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

School of Computer Science
and Electronic Engineering
University of Essex
CE807-7-SU-CO : Text
Analytics
registration number:2110366

ABSTRACT
Humans produce exabytes of data every day, which has increased the solution for the multi-label learning approaches big data has brought us. In addition, extreme multi-label classification, which deals with classification tasks with extremely large numbers of classes or labels, is an active and quickly expanding research area; using massive data with restricted supervision to build a multi-label classification model becomes useful for practical applications, etc.

Author Keywords
XMLT, Deep Learning, Text analysis

CCS Concepts
•**EXTREME MULTI-LABEL LEARNING** → *embedding approaches; Tree-based approaches; One-vs-all approaches;*
•**DEEP LEARNING FOR MULTI-LABEL LEARNING** → *Deep Embedding Methods for MULTI-LABEL Classification;*

1 INTRODUCTION (TASK 1)

It is crucial to use MULTI-LABEL classification (MLC), which simultaneously assigns several labels to each instance, in various fields, including the automatic classification of images and the classification of documents and protein functions. With today's large and complicated data structures, traditional multi-label classification algorithms are unable to meet the growing demands of the data. New multi-label learning paradigms are urgently needed as a result, and new trends are emerging. Extreme multi-label classification (XMLC), a new field of study that focuses on multi-label issues with an exceptionally huge number of labels, is emerging with the arrival of the big data era. It can help to formulate many difficult applications such as multi-label classification tasks with millions or even billions of labels, including web page categorization, gene function prediction, language modeling, and picture or video annotation. Due to the expensive computing cost and the abundance of labels, the MLC problem can not fix the same as current XMLC approaches. How to accurately estimate all of the positive labels to testing instances is a significant challenge in XMLC. Due to the current rapid growth of data volume, getting comprehensive monitoring frequently requires a substantial financial and time investment. Additionally, several effective models based on graphs, embeddings[49], [43], and [36], probability models[17], [23], and so forth have been proposed[37], [41], and [14]. The first approach to applying neural network (NN) architecture to MLC problems is P-MLL[50]. The first Deep NN (DNN) based embedding technique for MLC issues is Canonical Correlated AutoEncoder (C2AE)[45]. The Challenging MLC problems have also led to the development of various deep learning techniques, such as Extreme MLC[21], [48], and [40], partial and weakly supervised MLC[14], [23], and [10], and MLC with unseen labels[42] and [20]. The study of recently developed deep learning architectures[7], [27], [26], and [44]. The main challenge for MLC operations is caused by the Web's ongoing production of quintillion bytes of streaming data each day. First, because they involve storing all data sets in memory, the current offline MLC algorithms are unsuitable for streaming data sets. Second, it is difficult to apply off-line multi-label techniques to the sequential data. Because of this, a number of methods for online multi-label categorization have lately been put forth, including[29], [39], and [11]. But the current experimental and theoretical findings are both unsatisfactory and very constrained. Credible research on online multi-label learning is desperately needed. Numerous studies[51] and [22] have demonstrated that multi-label learning techniques that explicitly reflect label dependency will typically produce higher prediction accuracy. Modeling the label dependency has thus become one of the main points in multi-label classification problems over the past few years. Many different techniques have been developed to model dependence. For instance, the classifier chain (CC) model[32] captures label dependency by using binary label predictions as additional input attributes for the subsequent classifiers in a chain. Canonical correlation analysis(CCA)[52] is used to train label dependency. The label and feature dependencies are both captured by CPLST[6] using principal component analysis.

2 RELATED WORK (TASK 1)

2.1 EXTREME MULTI-LABEL LEARNING

In order to automatically annotate a data point with the most pertinent subset of labels from an extremely large number of labels, a classifier must be learned in extreme multi-label classification (XMLC), which has opened up a new research area in data mining and machine learning. A classifier that can identify the people in a figure might be built, for instance, using the millions of selfies that people upload to Facebook every day. A wide variety of fields, including language modeling,

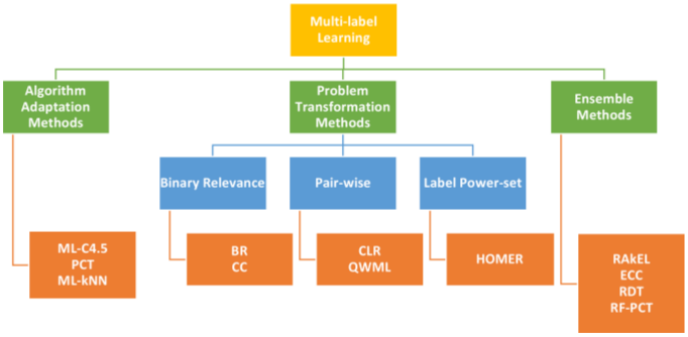

Figure 1. Extreme multi-label learning[11].

document classification, face recognition, and gene function prediction, have seen the use of XMLC in numerous applications. The primary difficulty with XMLC is that it learns using hundreds of thousands or even millions of labels, features, and training points. Modern XMLC techniques are heavily reliant on embeddings, trees, and one-vs-all classifiers to solve this problem.

