Multi-label Classification for Web Services

Mandar Badave
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
mrb5960@rit.edu

Abstract—Multi-label classification has gained popularity due to its uses in categorizing textual data, labeling images, audio files, E-mails etc. Medical sciences and bio-informatics are some other domains where people have started noticing its advantages. This paper summarizes various researches that are being carried out to make multi-label classification algorithms more efficient despite the challenges faced by multi-label data such as high dimensionality and imbalanced nature of the data. This paper also proposes the usage of multi-label classification in the the domain of web services which will help in web service tagging, discovery and replacement.

Index Terms—multi-label; classification; webservice

I. INTRODUCTION

We live in a “Data age” i.e. data matters the most and it is generated in huge amounts on a daily basis. Extracting information out of this data is a challenge and several supervised as well as unsupervised machine learning techniques have been utilized to get the maximum out of the data.

Classification is a supervised process of assigning labels or predicting a class of a data instance based on the training data. Classification can be divided into following types:

1) Binary classification: It has 1 label, which can have 2 values e.g. Yes/No, 0/1.
2) Multi-class (or Single Label) classification: It has 1 label, which can have multiple values.
3) Multi-label classification: It can have more than 1 labels and each label can have 2 values.
4) Multi-output classification: It can have more than 2 labels and each label can have multiple values.

Let’s take an example to understand the difference between the above types and why do we need Multi-label classification (MLC). Consider a movie as a data instance, and the genres it belongs to as the labels. A binary classifier would not be ideal for this task as it can only provide us whether a movie belongs to a certain genre i.e. Yes or No. Multi-class classifier (SLC) can give us only a single genre but in real-world a movie can belong to multiple genres and hence even SLC cannot provide us with the output that we expect. This problem is solved by MLC. Given a movie, it can predict all the genres a movie belongs to as it has multiple labels that specify if the movie belongs to a genre or not. Similarly, MLC can be applied to categorize texts, label images, etc.

Web services is a domain where the usage of MLC have not been explored. Web services and mashups can benefit from MLC in various ways. Web services can be tagged or labeled according to functionalities they offer. Similar to the above examples, a web service can offer more than one functionality. If SLC is used for classification of web services, it can result in loss of valuable information. Hence, it is necessary to use MLC. Tagging web services correctly will help in web service discovery and web service replacement as well. Keeping this in mind I plan to exploit the uses of MLC in the domain of web services.

As MLC is different from other types of classification, we need to consider the problems specific to MLC as well. Some of the challenges related to MLC are:

- High dimensionality: The dimensionality increases with increase in the number of labels. The number of label combinations grow exponentially with the number nof labels.
- Imbalanced data: Multi-label data is imbalanced in nature due the number of label combinations. A label set is associated with very few instances and this can hurt the accuracy.
- Size of data: It is difficult to find correlations among instances and labels if the size of data is small.

There has been a lot of work recently to overcome these challenges. Section II describes the approaches to solve a MLC problem and the relevant work carried out in each approach.

II. BACKGROUND

MLC classification can be divided into two broad categories such as Problem Transformation (PT) and Algorithm Adaptation (AA). These approaches along with some other miscellaneous methods are discussed in detail in the following sections.

A. Problem Transformation

The Problem Transformation method transforms a MLC problem into a multiple SLC problems so that traditional classification algorithms can be used to solve them. There are several ways to transform a multi-label problem into single label problem. Some of the methods are select-max, select-min and select random. If there a n number of multi-label instances, select-max assigns a single label that has the highest frequency in n instances whereas select-min assigns a single label which has the minimum frequency and select-random chooses a random label from the given set.
of labels for an instance. Another common approach called Binary Relevance builds separate binary problems for each label.

1) PT using Random Walks: Paper [1] uses a random walk model called MLRW for problem transformation. The first step in MLRW is building Multi-label Random Walk Graphs (RWG). MLRW maps training data to RWG. Every instance from training set is mapped to a vertex in RWG. Two vertices are connected if they are assigned one or more common labels.

The next step is building Multi-label Random Walk Graph Collection (RWGC). As the name suggests, RWGC is a collection of RWGs. A new RWG is formed for each label. An unlabeled instance vertex is connected to vertices having a common label. This is done for all the labels. If there are \(L\) number of labels, an RWGC will contain \(L\) RWGs.

After obtaining the RWGC, random walks are carried out on each RWG in RWGC. As the name suggests random walks are random in nature. In case of an RWG, random walks can be randomly visiting the vertices in the RWG. The output of a random walk is probability distribution which depicts the probability that a testing instance can have a specific label assigned to it. The probability distribution always forms a bell curve.

