Multi-Label Text Classification: Building a Classifier based on Title data

Registration Number: 1900939

KEYWORDS

text analytics, text classification, multi-Label classification, multi-label text classification, extreme multi-label classification, extreme multi-label text classification

1 INTRODUCTION

Intuitively, humans have the capability to distinguish things without much effort, up to the point where there are situations where humans cannot explain how they did it. As the big data era is upon us, data has increased massively in terms of volume, not only producing more data, but producing different type of data. Nowadays, instead of classifying something as A or B, many more letters and words have been added to the pool, making the task to classify even harder.

The purpose of this work is to introduce the field of multi-label classification, with a particular focus on text classification and extreme multi-label classification, and present classifiers that use a dataset with a high number of labels, and only contains title data from scientific documents. There have been numerous studies related to multi-label classification, however, the ones presented in this work will be related to extreme multi-label text classification. Additionally, different techniques and approaches used in multi-label and extreme multi-label classification will be explained. As [2, 4] proposed using only title data for document annotation, this work also seeks to take leverage of the fact that title data from scientific papers is easily available, and as new publications are constantly produced, the need for an automatic system that does this is necessary. Two approaches will be considered, 1-vs-All and deep learning based methods.

The structure of the rest of the document is: Explanation of multi-label classification, multi-label text classification, extreme multi-label classification, extreme multi-label text classification, the experiment section which explains the dataset, preprocessing, vectorization, classifiers, evaluation. Followed by a brief description of related works, and the presentation of results, discussion.

2 BACKGROUND (TASK 1)

As the complexity of classification problems increases, and knowing that multi-class classification has achieved outstanding results by taking a supervised learning approach to assign a single label to an instance [2], more complex problems have been presented such as multi-label classification. Multi-label classification tasks arose primarily from the need to analyse medical images and documents, and automate the process of a medical diagnosis from those resources [7]. The key difference between multi-label and multi-class classification, is that multi-label classification works with the assumption that a sample might have more than one correct label that describes it, whereas multi-class classification assumes that one sample will only have one correct label, but the pool of labels is more than two [10]. For example, a system that distinguishes between SPAM and no SPAM is binary-classification, a system that distinguishes if an email is an update, promotion, social media, and regular email, is a multi-class, whereas, a system that detects medical terms from an X-ray image, it might need to use terms that describe an arm, broken bone, and different medical terms, is multi-label classification.

Different approaches have been taken for multi-label classification, and they can be divided in three: problem transformation, algorithm adaptation and ensemble methods [4]. Transformation methods, as the name suggests, change the way the problem is tackled and instead of solving it as multi-label classification, the problem is divided into multiple binary classification problems [2, 4]. On the other hand, the algorithm adaptation approach does not change the type of problem at its core, but the algorithm is adapted to handle the multi-label assignation [3].

2.1 Multi-Label Text Classification

Many studies haven been done in the field of multi-label classification, especially for text. Basic machine learning algorithms have enable the rapid development of systems based on supervised learning techniques classic. Out of the many classic supervised learning algorithms, lazy learners have been considered a starting baseline for multi-label text classification problems, as they enable the easy adaptation of such problems [1].

Classic algorithms such as logistic regression, Näive Bayes, SVM and decision trees can also be used in multi-label classification problems, and this can be seen in [2], where it shows that problem transformation can enable the use of algorithms that do not have the capability to assign multiple labels to an instance. In this study, a comparison between using the full text of documents and only the title was investigated, and it was proved that using only the title of scientific papers can almost reach the same performance when using the full text [2]. At the same, it was shown how multi-label classification problems can be tackled, and how deep learning, in this case a multi-layer perceptron (MLP), outperformed the other classifiers [2].

Besides choosing the correct classifier, it is important to think carefully about feature representation, because it can affect the performance of an algorithm when training and predicting [5]. The text in documents needs to be represented as features, and in text classification, this is usually done by using bag-of-words (BOW) methods, such as unigrams, n-grams, binary vectors, TF-IDF vectors, which create a feature vector that represents the absence of words in a document or sentence [5]. These methods can be useful and often be considered as baseline, however, they ignore contextual information and might suffer from data sparsity [5]. To solve these problems, a solution can be the use of word embeddings, which captures the semantic information of text [5]. Studies such as [5], showed that by using word embeddings techniques such as word2vec, a better representation of the text can be gained and therefore, a possible increase in accuracy performance. Similarly to transfer learning, where a pre-trained model can be used in similar problems, there are similar models that have been trained with a large corpus of words and are possible to output a vector representation for a word or sentence, there are pre-trained generic models such as GloVe, or more domain-specific like PubMed. Theses models have been trained using a Convolutional Neural Network (CNN), which exploits and detects high-level representations of features in the input [5].