2.1.1 Embedding approaches

It is clear that label vectors have little support when dealing with a large number of labels[13]. In other words, it is possible to encode data by projecting each label vector into a lower-dimensional compressed label space. Following that, each compressed label's regression is learned. The labels from each testing instance's regression outputs are then decoded using the compressed sensing technique. Recently, a lot of embedding-based works have been created in this learning paradigm. Among these works, canonical correlation analysis (CCA)[52] and bloom filters[8] are two compression and decompression techniques that are different from one another. SLEEC is one of the foundational embedding techniques in XMLC because of its clarity and encouraging experimental findings[3]. A novel graph embedding technique based on the k-nearest neighbor (KNNG) graph is also demonstrated in AnnexML[38]. The goal of AnnexML is to build the KNNG of label vectors in the embedding space to increase the k-nearest neighbor classifier's prediction accuracy and processing speed. These representative vectors are then created, and DEFRAG uses hierarchical clustering to produce feature clusters, which is followed by agglomeration, which is accomplished by adding the coordinates of the feature vectors within a cluster.[18] demonstrates that DEFRAG provides quicker and superior performance. The embedding matrix used by the existing embedding approaches is in real space. As a result, we must use regressors for training, which may require tackling pricey optimization issues. Numerous references use coding techniques to effectively train the model to overcome this restriction. For instance,[8] designs a straightforward method to choose the k representative bits for labels for training and suggests a reliable decoding algorithm for prediction, both of which are based on Bloom filters[4], a well-known space-efficient randomized data structure designed for approximate membership testing. But Bloom filters might produce a lot of false positives. The most widely used methods for dealing with XMLC are embedding techniques. Among them, SLEEC is a seminal work that is suggested for beginners to try. The main drawback of existing embedding techniques is the disregard for correlations between input and output, which results in poorly aligned learned embeddings and decreased prediction accuracy. Future research will focus on how to construct an embedding space that can maintain the relationships between input and output. The complexity of the training and testing times is another drawback of current embedding techniques. The training and testing process might be sped up using some methods, including parallelization, hashing, and random projection.

2.1.2 Tree-based approaches

In hierarchically partitioning the instance set or the label set, tree-based methods can break the original large-scale problem into a series of smaller-scale subproblems. The set is initially initialized in the root node. The next step is optimizing a partitioning formulation to divide the set in a node into a predetermined number of subsets connected to child nodes. Nodes are broken down iteratively until a stopping condition is verified on the subsets. Optimizing the partition criterion and creating a condition or classifier on the feature space to determine which child node an instance belongs to are the two optimization problems that each node faces. An instance is sent from one leaf (instance tree) or numerous leaves down the tree during the prediction phase (label tree). The projected labels are found in a label tree's reaching leaves. A classifier trained using the instances in the leaf node predicts the future for an instance tree. Consequently, the primary benefit of tree-based approaches is that, if the tree is balanced, the prediction costs are sub-linear or even logarithmic. The ranking loss function normalized Discounted Cumulative Gain, and FastXML[31] are shown to understand the hierarchy (nDCG). For XMLC, nDCG offers two key advantages. First of all, since nDCG is a measurement sensitive to relevance and ranking, it assures that the most highly ranked positive labels are projected for the relevant positive labels. The Gini index or the clustering error, which are rank-insensitive measurements, cannot provide this assurance. Second, because nDCG is rank sensitive, it can be optimized across all labels at the present node, preventing local optimization from being too narrow. The results of the trials demonstrate that nDCG performs better in extreme multi-label learning. PfastreXML[16], a FastXML-based study, investigates ways to raise tail label prediction accuracy. A power law distribution is followed by the labels in XMLC. Labels that appear less frequently typically convey more information but have less training data and are more difficult to predict. By substituting the nDCG loss with its unbiased propensity-scored equivalent, PfastXML enhances FastXML and offers greater incentives for correctly predicting tail labels. Additionally, PfastreXML uses tail-label classifiers to rerank PfastXML's predictions[16] demonstrates that PfastreXML successfully adapts to tagging, recommendation, and ranking tasks and offers promising performance in predicting tail labels. SwiftXML prabhu2018extreme keeps all of PfastreXML's scaling properties while increasing PfastreXML's prediction precision by taking into account more information about exposed item preferences and item traits. By optimizing

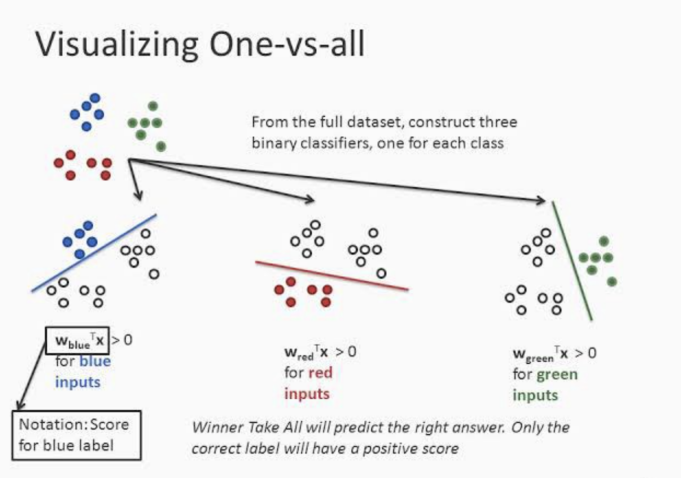

Figure 2. One vs all[18].

two separating hyperplanes in the user and item feature spaces, respectively, SwiftXML provides a novel node partitioning method. Studies on Wikipedia tagging and item-to-item recommendations on Amazon show that SwiftXML is 14 percent more accurate than top extreme classifiers. Ranking-based metrics like nDCG and its derivatives have been examined by FastXML, PfastreXML, and SwiftXML. Recent research by jasinska2016extreme focuses on the F-measure, a performance metric that is frequently used in multi-label classification and other disciplines including information retrieval and natural language processing. A unique sparse probability estimate (SPE) is suggested in jasinska2016extreme to simplify the threshold adjustment process in XMLC. Then, using SPEs, they create three algorithms for employing the Empirical Utility Maximization (EUM) framework to maximize the F measure. Additionally, FastXML and Probabilistic label trees (PLTs) are presented for computing SPEs. The effective approach for dealing with XMLC with the logarithmic dependence on the number of labels is to use tree-based approaches. FastXML is a well-liked approach that experts advise using. The fact that tree-based approaches like FastXML, PfastreXML, and SwiftXML involve difficult non-convex optimization issues at every node, however, is one of their main drawbacks. Future studies will focus on how to create a scalable, affordable tree structure. For instance, the gradient boosted decision trees (GBDT) for XMLC are studied in GBDT-SPARSE[34]. Each node first projects the feature into a low-dimensional space, after which a straightforward inexact search method is employed to locate a good split. To make GBDT acceptable for XMLC, they considerably cut the training and prediction times as well as the model size. The random forest method and quick partitioning techniques are attempted by CRAFTML[35]. The feature and label are initially randomly projected onto lower-dimensional spaces by CRAFTML. The cases are then divided into k temporary subsets using the projected labels and the k-means algorithm.

