Analyzing the Moving Parts of a Large-Scale Multi-Label Text Classification Pipeline: Experiences in Indexing Biomedical Articles

Anthony Rios* and Ramakanth Kavuluru*†‡
*Department of Computer Science, University of Kentucky, Lexington, KY
†Division of Biomedical Informatics, Dept. of Biostatistics, University of Kentucky, Lexington, KY
{anthony.rios1, ramakanth.kavuluru}@uky.edu

***Abstract*—Medical subject headings (MeSH) is a controlled hierarchical vocabulary used by the National Library of Medicine (NLM) to index biomedical articles. In the 2014 version of MeSH terminology there are a total of 27,149 terms. Librarians at the NLM tag each biomedical article to be indexed for the PubMed literature search system with terms from MeSH. This means the human indexers look at each article's full text and index it with a small set of descriptors, 13 on average, from over 27,000 descriptors available in MeSH. There have been many recent attempts to automate this process focused on using the article title and abstract text to predict MeSH terms for the corresponding article. There has also been an open automated biomedical indexing challenge, BioASQ [1], that started in 2013. The best general supervised learning framework in these challenges has been a pipeline with four different components: 1. pre-processing and feature extraction; 2. employing the binary relevance and/or nearest neighbor approaches to select a set of candidate terms; 3. ranking these candidate terms using corresponding informative features; and 4. applying label calibration to dynamically predict the number of top terms to be included in the final selection for the current instance. The specific details in how each of these components is implemented determines the performance variations of various entries in the challenge. In this paper, we analyze these moving parts of the MeSH indexing multi-label classification pipeline with experiments involving different combinations. Our best combination achieves $\approx 1\%$ increase in micro F-score compared with the top performing team across the five weeks of the final batch of the BioASQ 2014 challenge. The main take away from our efforts is that small improvements/modifications to different components of the pipeline can offer moderate improvements to the overall performance of the method. Our experiences show that, at least thus far, top performances have resulted mostly due to these improvements rather than drastic changes of the core methodology.**

I. Introduction

Indexing biomedical articles with medical subject headings (MeSH) is an important task that has significant impact on how researchers search and retrieve relevant information. This is going to be more essential in the future given the exponential growth of biomedical articles indexed by PubMed®, the main biomedical literature search system of the National Center for Biotechnology Information (NCBI). PubMed lets users search over 22 million biomedical citations available in the MEDLINE bibliographic database curated by the National Library of Medicine (NLM) from over 5000 leading biomedical journals in the world. To deal with the explosion of information on various topics, users rely on queries involving MeSH terms that are assigned to each biomedical article. This is because MeSH terms are assigned by librarians after considering the full text of an article and, as such, capture the semantic content of an article that cannot easily be captured by keyword or phrase searches. Once articles are indexed with MeSH terms, users can quickly search for articles that pertain to a specific subject of interest instead of relying solely on key word based searches. Besides this direct application, recent efforts also demonstrated that using the set of MeSH terms assigned to an article as its semantic proxy can be helpful in high level applications such as literature based knowledge discovery [2].

Indexing biomedical articles with MeSH terms involves human indexers' understanding of the article and their familiarity with the MeSH vocabulary. As such, the manual indexing task suffers from consistency issues and takes a significant amount of time leading to delays in the availability of indexed articles. Hence many recent efforts focused on assigning MeSH terms to biomedical articles to expedite this process. They typically use solely the abstract and title text of the articles (the citation text) given most full text articles are available based on paid licenses not subscribed by many researchers. Increased interest in automated indexing has led to the creation of the annual automated indexing challenge, BioASQ [1], where participants are asked to upload their predictions for a few thousand citations each week for a total of fifteen weeks divided into three batches of five weeks each. This paper describes our approach and results on the final batch of the 2014 BioASQ dataset [1] where we were able to outperform the best model by $\approx 1\%$ in micro F-score over the five weeks of the third and final batch. We primarily discuss our experiences in analyzing and fine tuning various moving parts of the multi-label classification framework used earlier by us [3] and other prior efforts including the top contenders in the challenge.

We outline the rest of the paper here. In Section II, we discuss prior work on predicting MeSH terms and also outline the basic approaches taken to that end. We subsequently elaborate the full multi-label classification pipeline we use

‡corresponding author

[1]We did not participate in the 2014 challenge; our analysis was conducted in a post-hoc fashion in Fall 2014. We participated in batches two and three of the recently concluded 2015 BioASQ challenge where our model is currently placed 2nd in second batch and 3rd in the final batch. The final results are not yet determined presumably because several citations have not yet been assigned terms at the NLM. Also, the participants and their methods are not disclosed and hence at this point we are not able to discuss these details.

