2015

APPLICATION OF RANDOM INDEXING TO MULTI LABEL CLASSIFICATION PROBLEMS: A CASE STUDY WITH MESH TERM ASSIGNMENT AND DIAGNOSIS CODE EXTRACTION

Yuan Lu
University of Kentucky, yuan.lu@uky.edu

APPLICATION OF RANDOM INDEXING TO MULTI LABEL CLASSIFICATION PROBLEMS: A CASE STUDY WITH MESH TERM ASSIGNMENT AND DIAGNOSIS CODE EXTRACTION

THESIS

A thesis submitted in partial
fulfillment of the requirements for
the degree of Master of Science in
the College of Engineering at the
University of Kentucky

By
Yuan Lu
Lexington, Kentucky

Director: Dr. Ramakanth Kavuluru, Professor of Biostatistics and Computer Science
Lexington, Kentucky

2015

ABSTRACT OF THESIS

APPLICATION OF RANDOM INDEXING TO MULTI LABEL CLASSIFICATION PROBLEMS: A CASE STUDY WITH MESH TERM ASSIGNMENT AND DIAGNOSIS CODE EXTRACTION

Many manual biomedical annotation tasks can be categorized as instances of the typical multi-label classification problem where several categories or labels from a fixed set need to assigned to an input instance. MeSH term assignment to biomedical articles and diagnosis code extraction from medical records are two such tasks. To address this problem automatically, in this thesis, we present a way to utilize latent associations between labels based on output label sets. We used random indexing as a method to determine latent associations and use the associations as a novel feature in a learning-to-rank algorithm that reranks candidate labels selected based on either k-NN or binary relevance approach. Using this new feature as part of other features, for MeSH term assignment, we train our ranking model on a set of 200 documents, test it on two public datasets, and obtain new state-of-the-art results in precision, recall, and mean average precision. In diagnosis code extraction, we reach an average micro F-score of 0.478 based on a large EMR dataset from the University of Kentucky Medical Center, the first study of its kind to our knowledge. Our study shows the advantages and potential of random indexing method in determining and utilizing implicit relationships between labels in multi-label classification problems.

KEYWORDS: MeSH, ICD-9, Random Indexing, Learning-to-Rank, Coordinate Ascent, Random Forest, Information Retrieval

Author's signature: Yuan Lu

Date: February 25, 2015

APPLICATION OF RANDOM INDEXING TO MULTI LABEL CLASSIFICATION PROBLEMS: A CASE STUDY WITH MESH TERM ASSIGNMENT AND DIAGNOSIS CODE EXTRACTION

By
Yuan Lu

Director of Thesis: Ramakanth Kavuluru

Director of Graduate Studies: Miroslaw Truszczynski

Date: February 25, 2015

ACKNOWLEDGMENTS

First, I would express my deepest gratitude and appreciation to my research advisor, Dr. Ramakanth Kavuluru. His wide knowledge and insightful understanding of biomedical information retrieval lighted this research process. Without him, I never could have accomplished this thesis for my master's degree and published a paper at Data and Knowledge Engineering.

I would also thank my committee members, Dr. Judy Goldsmith and Dr. Miroslaw Truszczynski for their participating at my thesis defense. In addition, I would like to extend my thanks to Dr. Raphal Finkel, former DGS of Computer Science, Dr. Matthew Beck at Materials Science and Engineering Department and Yaowei Zhang.

Finally, the most special thanks go to Tom Priddy, Dr Wanhong Wang and Matt Dixon. They are my colleges at Weather Center at College of Agriculture where I spent three years, as a part time web developer. I started my professional career from this job and am very grateful for the position they offered me.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

In multi-label classification each instance is typically assigned multiple labels or categories. Thus it is a variant of the traditional multi-class classification problem where exactly one of several labels is selected for each input instance. It arises in a variety of domains, such as text classification, scene and video classification, and in biomedical informatics in the context of assigning standardized codes to textual narratives. For example, a movie focused on the history and culture of native Americans can be be classified into both categories *history* and *nativeAmericans*. In biomedical information retrieval, MeSH term assignment and diagnosis code extraction can also be mapped as multi-label classification problems the details of which are explained a little later in this section. First we discuss the general approaches to multi-label classification with a focus on classifying textual documents.

1.1 General Approaches to Multi-label Classification

General approaches to multi-label classification can be categorized as two different groups.