A walker in a RWG has two options, it can visit one of its neighbors or it can teleport to a random vertex. The probability that the walker visits a random vertex is called the teleporting probability. Therefore, if the teleporting probability is \(\alpha\), the probability that the walker visits the random vertex is \(\alpha\) and the probability that it visits one of its neighbors is \(1-\alpha\). The walker has to choose which neighbor it visits next. This is decided by calculating the Euclidean distance between the vertices. The probability of visiting a neighbor is higher if the Euclidean distance is small. The random walk in an RWG takes four inputs to figure out its next move.

The inputs are an adjacency matrix, a vector that represents the initial value or the starting position, a teleporting vector and \(\alpha\), also known as the teleporting probability. The starting position vector is updated iteratively using the new values for the above mentioned parameters. The random walk is said to converge or come to a halt when two subsequent vectors are same. The final vector obtained when the random walk comes to a halt is considered as the similarity vector. If \(D\) is the number of training instances, the size of the similarity vector is \(D+1\), 1 representing the unlabeled instance \(t\). The similarity vector shows the similarity between the \(t\) and a training instance \(x\) in the RWG. If \(d\) is the number of training instances, there will be \(d\) similarity vectors.

These vectors can now be used to calculate the similarity between \(t\) and a label \(L\). The similarity scores for a label \(L\) from \(d\) similarity vectors are averaged to get a similarity score between \(t\) and label \(L\). These scores are then used to calculate the conditional probability vector.

The conditional probability that a label \(L\) can be assigned to \(t\) is given as the ratio of the similarity score between \(t\) and \(L\), and summation of similarities scores between \(t\) and all the labels.

The next part is transform the MLC into multiple SLC. This is done by taking a validation set. This set contains labeled instances from the data and are used to the train the classifiers for each labels. Conditional probability vectors are obtained for the validation data using the similar process as mentioned above and these vectors are used to build a new training dataset. A conditional probability vector for an instance \(t\) is a positive data instance for a label \(L\) if the actual label set of \(t\) contains \(L\) or else it is considered as a negative instance. The positive and negative instances together form a training set for a SLC. If there are \(q\) labels, there will be a SLC for each label. The \(q\) conditional probability vectors obtained for the unlabeled instance are passed as input to \(q\) classifiers to get the final prediction.

Experiments were performed on yeast and scene datasets. MLRW was compared with other state of the art algorithms. Two types of evaluation metrics were used i.e. ranking based measures such as multi-label average precision, one-error, coverage and classification based measures like precision, recall and F-measure. Adaboost.M1 was used to train the classifiers. 5-fold cross validation was used for the experiments and each experiment was run 10 times. In terms of F-measure, MLWR achieved approximately 10 percent improvement over BR and MLkNN and 22 percent over LP. Comparison of other metrics showed that MLRW is better than other MLC methods.

2) PT using Meta-labels: Paper [2] is a framework that uses LP techniques along with meta-labels. Meta-labels are nothing but different combination of the labels from the label set. For example, a meta-label of labels A and B can be \(\{\phi, A, B, AB\} = \{00, 01, 10, 11\} = \{0, 1, 2, 3\}\). The meta-labels are formed using three steps. They are partitioning the label set into new labels, relabeling the newly created label sets and recombining the predictions of the meta-labels into a label vector prediction[2].

Partitioning: Strategies for label partitioning differ based on the information provided about the labels in dataset. For example, partitioning can be done by using hierarchical proximity if the information regarding the hierarchy among the labels is defined in the dataset[2]. If not, hierarchy among the labels can be found out using balanced k-means clustering technique. This technique forms groups of similar instance spaces.

Relabeling: If a specific label set is assigned to very few instances, it can hurt the performance of the classification algorithm. It is important to remove or ignore such label sets. This paper uses a method called Pruned Sets (PS) for the same purpose. PS prunes or removes p-infrequent partitions or label sets i.e. partitions that occur less than p times in the dataset[2]. This may decrease the size of the dataset significantly if most of the partitions occur infrequently. To avoid such a situation, the label sets are further divided into subsets. For example, for a label set containing A and B, it can be further divided into two subsets, one containing label A and another containing label B. These subsets can then be reintegrated. The number of label subsets that are to be integrated is limited by
a number \( n \). First priority is given to the size of the label subsets followed by the frequency of the label subsets in the training data. Consider A, B, C are infrequent labels, and AB, BC and AC are the label subsets. If \( n=2 \), and AB and BC have more frequency in the training set, AC is left out and AB and BC are reintegrated. Consider labels A and B. The possible combination of these variables are \{00, 01, 10, 11\}. In this case, if 01 and 10 are pruned while performing PS, we are left with 00 and 11. This means that either we can assign both the labels A and B or neither of them. Such cases help the learning process of the classifier and makes the reconstruction task easier.