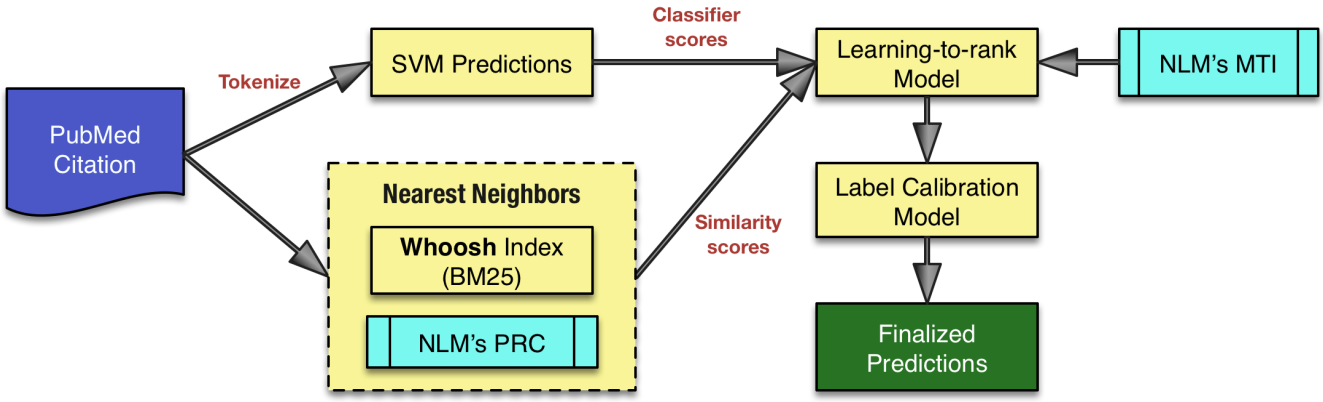

Fig. 1: High Level Overview of the MeSH Indexing Pipeline

for our experiments, including specific details of different components employed, in Section III. We present our results and lessons learned in Section IV and conclude with future research directions in Section V.

II. Background and Related Work

Indexing a biomedical article with multiple MeSH terms is an instance of the well known multi-label classification problem where multiple labels from fixed vocabulary need to be assigned for each instance. The large scale nature of indexing with MeSH terms comes due to two aspects: the large number of labels (over 27,000) and the label cardinality ≈ 13. Furthermore, Rubin et al. [4, Figure 1] also observe that MeSH is a power-law dataset with most terms having very few examples. Hence traditional approaches that seem to perform very well on multi-label learning problems do not perform well for MeSH indexing.

NLM initiated efforts in automatic MeSH term assignment with their medical text indexer (MTI) program that exploits terms from already tagged related citations in combination with named entity recognition (NER), unsupervised clustering, ad hoc indexing rules, and candidate term ranking heuristics in a pipeline [5]. MTI recommends MeSH terms for NLM indexers to assist in their efforts to expedite the indexing process.

All prior efforts on MeSH term assignment can be decomposed into following three distinct approaches:

1) The first depends on obtaining and mapping named entities from the input citation to MeSH concepts using knowledge based methods [5], [6]
2) The second relies on obtaining the top k already indexed nearest neighbors [3], [7] of the input instance (k-NN approach) and uses the learning-to-rank (L2R) approach with novel features to learn a function that ranks the MeSH terms from these neighbors.
3) the third relies on meta-learning to train custom binary classifiers [8] for each MeSH term and indexes the best performing model for each label to be applied on new citations. This is generally called the binary relevance approach where feature selection [9] and training data sampling [10] techniques can be separately applied to each term's classifier.

In general, state-of-the-art models (e.g., [11], winner of the 2014 BioASQ challenge) use a combination of all three approaches by combining the k-NN approach with the binary relevance approach, where candidates from a few 'close' neighbors are combined with top few predicted candidates from the binary relevance approach (based on classifier scores). This combined set of candidates is then ranked using L2R with a variety of features for each candidate that include neighborhood similarity scores from the k-NN approach and classifier scores from the binary relevance approach. Named entities and other knowledge based approaches are used as part of the features for L2R or for individual classifiers in the binary relevance approach. Finally, label calibration is used to select the top few terms in the ranked candidate list as the final set of predicted terms. Application of L2R followed by label calibration is especially important for power law datasets like MeSH where many labels have very few training examples. The general framework we use in this paper is along the same lines but we elaborate the specific details in the next section.