1. The first is known as the k nearest neighbor (or k-NN) approach. Given a new document to be classified, k-NN approach searches for the k "nearest" documents (for some k) among those already classified documents in the training dataset. The nearness or similarity is determined based on the comparison between the contents of the new document and all pre-classified training documents. Then the union of all label sets assigned to these k nearest documents will form the set of candidate labels for the new document. The confidence score of a specific label in this list is usually the sum of the similarity scores of all k nearest documents tagged with that label. After that, these candidate labels will be ranked by certain ranking criteria (besides the obvious confidence score based ranking) and the top few labels in this ranked list are selected as the final labels for the new document.

2. The second approach utilizes machine learning algorithms to build binary classifiers based on training data for each label from the list of all possible labels. Each new document is subject to all binary classifiers and all labels for which the corresponding binary classifiers output a positive response are treated as final set of labels for that instance. If too many or too few candidates are found, an addition ranking mechanism is typically applied.

Both approaches have advantages and disadvantages and are favored based on the situation (more on this later) and are at times used in a complementary hybrid fashion. In our experiments for MeSH term assignment we use the k-NN approach (Section 1.2) while in the research of diagnosis code extraction we employ binary classifiers approach (Section 1.3).

1.2 MeSH Term Assignment

Medical Subject Headings (MeSH) is designed by the US National Library of Medicine (NLM) for the purpose of speeding up the process and improving the accuracy of retrieving information on biomedical topics, specifically to annotate biomedical citations indexed by PubMed, NLM's search system for biomedical literature. MeSH is a controlled vocabulary and has a hierarchical structure with over 27000 terms at the time of this writing. The assignment of MeSH terms to articles is a routine task of NLM. Trained indexers view the full text of an article, capture its semantic content, and find the most appropriate MeSH terms to be assigned to the article. Given purely manually assignment is extremely time consuming, NLM's Medical Text Indexer (MTI) software was designed to suggest terms that can be further examined by human indexers before finalizing assignment. The MeSH term assignment task can be viewed as multi-label classification when we treat biomedical citations as documents and MeSH terms as labels. In our work, we utilize a k-NN algorithm to obtain the nearest 50 neighbors of citations in training set and create a list of candidate MeSH terms for each citation. Based on some selected features (more later), we rank the candidate MeSH terms by employing *learning-to-rank* methods and choosing the top 25 terms for the final recommendation. Our approach results in better precision, recall, and mean average precision (MAP) over the best published results at the time of this writing on two public datasets. In our work, we only considered the abstracts and titles without reviewing of full text given full text licenses are not available for all journals.

1.3 Diagnosis Code Extraction

We also apply our methods to extract diagnosis codes from electronic medical records (EMRs). Diagnosis codes are assigned to medical records in healthcare facilities by trained coders by reviewing all physician authored documents associated with a patient's visit. This is a necessary and complex task involving coders adhering to specific coding guidelines stipulated by the National Center for Health Statistics and the Centers for Medicare and Medicaid Services [1]. Codes are expected to record all assignable codes. With the popularity of EMRs, computational approaches to code assignment have been proposed in the recent years. The US healthcare system derives the ICD-9/10-CM coding terminology from the International Classification of Diseases (ICD). ICD-9 codes contain 3 to 5 digits and are organized hierarchically: they take the form `abc.xy` where the first three character part before the period `abc` is the main disease category, while the `x` and `y` components represent subdivisions of the `abc` category. For example, the code `530.12` is for the condition *reflux esophagitis* and its parent code `530.1` is for the broader condition of *esophagitis* and the three character code `530` subsumes all *diseases of esophagus*. Any allowed code assignment should at least assign codes at the category level (that is, the first three digits). At the category levels there are nearly 1300 different ICD-9 codes. Since all textual narratives are available (unlike for the MeSH task where only abstracts are available), we build binary classifiers for each ICD-9 code, process the new EMR with all binary classifiers to generate candidate ICD-9 codes. The candidates are then ranked using a learning-to-rank algorithm and finally top few codes are selected as the assigned codes.

1.4 Our Contributions

For both MeSH term and diagnosis code assignment, a candidate list is obtained which is further ranked using learning-to-rank approaches. Our main contribution is the design of novel features to be used in the candidate label ranking models. Based on output label sets from training dataset, we compute explicit relatedness based on co-occurrences frequencies and implicit relatedness based on *random indexing*. These label relatedness measures that consider label correlations are used in addition to some well known features that only measure the association between labels and documents. Both random Indexing and co-occurrence features model meaningful connections between labels under the assumption that if one label is closely related to another, there is non-trivial chance that both of them are assigned to the same document (depending on some other factors). Our central hypothesis is that incorporating label relatedness based on output label sets into the ranking process will help the final ranking and the classification performance. In our experiments for both terminologies, we demonstrate the validity of this hypothesis.

