Extreme Multilabel Classification: Project Proposal

Aidan Claffey (adc501), Kevin Wilson (ksw366), and Daniel Turkel (dgt238)

April 1, 2020

Introduction

The extreme classification setting involves problems in which we hope to assign multiple labels to items from a large set of candidate labels [1]. In our specific problem, we are tasked with predicting labels for the EUR-Lex dataset.[1] This dataset of European legal documents has roughly 15,000 examples in the training set, with about 4,000 labels in the full dataset applied at a rate of roughly 25 examples per label and 5.3 labels per example. Our version of the dataset has the data preprocessed so that the 5,000 features are TF-IDF scores of words from the original text and the labels are transformed into integers.

The challenge in extreme classification comes from problems of sparsity and computation. Some labels may occur only a very small number of times, or may only occur simultaneously with another label, making them difficult to distinguish. Additionally, some classical methods for multilabel classification become intractable in the extreme setting due to computational constraints: predicting combinations of labels from the power set of all labels, for example, is not feasible when the number of labels is in the thousands.

Approach

Our plan to approach this problem first involves understanding the dataset and the problem of multilabel classification. We will perform exploratory data analysis to understand the data including the co-occurrence, sparsity, and interdependence of labels. Given the nature of the data, we may find that the training set includes some labels that occur very infrequently and thus meaningful predictions around such classes will be difficult. We plan to identify and characterize such instances in this data exploration step.

Next, we intend to perform model selection from a broad set of approaches for multilabel classification. We have given ourselves a wide set of options to experiment both with strength of the results and also the computational feasibility on commodity hardware.

Multi-Layer Perceptron Classifier This model is a multi-layer Perceptron-based neural network. The model optimizes the log-loss function using stochastic gradient descent or a quasi-Newton method called limited memory BFGS, and the model is trained iteratively (at each step the partial derivatives of the loss function are recomputed to update the parameters). Scikit-learn offers a robust implementation. Using a logistic activation function allows us to get independent scores for each label; combining these scores with a cutoff lets us assign multiple labels to an input.

ML-KNN K-nearest-neighbors in the multilabel classification setting (ML-KNN) works similarly to KNN in the traditional classification setting. As in the single classification setting, the k-nearest neighbors are found for an unseen instance. Then, the ML-KNN algorithm uses the set of labels of the instance's neighbors to define a probability score for each label [2]. This algorithm is also implemented in Scikit-learn and supports multilabel classification "out of the box."

Radius neighbors classifier Radius Neighbors classifier will classify an instance based on all neighbors that fall into a certain radius, and then use the same technique as KNN to score labels.

Random forest Random forests are suitable for multilabel classification since they can produce probabilities for all known classes. A threshold, tuned via cross-validation, can be used to binarize the probability estimates.

Classifier chains Classifier chains are commonly used in multilabel setting though they may prove too computationally expensive for the number of labels in our problem. They build a sequence of classifiers where downstream models take the output of upstream models as input.

FastXML The FastXML algorithm [3] was specifically designed for extreme classification problems, and has been ported to Python.[2] The model is based on a random forest approach which learns a hierarchical structure of labels to manage the size of the label space.

Experiments

To tune each of the above models on the training data for optimal performance, we will use a holdout set of the training data and perform k-fold cross-validation and grid-search on appropriate parameters, using LRAP (Label Ranking Average Precision) as our validation metric. We will most likely use $k = 5$ folds, since the training set is large enough that 80% of the data (around 12000 rows) is representative of the entire training set. Increasing the number of folds will increase the compute time to train and evaluate each model, so using too many folds will make model validation infeasible. After each model's parameters are tuned to achieve optimal validation performance, we will train each model on the entire training set. Then, we will use the provided "development set" to evaluate and compare performance between models using the designated metric, LRAP score.

[1] `http://www.ke.tu-darmstadt.de/resources/eurlex`

[2] `https://github.com/Refefer/fastxml`

References

[1] K. Bhatia et al. *The extreme classification repository: Multi-label datasets and code*. 2016. URL: `http://manikvarma.org/downloads/XC/XMLRepository.html`.

[2] Min-Ling Zhang and Zhi-Hua Zhou. "A k-nearest neighbor based algorithm for multi-label classification". In: *2005 IEEE International Conference on Granular Computing*. Vol. 2. 2005, 718–721 Vol. 2.

[3] Yashoteja Prabhu and Manik Varma. "FastXML: A Fast, Accurate and Stable Tree-classifier for eXtreme Multi-label Learning". In: *ACM – Association for Computing Machinery*. Aug. 2014. URL: `https://www.microsoft.com/en-us/research/publication/fastxml-a-fast-accurate-and-stable-tree-classifier-for-extreme-multi-label-learning/`.