III. MeSH Indexing Pipeline

As we see from the high level schematic of the multi-label classification pipeline in Fig 1, it essentially uses binary relevance and nearest neighbors components to come up with a set of candidate terms that are ranked using the L2R model following by a calibration model that picks an appropriate number of top terms to be retained as the set of predicted terms for the input citation. Next, we elaborate on different options for each of the components.

A. Tokenization

This is essential for training the binary classifiers for each MeSH term. We experiment with three different tokenizers. First, we use the standard regular expression for tokenization available in the Scikit-Learn [12] machine learning framework: "(?u)\b\w\w+\b". The second is the Natural Language Toolkit [13] (NLTK) implementation of the Penn Treebank

(PTB) tokenizer, which is a more complex set of regular expressions expected to yield better tokenization. The third tokenizer was specifically built for biomedical text by Jiang and Zhai [14]. For this tokenizer we use a custom implementation which applies the heuristic rules, break point set one, and join-normalization from [14]. Table I provides sample text fragments that typically occur in biomedical text and the tokens obtained using each of the three different tokenization methods. Each token from a fragment is separated by a comma and escaped commas mean the comma is part of the token. Note that we remove single character tokens. We note that the biomedical tokenizer nicely groups the characters for the "hba-1c" test and "mip-1alpha" protein as single tokens, while the other two tokenizers split them. The PTB tokenizer retains certain characters like hyphens and slashes as part of tokens while the Scikit-Learn tokenizer ignores them in the tokens.

TABLE I: Sample outputs of the Scikit-Learn, NLTK PennTreebank, and biomedical tokenizers. Each token is separated by a comma. escaped commas mean the comma is part of the token. Note that we remove single character tokens

Fragment 1	(HbA(1c) >=6% or GO >=1.26 g/L)
Scikit-Learn	hba, 1c, or, go, 26
NLTK PennTreebank	hba, 1c, =6, or, go, =1.26, g/l
Biomedical Tokenizer	hba1c, or, go, 1.26, gl
Fragment 2	(MIP)-1alpha, pRB/p105
Scikit-Learn	mip, 1alpha, prb, p105
NLTK PennTreebank	mip, -1alpha, prb/p105
Biomedical Tokenizer	mip-1alpha, prbp105
Fragment 3	2',5'-linked 3'-deoxyribonucleotides
Scikit-Learn	linked, deoxyribonucleotides
NLTK PennTreebank	2'\,5'-linked, 3'-deoxyribonucleotides
Biomedical Tokenizer	2'\,5'-linked, 3'-deoxyribonucleotides
Fragment 4	(Langerhan's cells)
Scikit-Learn	langerhan, cells
NLTK PennTreebank	langerhan, 's, cells
Biomedical Tokenizer	langerhan, cells

B. Candidate MeSH Terms Generation

We take advantage of three different classifiers including a linear support vector machine (SVM), k-NN using the BM25 model [15, Chapter 11.4], and k-NN using PubMed Related Citations (PRC[2] [16]). It is well known that linear SVMs are one of the top performing algorithms for text classification. To use the SVM classifier we use the binary relevance transformation. Let T be the set of labels and let $q = |T|$. Binary relevance learns q binary classifiers, one for each label in T. It transforms the dataset into q separate datasets. For each label T_j, we obtain the dataset for the corresponding binary classifier by considering each document–label-set pair $(D_i, \mathbf{Y}_i)$ and generating the document-label pair (D_i, T_j) when $T_j \in \mathbf{Y}_i$ and generating the pair $(D_i, \neg T_j)$ when $T_j \notin \mathbf{Y}_i$. It is standard practice to make predictions based on the output of the SVM if the confidence score returned is greater than zero. However, because of the extreme class imbalance scenarios we face with MeSH terms, this simply does not work; for most low frequency terms the simpler approach would lead to many false negatives. So when predicting in this situation, the labels are ranked based on their score output by the corresponding binary classifiers and the top m labels are considered as the predicted set for a suitably chosen m (more later).

In the k-NN approach, we first fetch k instances D_i, $i = 1 \ldots k$, in the training dataset that are content-wise most similar to the test instance I. Hence, we assume that most correct labels for I are going to be in the neighborhood

$$\mathcal{N}_k(I) = \bigcup_{i=1}^{k} G(D_i), \tag{1}$$

where $D_i, \ldots, D_k$ are the k nearest neighbors and $G(D_i)$ is the set of correct labels for the training instance D_i. However, $\mathcal{N}_k(I)$ may be very large compared with the number of labels assigned per instance (which is in the range of 13-15 MeSH terms for our current problem). Hence, a ranking on labels in $\mathcal{N}_k(I)$ is imposed based on the document similarity scores of I with each D_i. Subsequently, just like the binary relevance approach the top m labels are retained as the predicted set for a suitably chosen m. In this study, we explore two different approaches to obtained nearest approaches, the first using the BM25 model implemented in the Whoosh[3] Python framework and the second using NLM's PRC model.