Chapter 2 Background

This chapter focuses on background information on methods used for our experiments. We first describe Random Indexing (RI) as way to compute implicit relatedness between terms based on co-occurrences. We also elaborate on its variants Term-based Reflective Random Indexing (TRRI) and Document-based Reflective Random Indexing(DRRI). For detailed information of RI, please refer to the paper by Trevor Cohen [8]. We use random indexing scores as novel features in a learning-to-rank framework. So we also provide details on Coordinate Ascent and Random Forest approaches to learning to rank. For further information of learning to rank, please refer to paper by Liu [15] and Breiman [4].

2.1 Random Indexing

RI is a technique that discovers nearest neighbors and implicit connections among "terms" by deriving vector representations for each term [8]. There are a few more conventional and established distributional methods such as Latent Semantic Analysis (LSA) [14] that employ computationally demanding Singular Value Decompositions (SVDs). RI has emerged as an alternative approach to LSA with much less computational burden and is shown to be effective in cases when the corpora are large. Basically, RI projects sparse term vectors into dense vectors in a low dimensional space while also roughly preserving relative distances between the vectors.

2.1.1 Random Indexing Algorithm

RI employs two steps: allocation of elemental vectors and training. Allocation of elemental vectors produces a set of nearly orthogonal vectors, one for each term in the corpus, of a particular user chosen dimension. The process of allocation is illustrated in the following pseudocode, in which V is an n dimensional elemental vector, k is a selected integer less than n, $initialize(V)$ sets all members of V to zero and $random(V, n)$ generates a random number $r \in \{0, \ldots, n-1\}$ such that $V[r]$ has not already been assigned a nonzero value.

```
allocate_elemental_vectors:
    initialize(V)
    for i = 1:k
       r = random(V,n),
       V[r] = −1;
       r = random(V,n),
       V[r] = 1;
```

Thus each vector has +1 and -1 values at exactly k randomly selected positions each. The -1 and +1 value are called seeds and the other $n - 2k$ values in each elemental vector are 0s. Underlying RI is the observation that although a d dimensional space can have only d perpendicular axes, it can contain many more nearly orthogonal axes. So repeated calls to *allocate_elemental_vectors* create a set of vectors that are nearly orthogonal to each other, hence the name "elemental" vectors. Note that k is typically very small compared to n; the more sparse the elemental vectors are, the higher the probability of these vectors being close to orthogonal to one another. However, if k is very small, there is a possibility that some elemental vectors are exactly the same. So the choice of k is key in RI and usually k is set to a number between 10 and 50.

After elemental 'document' vectors are allocated, they are trained to derive term vectors to also capture term-term connections as shown in the following pseudocode.

```
train_model(M, D):
  intialize matrix T
  for i = 1:q:
    for j = 1:p:
       T[i] += weight( M[i][j] * D[j]).
     normalize(T[i])
  return T
```

Here D is the document corpus containing p documents and $D[j]$ represents the n dimensional document vector for the j-th document. T is the $q \times n$ term vector matrix containing q term vectors. M is the frequency matrix with $M[i][j]$ containing the number of times the i^{th} term occurs in the j^{th} document. Note that the *weight* is some weighting function such as the tf-idf or log-entropy functions and normalization (in Normalize($T[i]$)) is applicable in most situations to prevent frequent terms gaining extremely large weights. Without any weight function or normalization, the training is simply an implementation of vector multiplication and addition.

RI's ability to draw meaningful connections between terms is outlined next. Each document in the corpus is assigned an elemental vector that is almost orthogonal to other documents' vectors, and the term vector for a given term is represented as a linear sum of the vectors of each document that contains the term. If i^{th} term does not co-occur with j^{th} term in any document, the representation of i^{th} term should be the linear sum of a set of vectors nearly orthogonal to all those that constitute the basis for the representation of j^{th} term. Consequently, cosine similarity between term vector of i^{th} term and term vector of j^{th} term will be very close to zero, suggesting that there is no relation between i^{th} term and j^{th} term. For highly related terms the similarity could be close to 1.

RI and its related methods such as random projection are possible due to the Johnson Lindenstrauss lemma [11]. This lemma calculates an upper bound of the error introduced when projecting points in a vector space into reduced dimension subspace. It states that a small set of points in a high-dimensional space can be embedded into a space of much lower dimension with distances between the points preserved. Compared to LSA, the computational advantages of using RI are in two ways. The sparse term vectors can be directly represented in a denser form and dimensionality reduction can be performed by simple vector addition and multiplication, rather than using computationally expensive methods like SVD.