2.1.3 One-vs-all approaches

One of the most widely used approaches for multi-label classification is one-vs-all (OVA) methods, which separately train a binary classifier for each label. However, this method has two significant drawbacks for XMLC. Firstly, utilizing commercial solutions like Liblinear to train one-vs-all classifiers for XMLC problems may be computationally and memory impractical. Secondly, sluggish prediction is caused by the potential for a very high model size for the XMLC data set. Numerous efforts have recently been created to address the aforementioned problems with the one-vs-all approaches in XMLC. Some sub-linear algorithms are presented to adapt one-vs-all methods in the extreme classification context by making use of the data's sparsity. For instance, PD-Sparse[47] suggests that in an Empirical Risk Minimization (ERM) architecture for XMLC, the separation ranking loss and penalty be minimized. The highest response from the collection of negative labels minus the lowest response from the set of positive labels are used by the separation ranking loss to punish the prediction on an instance. PD-Sparse outperforms SLEEC, FastXML, and certain other XMLC algorithms in correctness while obtaining an extremely sparse solution both in primal and in dual with a sub-linear time cost. PPDSparse[46] scales down the training by parallelizing PD-Sparse using sub-linear methods by introducing separable loss functions because the training for each label is separated in PPDSparse, and the memory cost of PDSparse can be drastically decreased. In order to scale one-vs-all solutions for issues containing hundreds of thousands of labels, DiSMEC[1] also proposes a sparse model with a parameter thresholding mechanism. ProXML[2] indicates that the minimizing one-vs-all strategy based on Hamming loss works well for tail-label prediction in XMLC based on the graph theory and suggests using regularised Hamming loss to address the tail label concerns. High prediction accuracy and small model sizes have been attained using PD-Sparse, PPDSparse, DiSMEC, and ProXML. To determine whether a label is significant or not, they continue to train a separate linear classifier for each label and linearly scan every single label. As a result, these approaches' training and testing costs increase linearly as the number of labels increases. To deal with this problem, some cutting-edge techniques are offered. For instance, [28] suggests predicting on a restricted set of labels, which is formed by projecting a test instance on a filtering line, and maintaining only the labels that have training instances close to this projection, in order to lower the linear prediction cost of one-vs-all approaches. The candidate label collection should contain the majority of the testing instances' actual labels and be as condensed as possible. By maximizing these two concepts as a mixed integer problem, they train the label filters. The label filters can produce equivalent prediction accuracy while reducing the testing time of conventional XMLC classifiers by orders of magnitude. In contrast to reading through a very big list of labels, [28] demonstrates an intriguing method for finding a small number of potentially relevant labels. The question of how to employ label filters to shorten training time remains unanswered. Through the learning of balanced binary label trees based on an effective and informative label representation, Parabel[30] shortens the training time of one-vs-all approaches. In order to apply hierarchical softmax to the multi-label context, they also provide a probabilistic hierarchical multi-label model. For handling XMLC, the logarithmic prediction technique is also suggested.

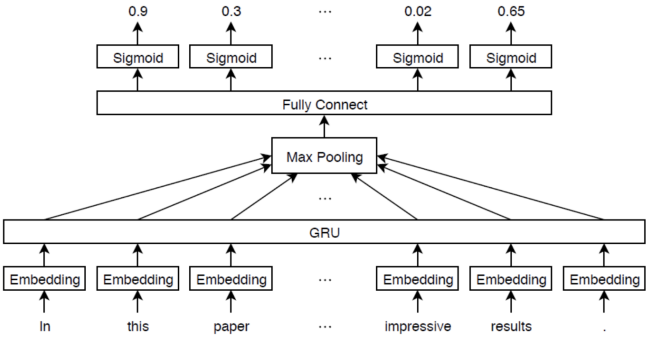

Figure 3. Multilabel Text Classification Using Deep Learning [35].

In comparison to the most advanced one-vs-all extreme classifiers, experiments suggest that Parabel could be orders of magnitude faster in training and prediction. However, because Parabel cannot ensure that labels with a similar meaning are placed in the same group and because the inaccuracy will spread across the deep trees, it is inaccurate for low-dimension data sets. Bonsai[19] illustrates a shallow label tree structure with generalized label representation to limit error propagation. Slice[15] also presents a novel negative sampling strategy to raise the accuracy of predictions for low-dimensional packed feature representations. Slice scales to 100 million labels and 240 million training points, and is up to 15percent more accurate than Parabel. The tests in [15] demonstrate that negative sampling in XMLC is a potent tool. One-vs-all methods are straightforward approaches to working with XMLC, and for newcomers, PD-Sparse is the first option to try. For XMLC, computation and memory cost offer an insurmountable problem because one-vs-all approaches independently train a binary classifier for each label and do not take label correlations into account. Although the methods examined in this part can help with the calculation issue, it may become increasingly difficult to employ label correlations to improve the performance of one-vs-all methods. Designing some one-vs-all learning models that take different label correlations into account is one approach.