As shown in Fig 1 we actually combine the candidates generated by both the binary relevance approach and the k-NN approaches. Specifically, we select the top 100 terms from the binary relevance approach based on SVM classifier scores. The final candidate set is generated by combining (set union) these 100 terms with the set of terms from the top 10 training set neighbors with the BM25 model and top 25 related citations with the PRC model.

C. Ensemble Modeling through Learning-to-Rank

L2R [17] was originally introduced for information retrieval to rank a set of documents for an input search query based on various features that measure relevance of the document to a given query. Here we adapt this approach to rank MeSH terms generated in Section III-B. L2R follows a supervised approach and in its training phase, takes as input a training dataset of PubMed citations and the corresponding ranked lists of MeSH terms:

$$\{(D_i, \mathcal{R}(T_i)) : i = 1, \ldots, n\}, \tag{2}$$

where D_i is an input citation, T_i is the set of candidate MeSH terms associated with D_i (generated as in Section III-B), $\mathcal{R}(T_i)$ is the ground truth ranking on the terms for D_i, and n is the size of the training dataset. We essentially rank all correct terms above all other irrelevant terms to generate $\mathcal{R}$ for training. The algorithm then learns a ranking model that minimizes an appropriate loss function that pertains to the ranking. We note that the training process extracts features $f_j(T_i^r)$, where $r = 1, \ldots, l(i)$, for each term $T_i^r \in T_i$ where $l(i)$ is the number

[2]Although PRC is not a traditional information retrieval based k-NN implementation, at a high level it essentially obtains closely related neighbors.

[3]https://bitbucket.org/mchaput/whoosh

of terms provided to the i-th citation instance for training and j is the feature index. The features for the MeSH terms are heavily based on the specific input citation to tightly constrain the ranking based on information available in it. Finally, given a new test citation and a set of candidate terms as input, the learned model imposes a ranking on the terms based on citation specific MeSH term features. For our effort, L2R can be thought of as an ensemble modeling approach similar to stacking. This means that the features used for learning-to-rank are the scores obtained by the individual classifiers described in Section III-B and additionally the MTI output for the input instance. We used the LambdaMART [18] method for L2R and maximized the expected reciprocal rank [19] function. Next, we describe the features used for each candidate MeSH term.

The first feature we use is based on the classifier score returned by the SVMs trained for each term. Specifically, it is

$$f^C(t, I) = \frac{1}{1 + e^{-H(t,I)}},$$

where $H(t, I)$ is the score returned on input citation I by the SVM trained for MeSH term t. We use the sigmoid function to normalize the scores to a $[0, 1]$ range, a standard practice when dealing with SVMs and is important depending on the specific L2R algorithm used.

For both of the k-NN methods we create neighborhood features. For each citation neighbor D_j for a test instance I, we also have the corresponding similarity score $S(D_j, I)$, which is essential to find the nearest neighbors in the first place (using either BM25 or NLM's PRC). Using these similarities, for a given candidate term $t \in \mathcal{N}_k(I)$, we compute the neighborhood feature as the sum

$$f^N_k(t, I) = \frac{1}{Z(I)} \sum_{D_j : t \in G(D_j)} S(D_j, I),$$

where $D_1, \ldots, D_k$ are the nearest neighbors of I and $G(D_j)$ are the correct MeSH terms for training citation D_j as in Equation 1 and $Z(I)$ is the normalization constant

$$Z(I) = \sum_{t \in \mathcal{N}_k(I)} \sum_{D_j : t \in G(D_j)} S(D_j, I).$$

From MTI we don't have a score, rather we have a binary indicator variable representing if a given label was predicted by MTI. Let $M(I)$ be the set of MeSH terms predicted by MTI for citation I. We use the binary feature

$$f^M(t, I) = \begin{cases} 1, & \text{if } t \in M(I); \\ 0, & \text{otherwise.} \end{cases}$$

D. Label Calibration

We transform the multi-label classification problem to an L2R problem and end up with a ranking based on the scores of the final output of the methods discussed in Sections III-B and III-C. In order to predict an appropriate small set of labels from among all candidates for each input citation, our goal is to pick the top m labels in the final L2R ranking for each citation so that we can optimize the micro F-score used for our final evaluation. In this section, we outline label calibration methods we used for this purpose.