Recently, with the increasing amount and variety of data generated, the amount of labels that a sample can be assigned to, also increases. One key characteristic of multi-label classification is that the amount of possible labels to assign to each instance is low, and therefore, approaches such as problem transformation is possible and efficient. However, as the amount of labels increases, this approach becomes highly time and resource consuming [10].

2.2 eXtreme Multi-Label Classification

eXtreme Multi-label classification (XMLC) main objective is to learn a classifier that is able to automatically distinguish the correct tag(s) of a sample from a large tag set [10]. Compared to multi-classification, the pool of labels that is used to classify a sample is drastically large in size, up to cases where there are more than one million labels within the dataset [10].

XMLC is an on going field of research because of the different and important application fields, however, as the number of labels increases, the difficulty of the taks does too [8]. Multi-label classification approaches might be a first answer for this type of problem, however, approaches such as one vs All, i.e train a classifier for each label, clearly is not an efficient approach as the amount of classifier to train will be proportional to the amount of labels, and in this context, the pool of labels can reach millions [8]. Additionally, the prediction phase will be computational expensive as well, when a prediction is needed, all the classifiers will need to be evaluated, which will result in expending high amount of computational resources and time [8].Out of the many methods suggested to tackle XMLC problems, the majority of them can be categorized into four types: 1-vs-All, embedding-based, label tree-based and deep learning based methods [9].

1-vs-All methods, as mentioned before, treat each label independently and a classifier for each label is trained. Clearly, the computational complexity increases and the number of labels do, and for XMLC problems, the models complexity is high because of the high number of labels and samples [9]. However, solutions to reduce the complexity of the 1-vs-All method have been proposed, such as using sparse learning and efficient parallelization of high computing clusters that take advantage of the high number of cores [9].

Embedding-based methods general concept is to reduce the number of labels by compressing them into a different size for training, this compression is done by a compression function [9]. During the training phase, the classifier learns from a lower dimensional embedding vector, and when predicting, the classifier will predict a lower dimensional embedding vector that needs to be decompressed into its original form[9]. Nevertheless, this method brings the possibility of losing information through the compression and decompression processes because the labels and feature values are transformed into a low dimensional space [9].

Label tree-based methods, as the name suggests, operate under the same idea as decision trees, as they generate a tree by separating instances by features and have their corresponding labels partitioned, as a result, each leaf will represent a label and within that leaf, the node will contain another label [9]. One disadvantage of this approach is that it can also be computational expensive and have a low performance, specially in cases where there are labels that are not frequently represented in the dataset, also known as "tail labels", as the tree will need to classify those instances [9]. A well known tree based method is FastXML [8], which instead of splitting instances by their features, the algorithm learns a hyperplane that separates them and uses a normalized discounted cumulative gain (nDCG) loss function instead of the Gini index or clustering error [8].

Deep learning based methods have two main types of approaches, sequence to sequence and discriminative learning based methods. From the discriminative learning type, a popular technique is the use of CNN that is able to learn a good text representation, the relation between the text and its corresponding labels might be missed because it cannot create a particular text representation for each label [9]. On the other hand, sequence to sequence methods use recurrent neural network (RNN) to encode an input text and a attentive RNN as decoder to predict labels sequentially. Nevertheless, the way this method predicts the label for XMLC problems is not reasonable because there is no order among each label [9].

2.3 eXtreme Multi-Label Text Classification

This section will be specifically related to a research that have used deep learning approaches to solve eXtreme multi-label text classification problems, however, there are other fields and techniques that have been used in other works and related to other fields.