**Reconstruction:** Reconstruction procedure is different for overlapping and non-overlapping subsets. It is easier to recombine the results of non-overlapping subsets. For a label set A, B and C the non-overlapping subsets can be thought of as AB and C. If AB is assigned 01 and C is assigned 1, the final result will be \{ A, B, C \} = \{ 0, 1, 1 \}. In case of overlapping subsets posterior predictive distribution is used and a threshold is applied to get the final prediction.

SVMs are used as the classification algorithm for the experiments. Jaccard index is used as the evaluation metrics. The proposed framework is used in EpRd which is an ensemble of 10 pruned Rd (K random disjoint equal sized partitions) was compared with other methods such as Binary Relevance, LP, HOMER (K Hierarchical partitions) etc. The accuracy of EpRd was highest followed by HOMER, LP and BR. An experiment was carried out between RAkEL and RAkELp to check the change in performance. RAkEL also makes use of meta-labels with overlapping partitions. The accuracy increases as number of partitions are increased but there is a considerable increase in the running time of the algorithm. RAkELp not only prunes the label sets with less frequency but also further divides them. The results show that the accuracy is not hampered by pruning and there is a significant drop in running time for RAkELp.

The main difference in the above two PT methods is that they use two different PT techniques. The Random Walk model uses a Binary Relevance approach whereas the Meta-labels model uses a Label Powerset approach. Another difference between the two paper is that [2] uses an ensemble approach i.e. it uses multiple techniques in unison to improve the classification accuracy. [1] does not alter the original dataset in any way but [2] uses pruning to remove instances whose label sets rarely occur in the dataset.

**B. Algorithm Adaptation**

SLC algorithms have been around for a while now. This inspired researchers to modify these SLC algorithms to handle multi-label data.

1) **Mr.KNN:** Mr.KNN approach is divided into two parts, first part calculates the soft relevance score by modifying the fuzzy c-means algorithm (FCM) and the second part is the modified kNN classifier[3].

The reason to calculate a relevance score is to overcome a drawback of binary relevance strategy. Consider a data point \( d \) in a 2D space containing two clusters of points \( c1 \) and \( c2 \). If \( d \) is a part of \( c2 \) but in reality it belongs to \( c1 \), it is not a part of \( c2 \) but only an outlier to \( c1 \). If binary relevance is used to classify \( d \), it will be classified as positive in both \( c1 \) and \( c2 \). This will affect the classification accuracy. Hence, the fuzzy c-means algorithm is modified to handle supervised data. In this process, every class is treated as an individual cluster. The centers of these clusters are the means of their respective sample space. The relevance score is calculated using the known labels and Minkowski distance between an instance and the centroid. This process gives a membership score between each instance and each know labels. If the label set of an instance does not contain a specific label, the membership score between that label and the instance is considered as 0. Two steps are performed iteratively in fuzzy c-means, first step is to update the relevance score based on the centroids and second step is to update the position of the centroids itself. The process stops when there is no change in the assignments of labels to clusters.

Second part is the Margin Ratio voting mechanism used by kNN. Traditional kNN voting function does not take into account the distribution of the data and also the distance between an instance and its k nearest neighbors. If a dataset is imbalanced, i.e. if there are more data instances that belong to a single category, there is a higher chance that a test data instance is classified incorrectly as it would be surrounded by data instances of a different category. These problems are addressed in Mr.KNN that makes changes to the voting function by using a distance weighting mechanism and the relevance score obtained from the first step. The goal of this process is to find values of \( f \) and \( k \) in Minkowski distance and in kNN respectively, where the average voting margin ratio is maximum[3]. For classification, the values of \( f \) and \( k \) that have the maximum average voting margin are chosen[3].

Experiments were carried on three multi-label datasets. The results were compared with an existing multi-label classification mechanism called ML-KNN. Hamming loss, precision, recall and accuracy were the metrics used for performance evaluation. The training time required for Mr.KNN was more than ML-KNN as it calculates the average margin ratio. The testing time for both the algorithms was same. Comparison of the evaluation metric values showed that Mr.KNN outperformed ML-KNN for all the three datasets. The result of increasing the number of unique labels is also compared. With increase in the number of unique labels, the performance of ML-KNN decreases because it uses Binary Relevance whereas performance of Mr.KNN remains consistent.

**C. Miscellaneous Approaches**

1) **Mining Instance and Label Correlations:** One of the challenges involved in multi-label classification is the possibilities or the ways in which multiple labels can be assigned to an instance. These possibilities are exponential to the total number of labels. Hence, it is important to learn the correlations among the labels in order to make the process of classification more efficient. Paper [4] proposes the use of Heterogeneous
Information Networks to find out the correlation between labels as well as correlation between data instances.