We experiment with four methods: RCut [20], MetaLabeler [21], OneThreshold [22], and finally we try a combination of MetaLabeler and OneThreshold approaches. Before we proceed, let $\mathcal{P}(I)$ be the ranked list of candidates output by L2R for input instance I, $\mathcal{P}^m(I)$ be the set of top m terms in $\mathcal{P}(I)$, and $\mathcal{F}(I)$ be the finalized list of candidates predicted for I based on label calibration. Thus $\mathcal{F}(I)$ is set used in computing performance measures for the entire pipeline.

The finalized predications using rank based cut (RCut) for an instance I is $\mathcal{F}(I) = \mathcal{P}^m(I)$, where $m > 0$ is fixed global value chosen such that it maximizes the micro F-score based on a validation dataset. This means every instance I will have exactly m labels in the final prediction.

OneThreshold builds on RCut by making it possible to have a different number of labels for each instance depending on the scores returned by the L2R method. For each instance I, we have

$$\mathcal{F}(I) = \{x \in \mathcal{P}(I) : \lambda(x, I) \geq p\}, \tag{3}$$

where p is a threshold value optimized over a validation dataset to get the best micro F-score and $\lambda(x, I)$ is the L2R score of the candidate MeSH term x for I.

The third method we experiment with is the use of a MetaLabeler where we learn a regression model with the same n-gram features used to train the SVM. However, instead of trying to predict a MeSH term for a given instance, a linear regression model f predicts the appropriate number of labels for I. Thus we have

$$\mathcal{F}(I) = \{x \in \mathcal{P}^{f(I)}(I)\}. \tag{4}$$

It is straightforward to see that RCut is very restrictive in simply selecting a fixed global number of terms given in reality the number of terms varies with each citation. Both MetaLabeler and OneThreshold counter this weakness by dynamically selecting the number of terms based on the input instance using different approaches. Although MetaLabeler adapts the number of terms to each instance, it might inadvertently also select those that have extremely low L2R scores leading to false positives. Hence we also experiment with a combination of both these approaches and select

$$\mathcal{F}(I) = \{x \in \mathcal{P}^{f(I)}(I) : \lambda(x, I) \geq p\},$$

where definitions of p, $f()$, and $\lambda()$ are those used earlier in equations 3 and 4.

IV. Experiments and Results

We conducted separate experiments to analyze the impact of choices made in various components of the pipeline. We used a dataset of 1.2 million[4] biomedical citations for training the SVM classifiers and indexed around 11 million citations for obtaining the BM25 neighbors. The testing for each component is done on the combined set of citations from all five weeks of the batch three of the 2014 BioASQ challenge. We ensured that training and validation sets used for the SVM, k-NN, and MetaLabeler are mutually exclusive from the training and

[4]The challenge does not impose any constraints on the number of training examples to be used. Given one needs to train over 27,000 classifiers, for practical reasons most teams including ours chose about a million.

validation datasets used for L2R. There is also no overlap between the test set and all the training and validation datasets used in the pipeline given our training datasets were subsets of the already index 2014 PubMed baseline data (released in Dec 2013) and the challenge was exclusively on citations that are not already indexed in PubMed.

For evaluation we use the popular micro F-score which was the main scoring measure in the challenge. The label-based micro precision, recall, and F-score are defined as

$$P^{mic} = \frac{\sum_{j=1}^{|T|} TP_j}{\sum_{j=1}^{|T|}(TP_j + FP_j)},\ R^{mic} = \frac{\sum_{j=1}^{|T|} TP_j}{\sum_{j=1}^{|T|}(TP_j + FN_j)},$$

$$\text{and Micro-F} = \frac{2P^{mic} \cdot R^{mic}}{P^{mic} + R^{mic}},$$

where TP_j, FP_j, and FN_j are true positives, false positives, and false negatives, respectively, of label T_j. It is important to note that micro F-score inherently weights more frequent terms higher than infrequent terms. So if frequently occurring terms are predicted with high accuracy, we will end up with a better micro F-score even if the performance is not very high for several infrequent labels.