2.2 DEEP LEARNING FOR MULTI-LABEL LEARNING

Deep learning has attained cutting-edge performance in numerous real-world multi-label applications, such as multi-label image classification, thanks to its robust learning capability. The benefit of deep learning must be fully utilized in MLC issues in order to more accurately capture label dependencies.

2.2.1 Deep Embedding Methods for Multi-Label Classification

The most popular deep approaches for MLC are deep embedding techniques. Among these, C2AE is a deep embedding pioneer for MLC and has been used in many practical applications, such as multi-label emotion classification, which merits investigation for novices to understand the fundamental mechanisms. Label correlation is essential for MLC, as is well known, and several objectives, such as [38], have been used to simulate label correlation in deep embedding techniques for MLC.

2.2.2 Deep Learning for eXtreme Multi-Label Classification

Multi-label learning is frequently difficult in real-world applications because labels must be set up in a sophisticated way. For instance, the XMLC standard has a relatively high number of labels, many of which are only partially or weakly specified, and numerous labels that have never been seen before. Convolutional neural networks (CNN) and dynamic pooling are employed in XMLCNN[21] to learn the text representation, and computational efficiency is achieved by using a hidden bottleneck layer that is significantly smaller than the output layer. The inability of XML-CNN to accurately capture the significant subtext for each label continues to be a problem. AttentionXML[48], which has two distinct properties, is suggested as a solution to this problem. The first is a multi-label attention mechanism that takes raw text as input and may capture the text that is most pertinent to each label. The second option is a shallow and wide probabilistic label tree (PLT), which can hold millions of labels, particularly "tail labels." The Ranking-based Auto-Encoder (Rank-AE)[40] is a novel deep embedding technique for XMLC that is based on C2AE. in order to construct a margin-based ranking loss that is more efficient for XMLC and noisy labels, Rank-AE first employs an efficient attention technique to learn rich representations from any sort of input characteristics. Next, it learns a latent space for instances and labels. The DNN-based embedding approaches for XMLC perform poorly due to overfitting, as empirically shown in [12]. In light of this discovery, [12] further suggests a new regularise, GLaS, for embedding-based neural network techniques. improves the feature representation of a pretrained deep transformer. For XMLC, they suggest a brand-new label clustering model, and the transformer acts as a neural matcher. On a number of frequently used extreme data sets, the proposed approaches achieve state-of-the-art performance[5]. Four sub-tasks, including intermediate representation, negative sampling, transfer learning, and classifier learning, have just been added to the DeepXML framework, which can produce a family of algorithms[9]. On publicly available short text data sets, it produces Accelerated Short Text Extreme Classifier (Astec), which is more precise and quick than cutting-edge deepXMLs. DECAF takes into account label metadata, such as textual explanations of labels, which are useful but sometimes disregarded by current approaches. With millions of labels, DECAF accurately predicts using a model that is jointly learned and enhanced by feature representation and label metadata[24]. In order to train a model with a label correlation graph made up of millions of labels, ECLARE includes label text and label correlations and creates a cheap architecture and scalable techniques[25]. Similar to this, GalaXC uses shared document-label graphs to cooperatively learn from a variety of sources, including label metedata[33]. GalaXC also adds label-wise attention to produce extremely powerful extreme classifiers. demonstrates that, while training and making predictions on benchmark data sets, GalaXC is up to 18 percent more accurate than state-of-the-art[33].

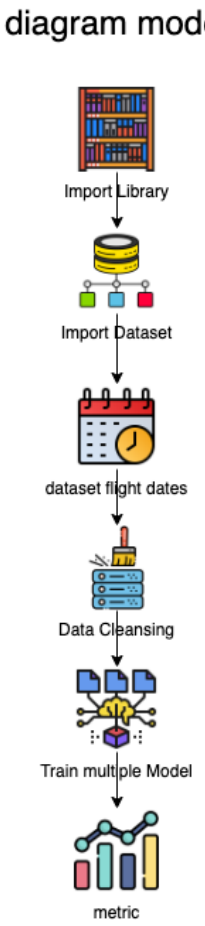

Figure 4. Diagram project

3 DATASET

In this research, we have used datasets from digital libraries that collect the English dataset are EconBiz and PubMed.

3.1 Econbiz

The Econbiz dataset is a dataset released from July 2017 by the ZBW - Leibniz Information Center for Economics that collects English-language publications and man-made labels from Standard Thesaurus Wirtschaft (STW). The total number of labels was 6000 labels. Duplicate publications were removed by examining the same labels and titles. The total data collected was approximately 1,064,000. publications.

3.2 PubMed

PubMed dataset This is an English language dataset related to index of biomedical articles on large-scale semantic subjects. Duplicate data has been removed by looking at the same title and labels, approximately 12.8 million publications, it has label MeSH terms, 28,000 publications, all data collected from the training set from the BioASQ challenge.

4 METHODOLOGY

This research conducted experiments using machine learning and deep learning related to labels and text, with a total of 2 datasets, Econbiz and PubMed. With a total of 12 models, we will compare all 12 models and find the best model of each dataset.

