 169–178. https://doi.org/10.1145/3197026.3197039

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[15] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Lyon, France, 993–1002. https://doi.org/10.1145/3178876.3185998

[16] Yashoteja Prabhu and Manik Varma. 2014. FastXML: A Fast, Accurate and Stable Tree-Classifier for Extreme Multi-Label Learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. Association for Computing Machinery, New York, New York, USA, 263–272. https://doi.org/10.1145/2623330.2623651

[17] Si Si, Huan Zhang, S. Sathiya Keerthi, Dhruv Mahajan, Inderjit S. Dhillon, and Cho-Jui Hsieh. 2017. Gradient Boosted Decision Trees for High Dimensional Sparse Output. In *International Conference on Machine Learning*. 3182–3190.

[18] Wissam Siblini, Pascale Kuntz, and Frank Meyer. 2018. CRAFTML, an Efficient Clustering-Based Random Forest for Extreme Multi-Label Learning. In *International Conference on Machine Learning*. 4664–4673.

[19] Yukihiro Tagami. 2017. AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-Label Classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. Association for Computing Machinery, Halifax, NS, Canada, 455–464. https://doi.org/10.1145/3097983.3097987

[20] Chang Xu, Dacheng Tao, and Chao Xu. 2016. Robust Extreme Multi-Label Learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, San Francisco, California, USA, 1275–1284. https://doi.org/10.1145/2939672.2939798

[21] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph Convolutional Networks for Text Classification. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (July 2019), 7370–7377. https://doi.org/10.1609/aaai.v33i01.33017370

[22] Ian E.H. Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. 2017. PPDsparse: A Parallel Primal-Dual Sparse Method for Extreme Classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. Association for Computing Machinery, Halifax, NS, Canada, 545–553. https://doi.org/10.1145/3097983.3098083

[23] Ian En-Hsu Yen, Xiangru Huang, Pradeep Ravikumar, Kai Zhong, and Inderjit Dhillon. 2016. PD-Sparse : A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification. In *International Conference on Machine Learning*. 3069–3077.

[24] Min-Ling Zhang and Zhi-Hua Zhou. 2014. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (Aug. 2014), 1819–1837. https://doi.org/10.1109/TKDE.2013.39

[25] Wenjie Zhang, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha. 2018. Deep Extreme Multi-Label Learning. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval (ICMR '18)*. Association for Computing Machinery, Yokohama, Japan, 100–107. https://doi.org/10.1145/3206025.3206030

[26] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-Level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 649–657.