With the increasing amount of text content being produced daily, the need for automatic classification of such resources has become an area of interest to alleviate the job of manual annotators that do this. Even though these annotators are expert in this job, it can be time consuming and expensive to do this, however, the ability for the human to do this is extremely unique [4]. Out of the many publications related to text classification, it is worth to note [4] research, which compares the performance of semantic subject indexing of scientific papers between using only the title and the full-text data, through the implementation of deep learning. Similar work was done in [2], which also compared the performance of using different parts of scientific papers data (title vs full-text), however, classical classifiers such as K-NN, logistic regression, SVM and Näive Bayes were also tested. From these two previous publications, it is worth to notice the use of neural networks, such as Multi-Layer-Perceptron (MLP) [2], CNN and RNN [2, 4].

In [4], EconBiz and PubMed datasets were used, and as the total number of labels for surpassed 4500, the problem of classifying such documents can be considered XMLTC. From this research, it was proved that for the EconBiz dataset, the title based classifiers reached a better F1-score performance than its counterpart (full-text), with MLP classifier having the best F1-score among all (CNN, LSTM) [4]. It was also noticed that as the number of training title data was increased, the performance of each classifier increased, while at equal size of training data for both showed better results when using the full text [4]. On the other hand, the results using PubMed dataset showed different results when using only the title data, as it did not reach better F1-scores than its counterparts, but most important, it proved that CNN does not necessarily benefit from more training data [4].

3 EXPERIMENT (TASK 3)

As it was mentioned before, the purpose of this work is to see the performance of different classifiers, and techniques under multi-label classification problem. The pipeline for the system can be divided into four stages: preprocessing, vectorization, classification and evaluation.

The dataset that will be used for this experiment is EconBiz, and as the number of available samples is higher than 1 million, two options for running the experiments are available, one using a small portion and other one using the whole dataset. The small portion of the dataset only consists of the documents where the full-text is free to access, please refer to Section 3.1.

For this work, two multi-label approaches will be used: deep learning based and 1-vs-All. For the 1-vs-All approach, two classifier will be tested, Näive Bayes and Logistic Regression. As for deep learning, a CNN and BiLSTM will be used.

The dataset will be split into training and testing set, and the testing set will only be used to evaluate the prediction of each classifier. For the 1-vs-All approach, the evaluation metric will be based on a 10 fold cross-validation.

All the experiments will be tested in CERES [1], which is a Linux based computational clusters, that has a high numbers of processors and GPU, at the same time, the available memory and the ability to submit batch jobs, do not make local machines slow. Primarily, Google Colab was used to test the preprocessing methods and vectorization, however, only a portion of the dataset was used for this, and CERES was then used with the whole dataset.

3.1 Dataset

The dataset "EconBiz"[2] used for this project consists of titles from scientific publication of economics and business studies. The dataset has been filtered by [4], only keeping the titles that have their labels annotated and removing duplicates. The dataset consists of 1,064,634 samples with the title from the scientific document and their annotations (labels). In this dataset, only 70,619 samples have their full-text available for free, and this is why in [4], one of the experiments only uses 70,619 samples to compared the performance of using only title data and full-text. However, for this work, no comparison using full-text will be made, and therefore, the whole dataset will be used. This dataset has 5,661 unique labels, the average number of labels assigned to each sample is 4.4, and each label is assigned to 819.1 samples [4].

3.2 Preprocessing

For all the classifiers, the title data and the labels will be preprocessed. For the title data, non letters, single characters, stop words, multiples spaces will be removed using regular expressions and NLTK stop words [3]. As for the labels, MultiLabelBinarizer() from [6] will be used, in order to create a binary matrix that represents the presence of a label for each sample, this matrix will have the size of the total number of unique labels in the dataset. The labels will be transformed into a binary representation.

3.3 Vectorization

For creating a representation of the title data that the classifiers can understand, this needs to be a vector. Therefore, two different methods will be used: TF-IDF and Keras Tokenizer. TF-IDF will only be used for the 1-vs-All classifiers, and Keras Tokenizer will be used within the deep learning approach.

3.4 Classifiers

3.4.1 Näive Bayes and Logistic Regression. For the Näive Bayes classifier, the structure of the code used for this project is based on the code provided by [4] as baseline when downloading the dataset. However, the code has logistic regression as classifier, and it is only using a portion of the full title dataset. For Näive Bayes classifier, a compliment classifier "ComplementNB" will be used in order to deal with the imbalanced classes in the dataset, as with a regular Näive Bayes classifier, the F1 score was lower. Multi-processing was also added to take advantage of the CERES multiple CPU's.