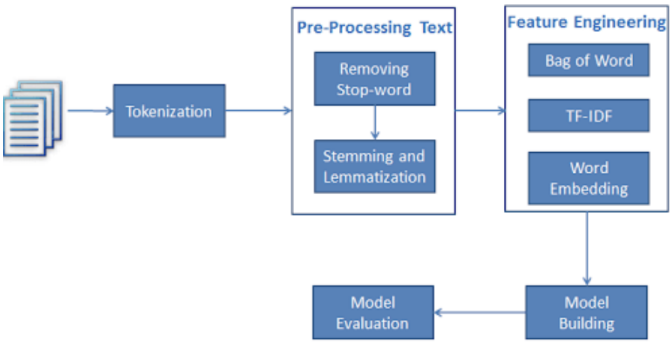

Figure 5. Clean data in text[25]

4.1 Data pre processing

The preliminary survey has a total of 1064634 records and is in English. Although there are no Null values at all but if we look closely we can see that the data is quite unclean. If the text is not clean it will affect the decreased accuracy of the model. This is why it is important that we clean the data. In the first step we need to change all uppercase letter in the title column to lowercase letters. After that, we will do the Stop Words step, where we will remove the words that we see often in sentences or documents, but that does not help to convey much. This allows us to remove those words from the vocabulary list. Filter it out of the document like a, an, the, also, just, quite, unless, etc. Then we will do Lemmatization. which is the process of converting a word with a list of words in a dictionary, proper grammatical analysis of the language to transform and conjugate words to eliminate inflection of words such as gender, tense, sound, mood, number, etc. Most will cut the footer. Leave only the basic form, which is a word in the dictionary called Lemma. For example, saw with Stemming works best with s, but with Lemmatization can work see or saw, depending on whether it is a Noun or Verb. After that we will do Stemming. It cuts the end of the word coarsely with a simple pattern, which works quite well. For most, but not all, English words, stemming reduces form. Leave only the front part of the same words in the same group of words. After that, we will delete numbers, single characters, and multiple spaces with regular expressions because in NLP we often have to deal with String Format, which is a quick method. If we have to write a program, loop, check every case by ourselves. The program is very complicated and slow. After we clean the title column for both datasets. We will also need to clean the label column. By making one-hot encoder using MultiLabelBinarizer, create dummy features to make it a multilabel for use in model training.

4.2 feature selection

we will give title column a feature that acts as inputs that are passed into machine learning and deep learning models, which are characteristics of sample data as descriptions to help Machine learning models predict labels precisely. As for labels, we will use labels columns that are made One-Hot

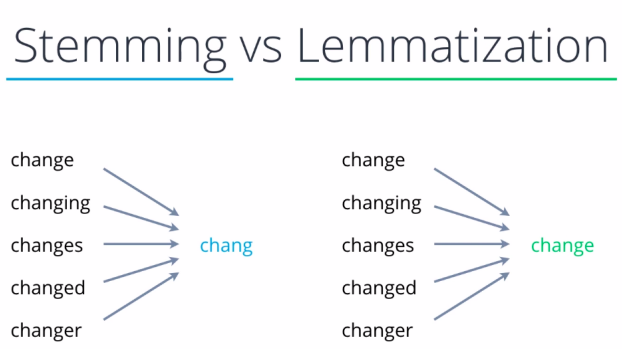

Figure 6. Lemmatization and Stemming[27]

Encoder with MultiLabelBinarizer and then as a label that acts as a solution in the prediction model.

4.3 Train test split

After we have clean data for both datasets in all columns. We will need to split the data with train test split afterwards to make the data not overfitting. We will make to divide the data set Training set 80 percent and test set 20 percent. Training Set is used to feed the model to use the training.

4.4 Train Model

We will train a model to find the best model of both datasets with 9 machine learning models: Logistic Regression, Random Forest, Naive Bayes, Decision Tree, K-nearest neighbors, SVC, XGBoost, Stochastic Gradient Descent, Extra Trees and Deep Learning 3 models: Multilayer Perceptron, Convolution Neural Network, Recurrent Neural Network.

4.4.1 Machine learning model

which follows the normal process of modeling. Modeling each cycle They are all different with the subject of information Assumption setting, so the code from the previous project Therefore, it cannot be used in new project, but if there is a clear pipeline, it can greatly reduce the workflow, helping to make the machine learning model more convenient. inside the pipeline, there is TfidfVectorizer, TF stands for Term Frequency, which counts how often a word is specified in a document, and idf reduces the size of words. That appears a lot in the document, so TF-IDF stands for The word frequency score attempts to highlight words that are commonly seen in the document. and the function TfidfVectorizer It generates tokens for documents, learns vocabulary and calculates word weights in addition. The OneVsRestClassifier function is very useful when we want to multiclass or classify a multilabel with a model and parameters that we can predict.

4.4.2 Deep Learning model

The deep Learning method is quite different from Machine Learning. In the first step, we will define the Class Sequential variable to be in Instance object. Then we will use .add to create layers one by one, from left to right, by specifying an argument that corresponds to the layer structure we designed. After that we will compile the model according to the structure we have defined. Then start training by calling Method.As

Mathematically,

$$\text{Hamming Loss} = \frac{1}{nL}\sum_{i=1}^{n}\sum_{j=1}^{L}\left[I(y_j^{(i)} \neq \hat{y}_j^{(i)})\right]$$

Where,

$n \implies$ Number of training examples

$y_j^{(i)} \implies$ true labels for the ith training example and jth class

$\hat{y}_j^{(i)} \implies$ predicted labels for the ith training example and jth class

Figure 7. hamming loss calculation[37]

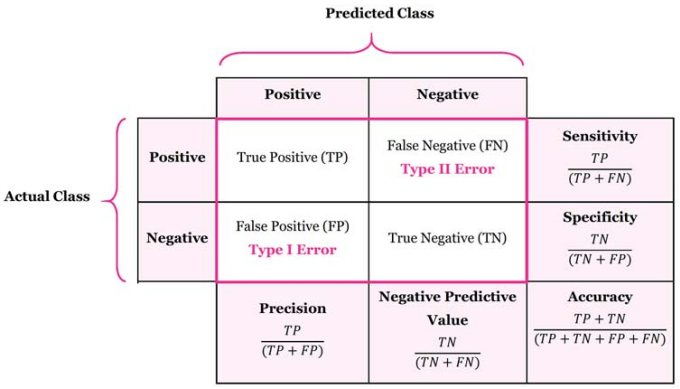

Figure 8. accuracy,recall,precision,f1-score confusion matrix [37]

well as define various arguments which contain epochs, the number of training cycles, each epoch decreases Loss, while Accuracy increases the batch size, the size of the Batch, which is the number of data items that the Optimiser will calculate at one time. Directly affects the calculation speed, the larger the Batch, the faster the calculation, but the Batch is too big may cause the calculated data to be larger than the memory of the machine we can support. validation split We can keep part of the train set as a validation set, which is a data set that the model never sees. Keras will calculate the Loss and Accuracy with the Validation set every time it ends 1 Epoch and display it to us. Incidentally, the Validation set will remove the last part of the Train set without random. Therefore, we should randomly shuffle the data in the first place.