The authors give a real-life example of Drug-Target binding prediction to show how Heterogeneous Information Networks can be used to facilitate the classification process. A drug can bind with multiple genes. Finding out which genes can a drug bind with to discover a cure for a disease can be thought of as a multi-label classification problem. The label space is large and hence it is difficult to find out the correlation among the labels using a small data set. A Heterogeneous Information Network related to above problem can have entities or nodes such as chemical compounds and genes and links representing the relationship among different genes and chemical compounds. The genes can have a link to other nodes such as diseases and chemical compounds can have a link to nodes such as side effects. Such a network can be build with the use of domain knowledge. The genes having a link to a common disease have a greater possibility of appearing in the same label set than the ones that do not have such common links. Similar outcome can be expected from chemical compounds having common side effects. This process of obtaining information from the Heterogeneous Information Networks can be called as linkage structure mining. This information can be used to infer relationship between labels as well as data instances.

**Heterogeneous Information Networks:** A Heterogeneous Information Network can be defined by a directed graph. The nodes are considered as vertices in the graph. The nodes are made up of instance types and label types. The nodes that correspond to the instances are called as instance types and the nodes that correspond to the labels are called as the label types.

**Meta-path:** A sequence of relations in a Heterogeneous Information Network is called a meta-path[4]. These relations are the links between two nodes in the network. A relation \( R \) exists between two nodes if they have a link \( R \) between them.

The correlation among the labels can be determined by mining information out of the linkage structure in a Heterogeneous Information Network. Meta-paths describe this relationship among the labels. In the drug-target binding prediction example, genes are the labels. Multiple genes can be connected by a \( PPI \) link. The genes that have a \( PPI \) link between them have a greater probability of either being present or absent together in a label set. The meta-path for this example is denoted as "\( \text{gene} \xrightarrow{\text{PPI}} \text{gene} \)[4]. Another example of a meta-path is "\( \text{gene} \xrightarrow{\text{hasGO}} \text{GO} \xrightarrow{\text{hasGO}^{-1}} \text{gene} \)[4]. This notation suggests that two genes have the same ontology. \( \text{hasGO}^{-1} \) denotes an inverted relation of \( \text{hasGO} \)[4]. These examples show that information about correlation among the labels can be extracted from Heterogeneous Information Network using meta-paths that start and end with a label.

Instance correlations can be extracted from the Heterogeneous Information Networks in the same way as mentioned above. The meta-paths just need to start and end with an instance node. These instance correlations can be used to find the correlation among the label sets of those instances.

For example, "chemical compound \( \xrightarrow{\text{causeSideEffect}} \text{SideEffect} \xrightarrow{\text{causeSideEffect}^{-1}} \) chemical compound" indicate that chemical compounds that linked to the same side effect node have a greater probability of having same label sets[4].

**The Unified Model:** The unified model makes use of both label correlations instance correlations to perform multi-label classification. The classification algorithm has three steps. First step is to construct the meta-paths from the Heterogeneous Information Networks. A set of meta-paths is denoted as \( S_t \). The task is to find all the non-redundant meta-paths. A meta-path can be called as non-redundant if it cannot be reconstructed from the paths in \( S_t \). The second step is training initialization. Here, a new extended training set is created using the meta-paths for labels and instances. The third step is the iterative inference. The classification algorithm (PIPL) iteratively updates the predicted label sets. The number of iterations is limited to a certain value.

Bioinformatic dataset called SLAP is used for experimental evaluation. It is a Heterogeneous Information Network which has 290K nodes and 720K edges. Two prediction tasks were performed on the dataset. Micro-F1, hamming loss and subset loss were used as evaluation metrics. The performance of PIPL was compared with five other algorithms. They are \( B_{SVM} \) (Binary SVM), ECC (Ensemble of Classifier Chains). PISL, ICML and PIML. \( B_{SVM} \) is a binary classification algorithm that does not consider any type of correlation among the labels or instances[4]. Ecc considers correlation among labels whereas PISL considers label correlation among instances from Heterogeneous Information Networks[4]. ICML considers label correlation from the data samples and correlation among instances from Heterogeneous Information Networks[4]. PIML considers correlation among instances from Heterogeneous Information Networks[4].

The results show that \( B_{SVM} \) under performs as compared to all other classification algorithms as it does not consider any kind of correlation among labels or instances. Another insight from the results is that the algorithms that exploit meta-path based correlations perform better to those that just use the correlation among the labels or instances from the Heterogeneous Information Networks.