A. Tokenization Comparison

Table II presents our results with the three tokenizers discussed in Section III-A. These are obtained by first using the binary relevance approach and training SVMs with the tokenized citations and using the LibLinear [23] SVM implementation from Scikit-Learn [12]. MetaLabeler is then applied to obtain the final predictions in the test set. The classifiers were trained using both unigrams and bigrams removing infrequent n-grams that occurred fewer than six times. As can be noticed from the table, the simpler Scikit-Learn tokenizer performed much better than the biomedical tokenizer or the PTB tokenizer. This was counterintuitive to us given the biomedical tokenizer seemed to collapse similar looking tokens that are semantically equivalent in the domain. Although it might be more useful for NER, for text classification simpler approaches seemed to work better. This may be because simpler approaches like the Scikit-Learn tokenizer tend to prefer smaller but, nevertheless, meaningful tokens (e.g., see fragment 2 in Table I) that are more common, than longer and rare tokens that can lead to major sparsity issues in the feature space. The main takeaway for us was that the choice of the tokenizers can greatly influence the performance of multi-label text classification pipelines.

TABLE II: Tokenizer performance comparison

Tokenizer	P^{mic}	R^{mic}	Micro-F
Scikit-Learn	**0.6037**	**0.5985**	**0.6011**
NLTK PennTreebank	0.5508	0.5689	0.5597
Biomedical Tokenizer	0.5510	0.5691	0.5599

B. Individual Classifier Comparison

In this section we compare the individual classifier performances for SVMs, k-NN (BM25 and PRC), and default MTI. Both the k-NN and SVM methods use MetaLabeler for obtaining the final predictions. The SVM classifiers were trained as discussed in Section III-A. The NLM PRC neighbors and scores were gathered using the NLM Entrez programming Utilities implemented in Biopython [24]. The default MTI scores are obtained using the MTI predictions made available on the BioASQ challenge website. We used the default parameters of $b = 0.75$ and $k_1 = 1.2$ (see [15, Chapter 11.4.3]) for the BM25 approach.

The results are shown in Table III where SVM based prediction significantly outperforms other approaches. We also see that NLM's PRC is a superior approach to getting related citations whose terms are more relevant to the input instance compared with the BM25 based model. We believe PRC's superior performance is mostly due to the so called "local weights" that are assigned to common terms between two citation texts; this was also observed in the original paper on PRC [16]. It is not surprising to see MTI doing better than PRC since PRC is used as a component in MTI (although the details of how it is incorporated is not a simple neighbor term set union).

TABLE III: Classifier performance comparison

Classifier	P^{mic}	R^{mic}	Micro-F
SVM	**0.6037**	**0.5985**	**0.6011**
BM25 k-NN	0.3576	0.3545	0.3560
NLM's PRC	0.5304	0.5295	0.5282
Default MTI	0.5923	0.5548	0.5729

C. Label Calibrator Comparison

To compare the different label calibration methods, we used the output from the final L2R algorithm trained on all the features discussed in Section III-C. Both RCut and OneThreshold were optimized using a held out validation dataset of around 10,000 citations. The MetaLabeler was trained on the same 1.2 million documents used for training the SVM classifiers. From Table IV we see that RCut underperforms significantly given each article must be labeled with the exact same number of labels. We also see that using MetaLabeler with OneThreshold is the best performing method for reasons discussed in Section III-D. However, there is a clear loss of 2.5% in recall with the combined method although the gain in precision more than makes up for it in the context of improvement in F-score.

TABLE IV: Label calibrator performance comparison

Calibrator	P^{mic}	R^{mic}	Micro-F
RCut	0.6150	0.6048	0.6099
OneThreshold	0.6233	**0.6217**	0.6225
MetaLabeler	0.6267	**0.6215**	0.6241
MetaLabler+OneThreshold	**0.6664**	0.5949	**0.6286**

D. Overall Results

The goal of this section is to compare our methods with the top scorers in the 2014 BioASQ challenge. Our ensemble is the best performing combination of different components in the pipeline (discussed in Section III) over a validation dataset. It includes

1) using the Scikit-Learn tokenizer,
2) combining candidates from the SVM classifiers and the BM25 and PRC nearest neighbors,
3) using classifier scores, neighbor similarity scores, and MTI Boolean L2R features, and finally
4) using the combination of MetaLabeler and OneThreshold for label calibration.

Table V shows our scores along with the top two performing teams' scores for every week of the third and final batch. We chose the final batch given improvements are normally made in the first two batches. We see that our ensemble consistently outperforms other top teams. A noteworthy aspect here is that the high level frameworks used by us and these other teams are similar and gains seems to be coming from variations in each of the components. This is our primary motivation for writing this paper – to convey our experiences in identifying and fine tuning different moving parts of the classification pipeline and to demonstrate that this process can lead to better performing models. In our case, we believe, the selection of the tokenizer, the choices of features (including number of neighbors in k-NN) for L2R, and the combination of OneThreshold and MetaLabeler for label calibration are responsible for performance gains. There are a total of 18,256 citations in the batch three test set (combining all five weeks). Hence a 1% performance gain typically means $\approx$ 2300 more terms are correctly handled (FPs removed or FNs recovered) given there 13 terms on average per citation. However, it is possible that some teams focus more on recall (which is reasonable given this is to assist human coders) and hence their F-score may not be as high even though they have reasonably decent precision. Given the large test set size (over 18,000 citations), we believe our performance improvements will persist over new citations. Building confidence intervals using bootstrapping is prohibitive (in the context of this conference paper) given the need to repeatedly train over 27,000 classifiers.