Limitations of Random Indexing: Documents that discuss similar topics and contain synonymous terms are naturally expected to have vectors that are similar (in terms of cosine similarity) instead of being nearly orthogonal. However, the RI is incapable of assigning similar random vectors to synonyms at the first place. This drawback limits the ability of the RI to find meaningful similarities between terms that do not co-occur directly in the same document. Fortunately, a variant of RI, Reflective Random Indexing [8], overcomes this disadvantage effectively.

2.1.2 Reflective Random Indexing

One observation of the training process is that a document is represented as the weighted sum of vector representations for the terms that it contains, and vice versa. In other words, each term in the corpus can be represented as the linear sum of the vectors for the documents that contain it. Therefore, we can calculate document vectors from terms vectors in the same way. So it is possible to train the term and document vectors cyclically. Intrigued by such an idea, several iterative variants of RI which are customized to enhance its ability to draw indirect similarities are developed. These RI variants are collectively terms as Reflective Random Indexing (RRI). A particular variant is the Term-based Reflective Random Indexing (TRRI), which is illustrated in Figure 2.1.

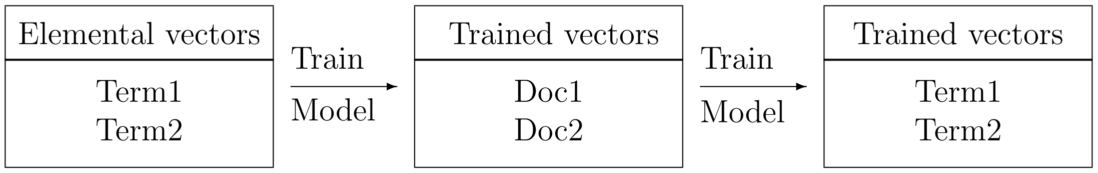

Figure 2.1: A schematic representation of the algorithm of TRRI

RRI not only retains the scalability of RI, but also enhances the similarity between documents on the same topic or that share some synonymous terms. Now for instance say document A and document B are on the same biomedical topic with common/synonymous biomedical terms. Then the document vector of document A would be very similar to document vector of document B, for they are both generated from a set of similar term vectors. If we apply one more iteration, this similarity is enhanced further, and thus RRI can draw meaningful associations between terms that do not co-occur directly in documents, called "indirect inferences".

To further explain the RRI, here is the pseudocode for TRRI and Doc based Reflective Random Indexing (DRRI):

```
pseudocode for DRRI
  $D_0$ = allocate_elmental_vectors
  $T_1$ = train_model( M, $D_0$)
  $D_2$ = train_model( M, $T_1$)
  $T_3$ = train_model( M, $D_2$)

pseudocode for TRRI
  $T_0$ = allocate_elmental_vectors
  $D_1$ = train_model( M, $T_0$)
  $T_2$ = train_model( M, $D_1$)
```

Thus, we can draw a figure that illustrates the relationship between RRI variants, show in Figure 2.2 (from Cohen's paper [8]). Note that this process can be repeated several times. However, vectors tend to converge after many iterations and so application of too many iterations may not be necessary to save resources.