2) **AEClass:** Paper [5] talks about a framework called activity-edge centric multi-label classification. The framework comprises of three parts. The first part is representing the heterogeneous information networks in terms of a collaboration graph and multiple activity graphs. The collaboration graph is converted to a collaboration multigraph using a technique called vertex-edge homophily. Second part is calculating the closeness among the vertices using label vicinity. Third part is using the AEClass classification algorithm that dynamically adjusts the weights for the activity graphs resulting into improved results.

Multi-label classification techniques used for heterogeneous information networks suffer with two drawbacks. First, the heterogeneous data is not divided into different activity graphs. Second, edge-centric multi-label classification and
vertex-centric multi-label classification are not used in unison to improve the results. The activity-edge centric approach overcomes both these drawbacks. The steps involved in AEClass are as follows:

Authors propose a technique called as vertex-edge homophily to improve the accuracy of multi-label classification. It means that the accuracy can be improved if the similarity between both the edges and the vertices is considered. Activity-based edge classification is used to achieve vertex-edge homophily. This process involves converting a collaboration graph to a activity-edge augmented collaboration graph. Every edge in a collaboration graph is divided into separate edges by comparing it with each activity graph.

The above process results into collaboration graphs which are then combined into a single multi-graph. After combining, if there exists multiple edges between two vertices, they may have different contributions in the vertex classification process. The weight of each edge is determined by a dynamic weight tuning mechanism and transition probability. This approach improves the accuracy of the multi-label classification as AEClass considers the edges that are only related to the objective.

The similarity between the activity categories itself, can affect how multi-label classification performs. The similarity between the categories with respect to each activity graph is calculated. This value is called as edge label dependency. These values are considered in the AEClass framework.

Label vicinity is calculating the closeness of the vertices using the labels on the activity-based collaboration multi-graph. The weights are dynamically adjusted to improve classification results and avoid unnecessary computations leading to an increase in efficiency.

The DBLP bibliography, Last.fm and IMDB were the datasets that were used for the experiments. The performance of AEClass was compared against four other classifiers. The evaluation metrics used were Micro-F1 and Hamming Loss. The larger the value of Micro-F1 and the smaller the value of Hamming Loss, the better the quality[5]. After performing 10 fold cross validation, AEClass achieved best performance figures on all the datasets among all five classifiers. The reasons for the achieving a better accuracy are information provided by the activity networks, vertex-edge homophily, edge label dependency and dynamic weight learning.

3) Mixture of Trees: Paper [6] talks about mixture of Conditional Tree-structured Bayesian Networks (CTBNs) that allows to define the class posterior probability \( P(Y|X) \) where \( Y \) are class variables and \( X \) are feature variables. Building a multi-label classification model using posterior probability helps in prediction, decision making and outlier analysis.

The mixture of conditional tree-structured Bayesian networks (MC) combines two concepts viz. Mixture of Trees (MT) and CTBN. MT is a combination of multiple trees which are used to define a model using joint distribution. The \( \lambda_k \) is called as the mixture components that represent the distribution of the outputs defined by \( k_{th} \) tree[6].

The next step is learning the parameters of the MC which are the mixture coefficients. The objective of learning the parameters to improve the log-likelihood which is done using expectation maximization.

The parameter learning procedure is followed by learning the structure of the CTBN from given weighted data. The optimal structure of the CTBN is found out using an algorithm that finds maximum weighted conditional log-likelihood for a CTBN. The initial obtained CTBN using weights which are initially uniform and adjusted iteratively. This CTBN structure is then used to find the next structure by normalizing the error for each instance using the training data.

The prediction of labels for an instance can take exponential time. CTBN approach overcomes this by avoiding unnecessary checks of all possible combination of the labels. This is done by an algorithm called Markov chain.

Experiments were performed on 10 datasets. Performance of the proposed approach was tested against simple binary relevance and other multi-label classification techniques. Exact match accuracy (EMA) was used as one of the evaluation metrics for the performance. EMA is calculated as the number of instances that are exactly classified. As EMA is harsh metrics, another metrics called conditional log-likelihood loss (CLL-loss) was used. Macro F1 and micro F1 were also used as evaluation metrics. 10-fold cross validation was used for evaluation. MC outperformed other methods in terms of EMA. MC also outperformed other methods in terms of CLL-loss as one of the features of MC is optimizing the conditional log-likelihood. Performance was also tested by increasing the number of trees in the mixture from 1 to 20 and CLL-loss and EMA were calculated. The performance improves till a certain number of trees are added after which it stabilizes. Similar behavior is observed with EMA but the improvement is not as significant as in the case of CLL-loss.