TABLE V: Comparison of our micro-F with the top two teams' scores in the 2014 BioASQ challenge

Team	week 1	week 2	week 3	week 4	week 5
Fudan	0.6001	0.6079	0.6302	0.6317	0.6221
NCBI	0.5885	0.5992	0.6255	0.6192	0.6077
UKY	**0.6181**	**0.6213**	**0.6366**	**0.6397**	**0.6304**

In Table VI we show ablation results after removing an L2R feature or a component of our label calibration method. We can see that removing the SVM score feature results in the most significant performance drop. This is not unexpected since SVM outputs give the most accurate ranking over all labels (first row of Table III). It is well established that L2R features should be not only informative but also diverse in how they rank terms. In this case, the PRC neighbors cause the next biggest drop in performance and the BM25 neighbors cause a negligible drop and might not be contributing significant complementary information in the presence of SVM and PRC based candidates with their corresponding scores. From the table we notice that the impact of a classifier in the full model aligns with its performance in isolation except for the default MTI inputs. There are two possible explanations for the slight increase in performance after removing it. First, it is possible that everything it gets correct is covered by the SVMs and the rest is noise. However, a more likely explanation may have to do with the nature of it being a binary feature. While the other features/classifiers give a ranking over the labels based on their confidence in predicting a certain MeSH term, MTI treats everything equally, and hence a sense of gradation is lost when it is used as an L2R feature. In our validation dataset experiments, however, MTI inclusion gave the best micro-F, and hence we included it in the final ensemble.

TABLE VI: Analysis of Ablation of Different Features (L2R and Label Calibration)

	P^{mic}	R^{mic}	Micro-F
ALL	**0.6664**	0.5949	0.6286
– SVM	0.6045	0.4796	0.5348
– BM25 k-NN	0.6645	0.5909	0.6255
– NLM's PRC	0.6472	0.5789	0.6112
– Default MTI	0.6589	0.6046	**0.6306**
– OneThreshold	0.6267	**0.6215**	0.6241
– MetaLabeler	0.6233	**0.6217**	0.6225

V. Conclusion

Large scale multi-label text classification with thousands of labels and label distributions following a power law is a challenging problem. Indexing biomedical articles with MeSH terms is one such problem with important practical consequences, specifically to expedite and improve consistency in indexing tasks at the NLM. Started in 2013, the BioASQ challenge presents an excellent format to go beyond smaller subsets of MeSH and focus on system level analysis of automated MeSH term assignment to biomedical citations. The top performer in the 2013 challenge had a micro F-score of 0.58, which improved to 0.63 in 2014, and as of now is at 0.65 (for some weeks) in 2015.

We conducted several experiments to prepare for competing in the BioASQ challenge having had prior results in automated MeSH term assignment [3], [6] on smaller datasets. We learned that significant fine tuning and ensembling is needed to scale to larger test datasets and improve performance. In this paper, we present the results of our experiments in analyzing various components of the full pipeline to predict MeSH terms. We found that choosing a good tokenizer is critical to build effective binary classifiers whose scores form a critical feature type in learning a good L2R model. NLM's PRC citations also form

a critical component in identifying similar, already indexed citations that generate candidate terms and effective relatedness scores that are effective L2R features. We also learned that trading off recall with label calibration can improve precision to an extent that can result in an overall increase in micro F-score. We believe our results are also applicable, at a general level, to other large scale multi-label learning problems such as assigning diagnosis codes to electronic medical records and international patent classification (IPC) codes to patents.

An important area of potential improvements that has not received much attention in the BioASQ challenge so far is to exploit label correlations among MeSH terms. Intuitively, similar themes in articles naturally need to be tagged with related groups of MeSH terms. Besides the MeSH hierarchy, we also have MeSH terms participating in non-hierarchical associative relations when they are mapped to unified medical language system (UMLS) concepts in the UMLS Metathesaurus[5]. Existing methods [25] that take advantage of the structure among labels for multi-label datasets are yet to be extended to MeSH. Distributional correlations between MeSH terms can also be observed in the output term sets of millions of citations in the training dataset. Modeling label correlations have been shown to improve classifier performance [26]; however the large vocabulary size and the power law nature of MeSH indexing makes it much harder to exploit correlations when compared with other datasets. It is, nevertheless, a ripe area of research to scale classification methods that leverage label correlations to large label spaces.