3.4.2 Deep Learning. Two deep learning models were tested, a CNN[4] and a BiLSTM[5], both models were taken from a Keras example in the documentation. For both models, Keras tokenizer is used to vectorize the title data into a desirable format, the maximum length of each vector is set to be 51. For the CNN model, an embedding layer is added that takes the input vectors and maps them into a different dimension (default 50), a dropout layer of 0.2 is added. Also, a 1-D convolutional layer is added, proceeded by a 1-D global max pooling layer. After that, a vanilla hidden layer is added, with a dropout layer of 0.2 and a "relu" activation function. Finally, the ouput layer of the model consists of the same number of unique labels, and the activation function is sigmoid. It is sigmoid, because it is assumed that the presence of each label in a sample is independent from each other. As for BiLSTM model, an embedding layer was used, followed by a bidirectional layer with dropout of 0.5, and a similar final layer as CNN. Both models were trained using a binary cross entropy loss function, and Adam optimizer with default 0.01 learning rate. The numbers of epochs for each model was set to 2, this was because after this, the difference is loss and accuracy was minimum.

For prediction, the models predicted based on the testing set that was set aside, and the model predicted a vector of size 5661, where each value denoted the probability of a label presence. After searching which probability gave the best results, it was found that the threshold for this was 0.15, which means that all the values that have a probability higher that 0.15, it was assumed that the model predicted the presence of that label, and it was set to 1, if not, it was set to 0.

[1]https://hpc.essex.ac.uk/summaryInformation.php

[2]https://www.kaggle.com/hsrobo/titlebased-semantic-subject-indexing

[3]https://www.nltk.org/

[4]https://keras.io/examples/imdb$_{c}nn/$

[5]https://keras.io/examples/imdb$_{b}idirectional_{l}stm/$

3.5 Evaluation

For the evaluation, the dataset was separated into 80% training and 20% testing. For the 1-vs-All classifiers a 10 fold cross-validation was done, and the mean score among the final scores was reported. For the deep learning based method, cross-validation was not performed, but it would be good to see it. A sample F1 score was use to measure the classifier performance, because it takes into account the precision, and recall in equal proportions, and it is based on the total number of documents that were used for testing.

4 RELATED WORK

Out of the numerous works related to multi-label classification, two works [2, 4] are closely related to this work. Both studies worked under the context of using title data and full-text data for multi-label text classification, and showing the difference in performance between both. The authors tried to prove if title data is sufficient enough to build a strong semantic subject annotation system, due to the limitations of having full-text data because of copy right laws.

In the case of [2], different techniques of vectorization were performed, however, it was found that using a combination of concept frequency and term frequency and then using TF-IDF, or as it was defined "CTF-IDF" was the most appropriate vectorization method. At the same time, many classifier were tested, such as Näive Bayes, Logistic Regression, SVM, Rocchio, KNN, these were used by taking a binary relevance approach and use a decision tree as a meta classifier to predict [2]. Additionally, a Multi-Layer Perceptron was used, which consisted in a fully connected feed-forward neural network with only one hidden layer, this classifier achieved a sample F1 score of 0.472 in an economic dataset, which had 62,924 instances and 4,682 unique labels [2]. The results showed that using title data, instead of full-text can achieved 82% of the F1 score when using full-text, and that eager learns such as MLP outperformed lazy learners for multi-label classification problems.

Moreover, [4]'s work is the most related to this project because of the dataset. At the same time, the context of the problem for [4] lied around the performance of using title data vs full-text, in particular using the EconBiz and PubMed datasets. Compared to [2], in this work only deep learning based methods were used, such as MLP (base and proposed), CNN and LSTM [4]. For the EconBiz dataset, and using all the samples, the proposed MLP was able to outperformed the classifiers that used full-text data, the difference in F1 score was 9.4%, in favour to the title data, however, it has to be considered that the proportion of full-text data and title data is completely different (70,619 vs 1,064,634 respectively), and shows that deep learning models benefit from having large datasets, and if not, there is a possibility that classical classifiers perform better than deep learning methods [4]. Their sample-based F1 scores for the MLP was 0.500, for CNN was 0.426 and for LSTM was 0.466, all of these scores are from using the whole 1,064,634 samples of EconBiz [4].

5 RESULTS

This section presents the F1 scores of the classifiers, some variations have been tested, such as adding bi-grams, tri-grams when vectorizing the title data for Näive Bayes (ComplimentNB).