From the above descriptions we can see that [4] and [5] uses Heterogeneous Information Networks (HIN) for classification and [6] uses mixture of tree structured Bayesian Networks for classification. [5] uses HINs to build collaboration and activity graphs so that distance among the vertices can be calculated which can be used for classification. On the other hand, [4] utilizes the linkage structure of the HINs to obtain correlation among the instances as well as correlation among the labels of various instances. [6] uses a probabilistic approach. It uses tree structure Bayesian Networks which reduce the learning time of the classification model.

III. PROJECT HYPOTHESIS

MLC hasn’t been explored in the domain of web services. There are two possible reasons for this. First, performing classification on textual data can be a challenging task as it involves a lot of data pre-processing tedious feature extraction. Second, inaccessibility to a readily or publicly available dataset. MLC can be used for web service tagging i.e. assigning labels which will result into better and efficient discovery of web services.
As mentioned earlier, MLC has several challenges such as high dimensionality, imbalanced data and size of training data. While there are several approaches to solve a MLC problem viz. problem transformation and algorithm adaptation, their performance highly depends on the features. The challenges of MLC can be overcome by carefully selecting and extracting these features.

When it comes to feature extraction from textual data, the most informative part of a web service is its description. There are various techniques of extracting features from textual data such as bag of words, TF-IDF, Word2Vec and Doc2Vec. Bag of words and TF-IDF use the frequency of words while Word2Vec and Doc2Vec represent words as vectors of real numbers. Every technique has its own advantages and drawbacks but using these techniques in unison can improve the classification accuracy. Word2Vec represents words in the form of vectors such that semantically similar words will have similar vectors. The advantage of these vectors is that their dimensions are low as compared to the size of the vocabulary and this can drop the computation time significantly. A document can be represented as a vector by taking the mean of the vectors of the words that are present in it. These vectors can then be used as features for MLC. However, the drawback of this method is that every word in the document is given equal importance which is not an ideal solution. A more meaningful approach can be assigning weights to important words and taking weighted average of the word vectors.

IV. DESIGN
The design of the proposed system is shown in Figure 1.
1) Data: The data consists of the API name, API description and API categories.
2) Feature selection: Several feature extraction methods are used to obtain feature sets.
3) MLC approaches: MLC is performed on the web services data using feature sets obtained from the above step. The categories of the web services are the labels or target variables. The data is divided into training and testing sets. The training set is used to train multiple models from each approach.
4) Predict Labels: The trained models are used to predict the labels of the validation set. This is followed by a comparison of performances of multiple models using evaluation metrics.
5) Evaluation and comparison: The accuracy of the each model is calculated using evaluation metrics which is used as a basis for comparison.

V. IMPLEMENTATION
This section consists of details related to data gathering, data formatting, data cleaning and preprocessing. These tasks are followed by implementation details of the framework mentioned in IV. Python was used to perform all the tasks in the implementation. The details of the packages used for each task are mentioned in the relevant sections.

A. Data gathering
The data was crawled from www.programmableweb.com. I used a scraping tool called Scrapy to crawl the website. Description, name, primary and secondary categories of the web services were crawled. The dataset has 16441 web services and a total of 477 unique labels after removal of 709 duplicates.

B. Data preprocessing
Textual data requires a lot of preprocessing before feature extraction. The data, obtained by crawling, has lots of inconsistencies and anomalies. The data is received in a JSON format which is then processed and stored in a csv file for further use. The nltk package was used for preprocessing. Following are the steps involved in data preprocessing.
1) Data Cleaning: Web services which did not have any categories or description are removed from the dataset followed by removal of duplicate records.
2) Lowercase: Converting everything to lowercase makes the data consistent. This helps in finding a pattern using regular expressions easier.
3) Removal of special symbols: All characters, except the alphanumeric characters and period, are removed from the corpus as they do not provide any information. Sentences can be formed by splitting the corpus based on the period. Hence, period is required while building a Word2Vec model as the input to the model is sentences. The data generated after this step is used as input data for Word2Vec.
4) Tokenizing: This step is carried out only on the description of the web service as it is used for feature extraction. The description is a single string which is broken down into tokens or words using space as a separating character.
5) Lemmatizing: Lemmatizing converts a word to its base lemma or a word that exists in a dictionary. This helps in reducing the dimensions of TF-IDF matrix.
6) Removing stopwords: Stopwords are the words that are of less significance when it comes to document classification. These are the words that occur with high frequency throughout the documents. Generally, articles and pronouns are thought of as stopwords. For example, in the sentence ‘This is a cat’, ‘this’, ‘is’ and ‘a’ are called as stopwords.
7) Removing short words: The words whose length is less than 2 are removed. e.g. ‘a’, ‘an’, ‘is’
8) Removing less informative records: The web services with description less than 20 words are removed from the dataset.