4.5 Metric

In this project, there are 2 that we choose to measure the performance of the model namely. Firstly, F1 Score, which is used as the main value in evaluating the performance of each model because F1 Score will focus on both precision and recall values, therefore it is likely to be suitable for models that are multilabel and have imbalance data problems than using traditional Accuracy Score. Generic Classification Model. We also use Hamming loss, where hamming loss is a calculated value of a misclassified label whose lower value shows excellent classification efficiency.

5 RESULT

from all experiments12 The models are Logistic Regression, Random Forest, Naive Bayes, Decision Tree, K-nearest neighbors, SVC, XGBoost, Stochastic Gradient Descent, Extra

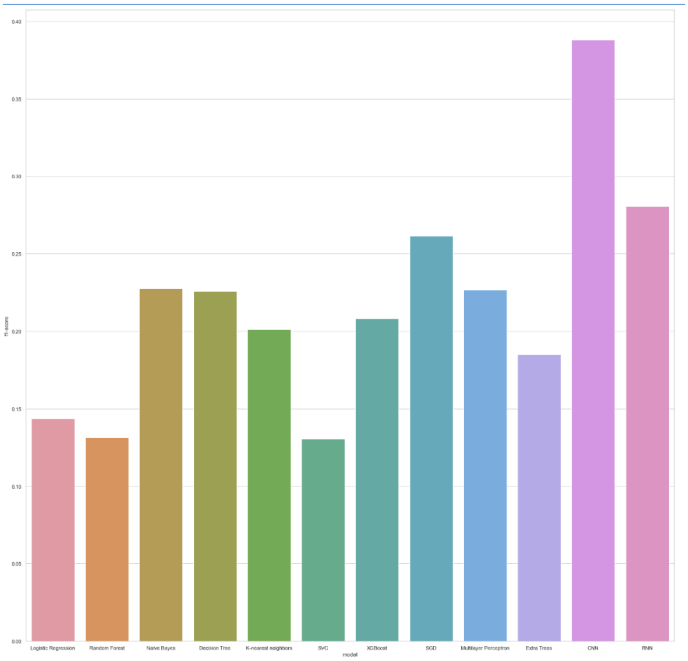

Figure 9. visualization for f1 score with econbiz dataset

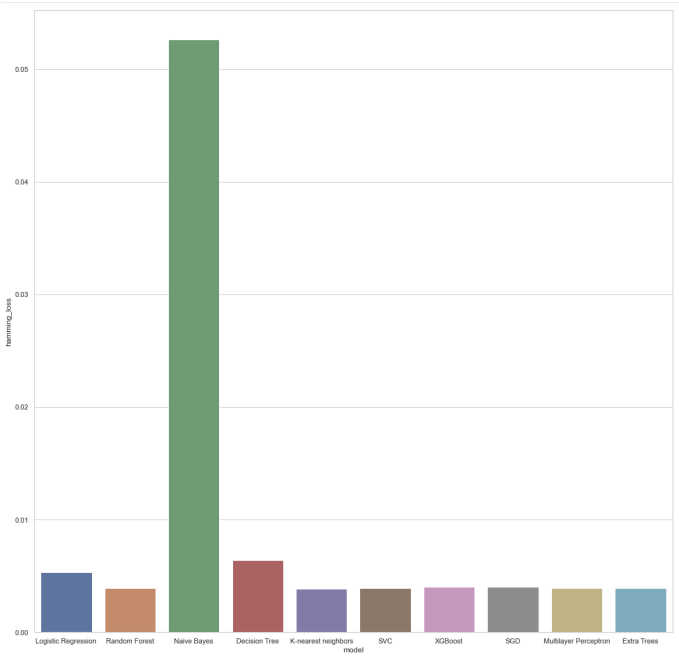

Figure 10. visualization for hamming loss with econbiz dataset

Trees Deep Learning 3 models are Multilayer Perceptron, Convolution Neural Network, Recurrent Neural Network. In Dataset econbiz model Convolution. Neural Network has the highest f1-score at 38.79 by models with f1-score followed by Recurrent Neural Network(28.0), Stochastic Gradient Descent(26), Naive Bayes(0.227), Multilayer Perceptron(22.6), Decision Tree(22.5), XGBoost(20.8027), K-nearest neighbors(20.1), Extra Trees(18.5), Logistic Regression(14.3), Random Forest(13.1), SVC(0.130). while the lowest hamming loss is K-nearest neighbors with the hamming value. loss at 0.003861758 Followed by SVC(0.003884297520661157), Extra Trees(0.00389181), Multilayer Perceptron(0.00389181), Random Forest(0.0039293), XGBoost(0.00399699), Stochastic Gradient Descent(0.0040120), Logistic Regression(0.0053418482) and ,Naive Bayes(0.0526070). While PubMed dataset model Naive Bayes had the highest f1 score at 0.240786 followed by Decision Tree(0.239597), Stochastic Gradient Descent(0.23466), Convolution Neural Network(0.2295436), Multilayer Perceptron(0.2282401284), Logistic Regression(0.208485), XGBoost(0.20349), K-nearest neighbors(0.188369),LSTM(0.17925790459540464), Random Forest(0.13176008), SVC(0.079303), Extra Trees(0.03451559934318555). While the hamming loss Multilayer Perceptron was considered low. The next more than were Random Forest(0.0057863), Stochastic Gradient Descent(0.00587), SVC(0.00588519), XGBoost(0.0058972), K-nearest neighbors(0.00592136), Naive Bayes(0.0071587071), Decision Tree(0.008919440), Logistic Regression(0.009674), Extra Trees(0.018336314847942754).