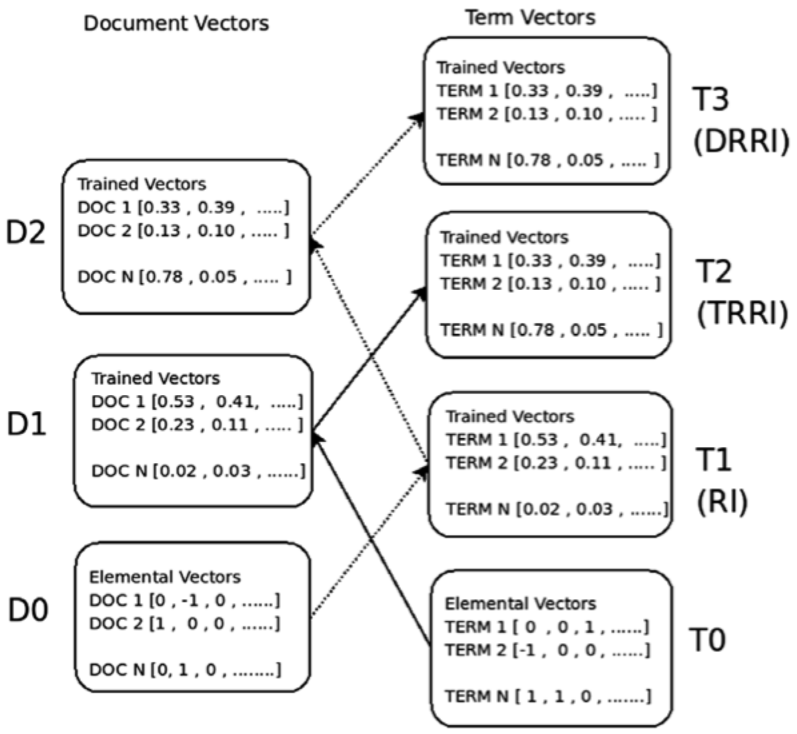
Document Vectors
Term Vectors
Trained Vectors
TERM 1 [0.33 , 0.39 , ...]
TERM 2 [0.13 , 0.10 , ...]
TERM N [0.78 , 0.05 , ...]
T3
(DRRI)
D2
Trained Vectors
DOC 1 [0.33 , 0.39 , ...]
DOC 2 [0.13 , 0.10 , ...]
DOC N [0.78 , 0.05 , ...]
Trained Vectors
TERM 1 [0.33 , 0.39 , ...]
TERM 2 [0.13 , 0.10 , ...]
TERM N [0.78 , 0.05 , ...]
T2
(TRRI)
D1
Trained Vectors
DOC 1 [0.53 , 0.41, ...]
DOC 2 [0.23 , 0.11 , ...]
DOC N [0.02 , 0.03 , ...]
Trained Vectors
TERM 1 [0.53 , 0.41, ...]
TERM 2 [0.23 , 0.11 , ...]
TERM N [0.02 , 0.03 , ...]
T1
(RI)
D0
Elemental Vectors
DOC 1 [0 , -1 , 0 , ...]
DOC 2 [1 , 0 , 0 , ...]
DOC N [0, 1 , 0 , ...]
Elemental Vectors
TERM 1 [0 , 0 , 1 , ...]
TERM 2 [-1 , 0 , 0 , ...]
TERM N [1 , 1 , 0 , ...]
T0

Figure 2.2: A schematic representation of the generation of RI variants (from [8])

2.1.3 Comparison of Different Reflective Random Indexing Approaches

Cohen et al. [8] conducted a study on the proportion of predicted direct occurrences in 10 nearest-indirect neighbors (precision at k = 10) with cyclical retraining of TRRI and DRRI on a corpus of MEDLINE abstracts. From the experiment results, they found out that the second iteration (TRRI2) of TRRI predicts a greater number of direct co-occurrences than any iteration of DRRI while TRRI shows the greatest sensitivity to cosine similarity and lower term-frequency of future connections. However, with repeated iterations of RRI, all vectors converge on a single vector, and the ability to discriminate between them is lost.

In our study, based on the fact that the number of our documents is much larger than the number of terms, we decided to use TRRI. If DRRI was used, the huge number of documents will result in high possibility of many documents ending up with the same elemental document vector and many similar term vectors even though those terms are not related. Considering time and space consumption in TRRI2 and the fact that TRRI2 doesn't improve much compared to TRRI (from [8]), we choose TRRI in our experiments.

2.1.4 Selection of dimension

The selection of an appropriate vector dimension is critical in RI, since a very large dimension leads to huge computational storage and time and too small a dimension will result in a set of similar elemental vectors. Thus, we test different dimensions based on mean average precision (MAP), shown in the following figure:

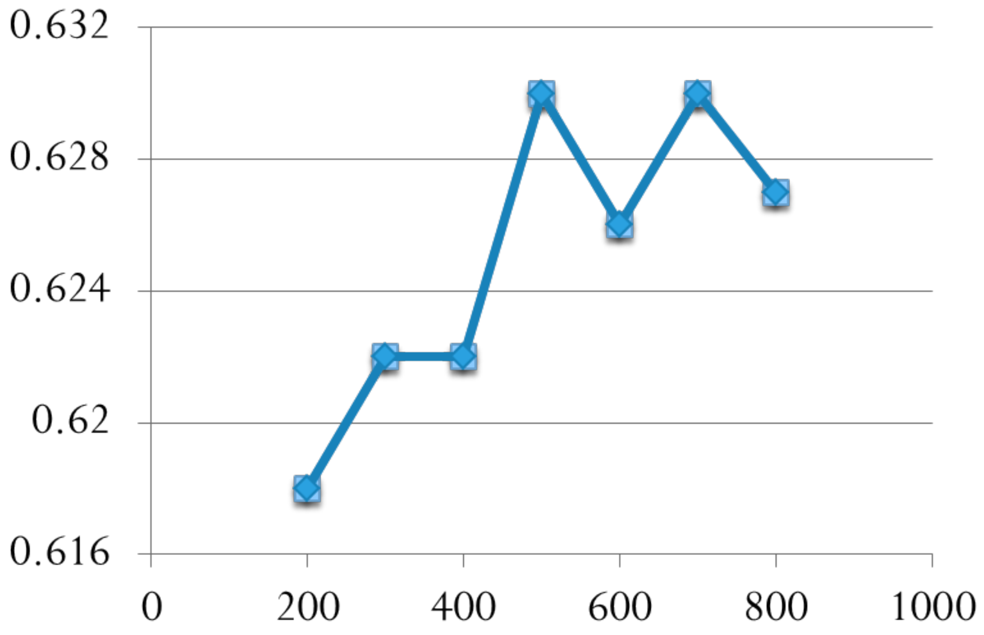

Figure 2.3: MAP values based on different TRRI dimensions

An observation of this figure is that from dimension 200 to 500, the MAP raises on an upward slope and after dimension 500, MAP increases and decreases within a upper bound and lower bound, suggesting that after dimension 500, the increasing of dimension doesn't lead to gains in MAP. Based on this observation, we select 500 as the TRRI dimension.

2.2 Learning to Rank

Learning-to-rank is a machine learning technique that is widely utilized in information retrieval to rerank search results based on a learned ranking function using a training dataset of relevant results for specific queries. Recently, a number of supervised feature-based learning-to-rank algorithms were proposed such as RankNet [5], AdaRank [22], ListNet [7]. Coordinate Ascent [16] and Random Forest [4] are among the best of them with an advantage of scalability and flexibility.

2.2.1 Learning to Rank in Traditional Information Retrieval

In traditional Information Retrieval, an search engine is used to return a ranked list of documents based on an input query along with their relevance score. Traditionally, the relevance score is computed relying on the query term and text of documents by some established algorithms such as BM25 [19]. The learning to rank method is different from traditional methods in two aspects: learning to rank will take query specific features which evaluate the similarity between query term and document as input to a supervised model and then computes relevance score by combining those features. Two functions are important in learning to rank: the ranking function and the loss function. The ranking function is learned through training data, and applied to the test data. Loss function is defined to evaluate the differences in the positions of correct rankings of documents and rankings generated by ranking function. Thus, the training process in learning to rank is to learn a ranking function that minimizes the loss function. Given a new query and a set of documents, the ranking function will impose a ranking based on the query specific document features.

2.2.2 Learning to Rank in Our Approach

In our approach, the input query instance is document instead of query term; the items to be ranked are labels instead of documents. Thus, we use learning to rank to rank a set of labels in response to a new document based on several features. In our efforts, we experimented with all the learning to rank algorithms made available in the RankLib library (`http://sourceforge.net/p/lemur/wiki/RankLib/`). Among these co-ordinate ascent approach, a linear featured model, seemed effective for predicting MeSH headings and random forest based approach worked best for extracting diagnosis codes.

2.3 Evaluation Measures

Multi label classification measures are generally evaluated using exampled based and micro/macro averages of precision, recall, and F-score. In this section we define these measures to be used in our experiments. The definition of example-based precision, recall and F-score are listed in the following equations, where the Y_i and Z_i are set of correct labels and predicted labels extracted by our methods, respectively, for the i-th input instance and m is the number of instances in test data L:

$$P_{ex} = \frac{1}{m}\sum_{1}^{m}\frac{|Y_i \cap Z_i|}{|Z_i|},\ R_{ex} = \frac{1}{m}\sum_{1}^{m}\frac{|Y_i \cap Z_i|}{|Y_i|}$$

$$and\ F_{ex} = \frac{1}{m}\sum_{1}^{m}\frac{2|Y_i \cap Z_i|}{|Z_i| + |Y_i|}$$

For each label T_j, we define the label-based precision ($P(T_j)$), recall ($R(T_j)$), and F-score ($F(T_J)$) as follows where the TP_j, FP_j, and FN_j are true positives, false positives, and false negatives, respectively, for label T_j.

$$P(T_j) = \frac{TP_j}{TP_j + FP_j},\ R(T_j) = \frac{TP_j}{TP_j + FN_j},$$

$$and\ F(T_j) = \frac{2P(T_j)R(T_j)}{P(T_j) + R(T_j)},$$

The label-based macro precision, recall and F-score are defined as the average of label-based precision, recall and F-score over all labels in set T:

$$P_M = \frac{1}{|T|}\sum_{j=1}^{|T|}P(T_j)\ and\ R_M = \frac{1}{|T|}\sum_{j=1}^{|T|}R(T_j)$$

$$F_M = \frac{1}{|T|}\sum_{j=1}^{|T|}F(T_j).$$

The label-based micro precision, recall and F-score are defined as

$$P_\mu = \frac{\sum_{j=1}^{|T|}TP_j}{\sum_{j=1}^{|T|}(TP_j + FP_j)},\ R_\mu = \frac{\sum_{j=1}^{|T|}TP_j}{\sum_{j=1}^{|T|}(TP_j + FN_j)},$$

$$and\ F_\mu = \frac{2P_\mu \cdot R_\mu}{P_\mu + R_\mu}$$

The micro measures emphasize on labels that are more frequent while macro measures tend to distribute the weight of all labels equally.