C. Data formatting
The categories of the web services are obtained in the form of a list. Every category from unique set of categories was converted to a column in the dataset. The values in the columns are binary in nature i.e. 0 and 1. If a web service belongs to a category then the value in the category column is set to 1, else it is set to 0.
D. Feature Extraction

Following are the feature extraction techniques used in this paper:

1) **TF-IDF**: A TF-IDF matrix is generated using the sklearn package and the top 1000 features are used as the features for classification algorithms. The unique number of words in all of the web service description exceeds 20000. Performing operations using a TF-IDF matrix of all of the words is an resource intensive task. Hence, it is important to consider only a limited number of most important features.

2) **POS tagging and TF-IDF**: POS tagging is used to extract only the nouns from the description of the web services. There are two advantages of POS tagging. First, the nouns provide valuable information related to a document and second, it reduces the dimensions of the TF-IDF matrix.

3) **D2V**: In this step, a word2vec model is trained using the web service descriptions. A document vector representing the entire web service is formed by taking the mean of the word vectors. The gensim package is used to build the word vectors. A parameter can be passed to the Word2Vec which specifies the number of dimensions for the word vectors. The parameter is chosen to be 300.

4) **Google D2V**: This step is similar to the above step. The only difference is that instead of training the word2vec model on the web service descriptions, an already trained model is used. GoogleNews model is used as a pre-trained model and the word vectors in this model are used to generate the document vectors. An advantage of using a pre-trained model is that the word vectors are more accurate as the model is trained on a very large corpus.

5) **Weighted D2V**: In the above D2V steps, the document vectors are generated by taking the mean of the word vectors. But in this step, we take the weighted average of the document vectors to form the document vectors. The TF-IDF values are used as the weights.

E. MLC approaches

This section discusses the algorithms that are used from each MLC approach. The scikit-multilearn package for python has implementations for some of the common MLC algorithms. Following are the algorithms used in this paper.

1) **Problem Transformation approaches**: Consider the example below where $X_1, X_2, X_3$ are the features and $Y_1, Y_2$ are the labels. The results of the PT approaches are shown in the respective sections.

   $$
   \begin{array}{cccccc}
   X_1 & X_2 & X_3 & X_4 & Y_1 & Y_2 \\
   0.26 & 0.72 & 0.54 & 0.11 & 1 & 0 \\
   0.31 & 0.48 & 0.13 & 0.52 & 1 & 1 \\
   0.09 & 0.65 & 0.52 & 0.17 & ? & ? \\
   \end{array}
   $$

   - **BinaryRelevance(BR)**: A binary classifier is trained for every label individually which is used for prediction of each label separately.

   $$
   \begin{array}{cccccc}
   X_1 & X_2 & X_3 & X_4 & Y_1 \\
   0.26 & 0.72 & 0.54 & 0.11 & 1 \\
   0.31 & 0.48 & 0.13 & 0.52 & 1 \\
   0.09 & 0.65 & 0.52 & 0.17 & ? \\
   \end{array}
   $$

   - **ClassifierChain(CC)**: This approach is similar to BR. The only difference is that once a label is predicted, it is included as a feature that can be used to train the next binary classifier.

   - **LabelPowerset(LP)**: The labelsets for an instance are considered as a single class. LP transforms a multi-label problem to a multi-class problem.
All of the above algorithms require a base classifier as a parameter that can be used to train a binary classifier. The base classifier can be any multi-class classifier. The classifiers used in this paper are Naive Bayes (NB), Decesion Tree (DT), Logistic Regression (LR) and Support Vector Machine (SVM).

2) Algorithm Adaptation approaches
- **MLkNN**: Finds k nearest neighbors. Uses maximum a posteriori to assign labels to an instance[7]

As all of the above algorithms are adapted from kNN, a value of k has to be given as a parameter.

F. Evaluation

The data is divided into training and testing sets for evaluation. Precision, recall and f1-score are used as the evaluation metrics. Accuracy is not used as it is considered as a harsh evaluation metric for multi-label classification. 10-fold cross validation is used to calculate accurate values of precision, recall and f1-score. The advantage of using cross validation is that all the data points are considered in the training as well as testing phase. The model is trained and tested 10 times and the final score is the average of the scores of 10 models.

- **Precision**: The ratio of true positives to the sum of true positives and true negatives, averaged over all the instances.
- **Recall**: The ratio of true positives to the sum of true positives and false negatives, averaged over all the instances.
- **F1-score**: The harmonic mean of precision and recall.