6 CONCLUSION

From all comparisons it is clear that in the dataset, the most suitable f1-score Econbiz dataset model is the Convolution Neural Network with 38.79 percent accuracy, while Stochastic Gradient Descent is a machine learning model. The most accurate at 26.15 percent, although the hamming loss is higher than other models, but it's not much. And it's not much different from other models. While Naive Bayes, despite having a good F1-score at 22.76 percent, hamming loss is very high and does not should to use. While PubMed dataset most attractive model is model Naive Bayes with an f1 score of 0.24078656552816655. Hamming loss is slightly higher than average but is considered slightly higher, while the most useless model is Extra Trees with precision are at 0.03451559934318555 and the loss is 0.018336314847942754

REFERENCES

[1] 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*. 721–729.

[2] Rohit Babbar and Bernhard Schölkopf. 2019. Data scarcity, robustness and extreme multi-label classification. *Machine Learning* 108, 8 (2019), 1329–1351.

[3] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems* 28 (2015).

[4] Burton H Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (1970), 422–426.

[5] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. 2020. Taming pretrained transformers for extreme multi-label text classification.

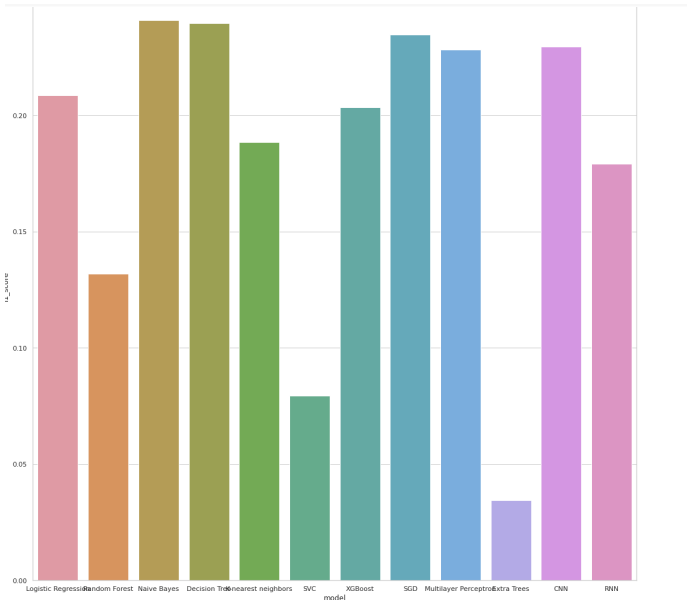

Figure 11. visualization for f1 score with pubmed dataset

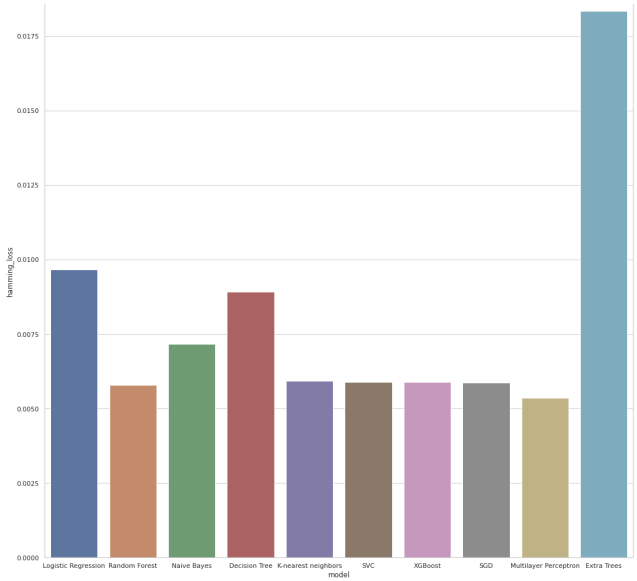

Figure 12. visualization for hamming loss with pubmed dataset

In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3163–3171.

[6] Yao-Nan Chen and Hsuan-Tien Lin. 2012. Feature-aware label space dimension reduction for multi-label classification. *Advances in neural information processing systems* 25 (2012).

[7] Moustapha Cissé, Maruan Al-Shedivat, and Samy Bengio. 2016. Adios: Architectures deep in output space. In *International Conference on Machine Learning*. PMLR, 2770–2779.

[8] Moustapha M Cisse, Nicolas Usunier, Thierry Artieres, and Patrick Gallinari. 2013. Robust bloom filters for large multilabel classification tasks. *Advances in neural information processing systems* 26 (2013).

[9] Kunal Dahiya, Deepak Saini, Anshul Mittal, Ankush Shaw, Kushal Dave, Akshay Soni, Himanshu Jain, Sumeet Agarwal, and Manik Varma. 2021. Deepxml: A deep extreme multi-label learning framework applied to short text documents. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 31–39.

[10] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. 2019. Learning a deep convnet for multi-label classification with partial labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 647–657.

[11] Xiuwen Gong, Dong Yuan, and Wei Bao. 2020. Online metric learning for multi-label classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 4012–4019.