For the medical subject extraction task, we also use the mean average precision (MAP) measure. Let us additionally assume that labels in Y_i are ranked. Let $c(N, L_i, Y_i)$ be the number of true positives in the top N labels in Y_i for instance L_i, Y_i^r be the r-th ranked label in Y_i, and $I(Y_i^r)$ be the Boolean function returning 1 if Y_i^r is a correct label, and otherwise 0. We define the average precision for L_i as

$$AP(L_i, N) = \frac{1}{|Z_i|}\sum_{r=1}^{m}I(Y_i^r)\cdot\frac{c(r, L_i, Y_i)}{r}$$

and MAP as

$$MAP(L, N) = \frac{1}{m}\sum_{i=1}^{m}AP(L_i, N).$$

Chapter 3 Novel Learning-to-Rank Features Based on Random Indexing

In addition to the traditional learning-to-rank features such as the nearest neighbor scoring feature and the binary classifier scoring feature, we also incorporate RI based scores as features into the ranking function. While conventional features evaluate the similarity between candidate labels and text of documents, the RI feature incorporate the implicit connections between labels. However, we first briefly outline the conventional features used.

3.1 Normal Learning-to-Rank features

From Section 1.1, we recall that the k-NN and the binary classifier based approaches are two types of general approaches, for solving multi label classification problems, each of which results in a set of candidate labels which can be reranked used learning to rank. These approaches also naturally lead to two different features that are typically used in the ranking function.

3.1.1 Nearest Neighbor Feature

Each of the nearest neighbors in the k-NN approach has a corresponding similarity score with the new input instance to be assigned labels. In most cases, the similarity scores between documents are computed based on the weighted score of the words they have in common. The weighting is normally carried out by the popular `tf-idf` (term frequency and the inverse of document frequency) transformation. The nearest neighbor feature of a candidate label is the sum of the similarity scores of the input instance with all nearest neighbors in the training data that have the candidate label assigned to them.

3.1.2 Binary Classifier Feature

Given a new input instance, the binary classifier outputs a classifier score for each of the labels. For classifiers that use models like logistic regression, the scores are between 0 and 1 with the score close to 1 indicating that the corresponding label should be assigned. However, when dealing with labels with very low frequency, a strict 0.5 threshold may not be ideal. Hence the top several hundred labels are typically selected as candidates for scoring without actually looking into the individual scores. These candidates labels are typically reranked to select the top few as the final candidates. One of the learning to rank features is the actual classifier score output by the models.

3.2 Random Indexing Feature

As discussed in the previous section, conventional features in learning-to-rank are designed to measure the association between labels and documents. In our research, in addition to these features, we also include novel features that involve label co-occurrence and random index scores. The intuition behind this is that a high confidence label to be assigned to a document will increase the chance of other labels being associated with document if they are related to the original labels based on distributional semantics (from historical data).

Co-occurrence feature [12] is computed based upon co-occurrence frequency of candidate labels from historical data, basically using an estimate of the conditional probability of the occurrence of a label given another label has occurred. The co-occurrence feature evaluates the relatedness of labels based on explicit co-occurrence. However, it is not able to capture latent or implicit association between labels that might not have co-occurred frequently in historical data but are strongly associated from a distributional semantics perspective. We employ an RI feature to discover these implicit links between labels.

Random Indexing feature is computed using Term-Based Reflective Random Indexing with the 500 dimension given that TRRI show the best performance among variants of RRI [8]. The size of dimension is selected based on a series of experiments with consideration of both accuracy and time complexity. The actual feature is based on first building a random index with all label sets from historical data. Each of the label sets is considered a document with each constituent label being a term in the document. With the TRRI approach, first elemental term vectors are assigned and document vectors are computed from them which are again used to estimated the term vectors. At the end of this index building process, we will have term vectors for all labels in the vocabulary and the index can rank all labels in the vocabulary given an input label based on cosine similarity. To define the RI feature, we will need to define 'context' terms for an input instance. For any new input instance to be assigned labels, we can run a named entity recognizer that can automatically identify some of the labels that are directly mentioned in the input text. However, a mere mention does not make a label a true label and many labels assigned are not mentioned in the input text. However, labels identified through named entity recognition (NER) are definitely related to the correct labels since they are mentioned in the input instance. For a new input instance, we first identify the context labels and then for a candidate label the RI score is sum of RI relatedness scores between the candidate and each of the context labels.