The above scores will be used for comparing the performances of multiple models.

VI. RESULTS

Experiments are carried out on three approaches as follows: As the total number of labels is 477, training and testing all models with 10-fold cross validation is an expensive task in terms of computation time. It is difficult to obtain the results even after several hours of computing. Hence, only the 20 most common labels are selected for experimentation and the web services which did not have any of those labels assigned to them are not considered. The new dataset contains 11551 web services and has a label cardinality of 1.4. The feature sets are extracted from the dataset and are used to train PT and AA models. The results of PT approach for each feature set are shown in table I to V. The results of AA approach for all the feature sets are shown in table II. MLkNN algorithm is used with the number of neighbors i.e. the value of k chosen to be 4. Figure 2 shows a comparison of F1 scores for PT approach using all feature sets. Figure 3 shows a comparison of time taken PT approach using all features.

The first thing we notice after taking a look at the graph in Figure 2 is that the combination that gives the best result is CC-SVM and using TF-IDF as the feature set. It also takes very less time as seen from Figure 3. In terms of MLC algorithms, LP is the most consistent classifier for all feature sets followed by CC and BR. In terms of base classifiers, SVM performs better than the others irrespective of what MLC algorithm is used. It also takes less time except for LP. The time taken by DT varies significantly based on the MLC algorithm used. There is an increase in time when LR and SVM are used with LP.

TF-IDF performs better than combining TF-IDF and POS tagging because POS tagging results in loss of information as only the nouns are selected to generate the TF-IDF matrix. There may be words which are not nouns but still are unique to a web service. Neglecting these words degrades the performance. While generating the TF-IDF matrix, a parameter can be used to limit the number of features. This parameter selects only the top n important features irrespective of whether it is a noun or not. Hence, TF-IDF performs better in every case. The performance of TF-IDF and TF-IDF with POS tagging degrades in the case of Naive Bayes irrespective of the MLC algorithm.

Google D2V performs better than D2V as it is trained on a very large corpus and hence the word vectors are much more accurate that the word vectors obtained by training a model only on the descriptions of the web services. The results of weighted D2V show that it closely matches the results of D2V but lags behind Google D2V. Weighted D2V may or may not perform better than Google D2V. In this case, the size of the web service description degrades the performance. The average number of words in the web service description is approximately 75. The performance may increase if there are more number of words in the web service description as more number of words would result into more accurate document vectors. Another reason could be the size of the TF-IDF matrix. The unique words in the web service descriptions of all the web services is approximately 25000 whereas we consider only top 1000 features due to limitation of resources. Hence, the weights of only 1000 words are used for their corresponding word vectors and the words which are not present in the TF-IDF matrix are not considered.

VII. CONCLUSION

From the results we can conclude that TF-IDF performs slightly better in most cases than Word2Vec for this dataset. Using POS tagging to select only nouns results in loss of information as there can be other words that are not nouns which are of importance while classifying a web service. A Word2Vec model will be trained better on a larger corpus.
and will result in better accuracy of the word vectors. Label Powerset performs the best among all the MLC approaches and SVM performs the best among the base classifiers.

VIII. FUTURE WORK

In the future, we can take the relationship among the labels into consideration i.e. the degree of correlation among two labels. If two labels have a high positive correlation, both the labels can be assigned to a web service. If two labels have a high negative correlation and if one label is assigned to a web service, the other can be automatically skipped.

ACKNOWLEDGMENT

I would like to thank my project guide Dr. Xumin Liu who supported me and directed me towards the right path. I would like to thank Wei Shi Shi who helped me with his expertise in coming up with better ideas to solve the problem in this paper. I would like to thank my colloquium advisor Dr. Joe Geigel for helping me in keeping track of the milestones and giving valuable feedback on the milestone presentations. He also gave the guidelines we need to follow while preparing the poster.
<table>
<thead>
<tr>
<th>Features</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF-IDF</td>
<td>0.54</td>
<td>0.44</td>
<td>0.47</td>
<td>2195</td>
</tr>
<tr>
<td>TF-IDF w/ POS</td>
<td>0.54</td>
<td>0.41</td>
<td>0.45</td>
<td>2112</td>
</tr>
<tr>
<td>D2V</td>
<td>0.43</td>
<td>0.32</td>
<td>0.35</td>
<td>832</td>
</tr>
<tr>
<td>Google D2V</td>
<td>0.57</td>
<td>0.42</td>
<td>0.47</td>
<td>855</td>
</tr>
<tr>
<td>Weighted D2V</td>
<td>0.47</td>
<td>0.32</td>
<td>0.36</td>
<td>1765</td>
</tr>
</tbody>
</table>

**REFERENCES**