ACKNOWLEDGEMENTS

Many thanks to anonymous reviewers for their careful reviews and comments, which helped improve the manuscript. We are grateful to the BioASQ challenge organizers [1] for facilitating a platform to work on large scale multi-label classification. This publication was supported by the National Center for Research Resources and the National Center for Advancing Translational Sciences, US National Institutes of Health (NIH), through Grant UL1TR000117. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

REFERENCES

[1] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M. R. Alvers, D. Weissenborn, A. Krithara, S. Petridis, D. Polychronopoulos *et al.*, "An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition," *BMC bioinformatics*, vol. 16, no. 1, p. 138, 2015.

[2] D. Cameron, R. Kavuluru, T. C. Rindflesch, A. P. Sheth, K. Thirunarayan, and O. Bodenreider, "Context-driven automatic subgraph creation for literature-based discovery," *Journal of biomedical informatics*, vol. 54, pp. 141–157, 2015.

[3] R. Kavuluru and Y. Lu, "Leveraging output term co-occurrence frequencies and latent associations in predicting medical subject headings," *Data & Knowledge Engineering*, vol. 94, no. Part B, pp. 189–201, 2014.

[4] T. N. Rubin, A. Chambers, P. Smyth, and M. Steyvers, "Statistical topic models for multi-label document classification," *Machine Learning*, vol. 88, no. 1-2, pp. 157–208, 2012.

[5] A. Aronson, J. Mork, C. Gay, S. Humphrey, and W. Rogers, "The NLM indexing initiative's medical text indexer," in *Proceedings of MEDINFO*, 2004, pp. 268–272.

[5] http://www.ncbi.nlm.nih.gov/books/NBK9684/

[6] R. Kavuluru and Z. He, "Unsupervised medical subject heading assignment using output label co-occurrence statistics and semantic predications," in *Natural Language Processing and Information Systems*, ser. NLDB. Springer, 2013, pp. 176–188.

[7] M. Huang, A. Névéol, and Z. Lu, "Recommending MeSH terms for annotating biomedical articles," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 660–667, 2011.

[8] A. Jimeno-Yepes, J. G. Mork, D. Demner-Fushman, and A. R. Aronson, "A one-size-fits-all indexing method does not exist: Automatic selection based on meta-learning," *Journal of Computing Science and Engineering*, vol. 6, no. 2, pp. 151–160, 2012.

[9] M. Yetisgen-Yildiz and W. Pratt, "The effect of feature representation on medline document classification," in *Proceedings of AMIA Symposium*, vol. 2005. American Medical Informatics Association, 2005, pp. 849–853.

[10] S. Sohn, W. Kim, D. C. Comeau, and W. J. Wilbur, "Optimal training sets for bayesian prediction of MeSH assignment," *Journal of the American Medical Informatics Association*, vol. 15, no. 4, pp. 546–553, 2008.

[11] K. Liu, J. Wu, S. Peng, C. Zhai, and S. Zhu, "The fudan-uiuc participation in the BioASQ challenge task 2a: The antinomyra system," *Proceedings of Question Answering Lab at the Conference and Labs of the Evaluation Forum (CLEF)*, 2014.

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

[13] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[14] J. Jiang and C. Zhai, "An empirical study of tokenization strategies for biomedical information retrieval," *Information Retrieval*, vol. 10, no. 4-5, pp. 341–363, 2007.

[15] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[16] J. Lin and W. J. Wilbur, "Pubmed related articles: a probabilistic topic-based model for content similarity," *BMC Bioinformatics*, vol. 8, no. 1, p. 423, 2007.

[17] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[18] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao, "Adapting boosting for information retrieval measures," *Information Retrieval*, vol. 13, no. 3, pp. 254–270, 2010.

[19] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, "Expected reciprocal rank for graded relevance," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 621–630.

[20] Y. Yang, "A study of thresholding strategies for text categorization," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2001, pp. 137–145.

[21] L. Tang, S. Rajan, and V. K. Narayanan, "Large scale multi-label classification via metalabeler," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 211–220.

[22] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, dec. 2008, pp. 995 –1000.

[23] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[24] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski *et al.*, "Biopython: freely available python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[25] W. Bi and J. T. Kwok, "Multi-label classification on tree-and DAG-structured hierarchies," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 17–24.

[26] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, pp. 335–359, 2011.