[12] Chuan Guo, Ali Mousavi, Xiang Wu, Daniel N Holtmann-Rice, Satyen Kale, Sashank Reddi, and Sanjiv Kumar. 2019. Breaking the glass ceiling for embedding-based classifiers for large output spaces. *Advances in Neural Information Processing Systems* 32 (2019).

[13] Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. 2009. Multi-label prediction via compressed sensing. *Advances in neural information processing systems* 22 (2009).

[14] Dat Huynh and Ehsan Elhamifar. 2020. Interactive multi-label cnn learning with partial labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9423–9432.

[15] 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*. 528–536.

[16] 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*. 935–944.

[17] Vikas Jain, Nirbhay Modhe, and Piyush Rai. 2017. Scalable generative models for multi-label learning with missing labels. In *International Conference on Machine Learning*. PMLR, 1636–1644.

[18] Ankit Jalan and Purushottam Kar. 2019. Accelerating extreme classification via adaptive feature agglomeration. *arXiv preprint arXiv:1905.11769* (2019).

[19] Sujay Khandagale, Han Xiao, and Rohit Babbar. 2020. Bonsai: diverse and shallow trees for extreme

multi-label classification. *Machine Learning* 109, 11 (2020), 2099–2119.

[20] Chung-Wei Lee, Wei Fang, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. 2018. Multi-label zero-shot learning with structured knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1576–1585.

[21] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017a. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 115–124.

[22] Weiwei Liu, Ivor W Tsang, and Klaus-Robert Müller. 2017b. An easy-to-hard learning paradigm for multiple classes and multiple labels. *The Journal of Machine Learning Research* 18 (2017).

[23] Weiwei Liu, Haobo Wang, Xiaobo Shen, and Ivor Tsang. 2021. The emerging trends of multi-label learning. *IEEE transactions on pattern analysis and machine intelligence* (2021).

[24] Anshul Mittal, Kunal Dahiya, Sheshansh Agrawal, Deepak Saini, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2021a. Decaf: Deep extreme classification with label features. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 49–57.

[25] Anshul Mittal, Noveen Sachdeva, Sheshansh Agrawal, Sumeet Agarwal, Purushottam Kar, and Manik Varma. 2021b. ECLARE: Extreme classification with label graph correlations. In *Proceedings of the Web Conference 2021*. 3721–3732.

[26] Jinseok Nam, Young-Bum Kim, Eneldo Loza Mencia, Sunghyun Park, Ruhi Sarikaya, and Johannes Fürnkranz. 2019. Learning context-dependent label permutations for multi-label classification. In *International Conference on Machine Learning*. PMLR, 4733–4742.

[27] Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. *Advances in neural information processing systems* 30 (2017).

[28] Alexandru Niculescu-Mizil and Ehsan Abbasnejad. 2017. Label filters for large scale multilabel classification. In *Artificial intelligence and statistics*. PMLR, 1448–1457.

[29] Sunho Park and Seungjin Choi. 2013. Online multi-label learning with accelerated nonsmooth stochastic gradient descent. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 3322–3326.

[30] 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*. 993–1002.

[31] 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.

[32] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning* 85, 3 (2011), 333–359.

[33] Deepak Saini, Arnav Kumar Jain, Kushal Dave, Jian Jiao, Amit Singh, Ruofei Zhang, and Manik Varma. 2021. GalaXC: Graph neural networks with labelwise attention for extreme classification. In *Proceedings of the Web Conference 2021*. 3733–3744.

[34] 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*. PMLR, 3182–3190.

[35] 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*. PMLR, 4664–4673.

[36] Lijuan Sun, Songhe Feng, Tao Wang, Congyan Lang, and Yi Jin. 2019. Partial multi-label learning by low-rank and sparse decomposition. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5016–5023.

[37] Yu-Yin Sun, Yin Zhang, and Zhi-Hua Zhou. 2010. Multi-label learning with weak label. In *Twenty-fourth AAAI conference on artificial intelligence*.

[38] 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*. 455–464.

[39] Rajasekar Venkatesan, Meng Joo Er, Mihika Dave, Mahardhika Pratama, and Shiqian Wu. 2017. A novel online multi-label classifier for high-speed streaming data applications. *Evolving Systems* 8, 4 (2017), 303–315.

[40] Bingyu Wang, Li Chen, Wei Sun, Kechen Qin, Kefeng Li, and Hui Zhou. 2019. Ranking-based autoencoder for extreme multi-label classification. *arXiv preprint arXiv:1904.05937* (2019).

[41] Haobo Wang, Weiwei Liu, Yang Zhao, Chen Zhang, Tianlei Hu, and Gang Chen. 2019. Discriminative and Correlative Partial Multi-Label Learning.. In *IJCAI*. 3691–3697.

[42] Zhen Wang, Liu Liu, and Dacheng Tao. 2020. Deep streaming label learning. In *International Conference on Machine Learning*. PMLR, 9963–9972.

[43] 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*. 1275–1284.

[44] Liang Yang, Xi-Zhu Wu, Yuan Jiang, and Zhi-Hua Zhou. 2019. Multi-label learning with deep forest. *arXiv preprint arXiv:1911.06557* (2019).

[45] Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. 2017. Learning deep latent space for multi-label classification. In *Thirty-first AAAI conference on artificial intelligence*.

[46] Ian EH 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*. 545–553.

[47] 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*. PMLR, 3069–3077.

[48] 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. *Advances in Neural Information Processing Systems* 32 (2019).

[49] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. 2014. Large-scale multi-label learning with missing labels. In *International conference on machine learning*. PMLR, 593–601.

[50] Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering* 18, 10 (2006), 1338–1351.

[51] Min-Ling Zhang and Zhi-Hua Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition* 40, 7 (2007), 2038–2048.

[52] Yi Zhang and Jeff Schneider. 2011. Multi-label output codes using canonical correlation analysis. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 873–882.