Classifier	F1 score
Logistic Regression	0.196
Näive Bayes (MultinomialNB)	0.05
Näive Bayes (ComplimentNB)	**0.262**
Näive Bayes (ComplimentNB)*	0.174
Näive Bayes (ComplimentNB)**	0.161
CNN	**0.405**
LSTM	0.292

*Uses uni-grams and bigrams to vectorize the data
** Uses uni-grams, bi-grams and tri-grams.

6 DISCUSSION

From the results, it can be seen that classical methods such as logistic regression and Näive Bayes falt behind the other classifiers. The highest F1 score was from using CNN, a F1 score of 0.405, and our lowest, using Näive Bayes (MultinomialNB), was 0.05. First, the 1-vs-All approach falls behind compared to deep learning based methods, specially when the classifier does not handle imbalance labels distribution in the dataset, this is for the case of MultinomialNB, which the result can be primarily related to it. On the other hand, when using ComplimentNB, the score increases significantly, with 0.262 as the highest. Logistic regression does not fall behind completely, however, it is worth noticing that a reason might be that the iteration to converge was limited up to 100, and there were times where the program gave warnings about this.

On the other hand, the deep learning based methods showed promising results, specially for CNN, which the highest F1 score was 0.405, only 0.021 behind the CNN model presented by [4]. For this work, the vectorizer method was using a count vector based on the frequency of each word based on a corpus, while in [4] used word embeddings, that were taken from a pretrained model that was fine tuned during training. It is possible that with a different vectorizer method, the score for the CNN model for this work increases. Moreover, for the BiLSTM model, the highest score was 0.292, and in comparison to [4], our score clearly fall behind, however, the vectorization methods can be a reason for this, and the structure of the model as well.

7 CONCLUSION

In retrospect, multi-label classification has been considered a standard task in text analytics, and many techniques have been developed to tackle problems like this, such a problem transformation, algorithm adaptation, and ensemble methods. Deep learning has shown that these algorithms have the ability to improve performance over classical classifiers with reasonable computational resources, compared to 1-vs-All approach. Morever, deep learning methods have shown their capability to take advantage of larger datasets by showing improvements in their accuracy at classifying. Nonetheless, with the high explosion of data, new approaches to solve multi-label classification problems have been developed to cope with the high number of possible labels, that can reach millions of possible tags in certain fields. To cope with eXtreme multi-label classification, embedding based methods, label tree based methods and deep learning have resulted in useful techniques to tackle this problems.

For the classifier, it was found that deep learning based methods are strong classifiers and can benefit from the high amount of data, specially for CNN, as it was the highest score (0.405), and was almost at the same score compared to [4] CNN model. A very basic vectorizer method was considered for this work, and the possibility to use other methods such as word2vec that are able to take into account the context of the word will might increase the performance of the classifiers. At the same time, die to the vectorizers used, the dimension of the vectors are high, and might impact the performance of the classifier due to sparsity problems, and this might be solved using feature selection, such as PCA, to reduce the dimensionality of the vectors.

REFERENCES

[1] BK Bhavitha, Anisha P Rodrigues, and Niranjan N Chiplunkar. 2017. Comparative study of machine learning techniques in sentimental analysis. In *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*. IEEE, 216–221. https://doi.org/10.1109/ICICCT.2017.7975191

[2] 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*. 1–4. https://doi.org/10.1145/3148011.3148039

[3] Kwan Hui Lim, Shanika Karunasekera, and Aaron Harwood. 2017. Clustop: A clustering-based topic modelling algorithm for twitter using word networks. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2009–2018. https://doi.org/10.1109/BigData.2017.8258147

[4] 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*. 169–178. https://doi.org/10.1145/3197026.3197039

[5] Md Aslam Parwez, Muhammad Abulaish, et al. 2019. Multi-label classification of microblogging texts using convolution neural network. *IEEE Access* 7 (2019), 68678–68691. https://doi.org/10.1109/ACCESS.2019.2919494

[6] 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.

[7] Obioma Pelka, Christophe M Friedrich, A García Seco de Herrera, and Henning Müller. 2019. Overview of the ImageCLEFmed 2019 concept detection task. *CLEF working notes, CEUR* (2019).

[8] 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*. 263–272.

[9] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. In *Advances in Neural Information Processing Systems*. 5812–5822.

[10] 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*. 100–107.