Chapter 4 Prediction of MeSH Terms

This chapter mainly focuses on our experiments and results in predicting medical subject headings using the k-NN approach with learning to rank for ranking candidate terms. We show that incorporating random indexing features along with co-occurrence frequency based features improves over the best results on two public datasets.

4.1 Existing Approaches for Predicting MeSH terms

As discussed in the first chapter, the indexing of a growing number of biomedical citations by human indexers is not effective and thus a number of approaches are proposed to aid indexers to speed up the indexing process and increase the indexing correctness. Existing approaches use a combination of simple named entity recognition with the traditional binary classifier approach or the k-NN approach [13, 17, 20, 24, 23, 9, 2, 3, 21]. These methods do not use the same evaluation metrics and are not tested on a common test set. Therefore, it is not easy to compare them directly with our approach. For our experiments we use two public datasets created by NLM researchers [10] and also use the nearest neighbors provided by their effort.

4.2 Outline of Our Approach

For each input citation to which we need to assigned MeSH terms, we obtain 50 nearest neighbors obtained by Huang et al. [10]. All MeSH terms assigned to these neighbors form the set of candidate terms for the input citation. We also retrieved context MeSH terms from the title and abstract of the given document by running NLM's NER tool MetaMap on the title/abstract text and restricted the obtained concepts to those from MeSH. Each candidate MeSH term is assigned a score by our learning to rank algorithm based on multiple features. The top N candidate terms are considered relevant to given article. N is set to be 25 in our experiments following prior attempts by others [10]. We use micro precision, recall, and F-score and MAP score to evaluate our methods.

4.3 Features for Learing-to-Rank

Here we provide specific details of the features used for learning to rank for the MeSH term prediction task.

4.3.1 Neighborhood features

We have already outlined (in Chapter 3) the neighborhood features to be used in our work. For candidate term the feature is essentially the sum of similarity scores of the input citations with all neighbors tagged with the candidate term. However, instead of just considering all $k = 50$ neighbors, we can consider the feature for different values of k. Our experiments with various k are shown in Table 4.1.

Table 4.1: Comparison of combinations of different k values

Combination	MAP	Combination	MAP
k=10	0.616	k=50	0.617
k=10,20	0.625	k=50,10	0.630
k=10,30	0.626	k=50,20	0.625
k=10,40	0.626	k=50,30	0.620
k=10,50	0.630	k=50,40	0.620

We experimented with neighborhood score sum features for different k values from $k = 10$ to 50 with increments of 10. Intuitively, using a combination of neighborhood features for several k helps the algorithm learn the optimal importance (feature weights) it should assign to neighbors at different distances from the testing instance; choosing a fixed k handicaps the learning process in this sense. The best combination used features with $k = 10$ and $k = 50$. We believe these different neighborhoods provide complementary information with score in the top 10 neighbors indicating the high relevance of a candidate term and the score from top 50 neighbors providing the plausibility of candidates that may be included in the final terms if they are favored by other feature types.

4.3.2 Random Indexing Feature

Before computing RI feature score for each candidate term, we need to establish a set of term vectors for MeSH. First, we built a training set comprised over 18 million citations that have already been assigned with MeSH terms and are published after 1990, available in MEDLINE except for the 1200 citations that are in the test datasets. In other words, there is no overlap between the test and training sets. So as discussed in Section 3.2, a document is completed consists of MeSH terms assigned to a citation. We used integers to represent the MeSH terms to easily tokenize terms given their string descriptions often have more than one word. Second, to generate the vector representations for documents and MeSH terms in the training set, elemental vectors are assigned to each MeSH term. The dimension of the vectors is set to 500 and seed length is set to 10 to keep the sparse vectors near orthogonal. The document vectors are then generated as the linear sum of the random term vectors (MeSH term vectors). To stop the magnitudes in document vectors from increasing heavily, normalization is done in every step; this doesn't change the distance of two document vectors because the distance is measured as cosine similarity (the scalar product of normalized document vectors) in RI. After this step, document vectors are used to generate new term vectors. New term vectors are produced by sum of the document vector of the documents in which the term occurs. Finally, after two iterations, the MeSH term vectors are represented in 500 dimensional vector spaces and written to a file.

After previous steps, each MeSH term has a representation vector that shows the relation of this term to other ones. To obtain the RI feature of a specific candidate term, the cosine similarity of this candidate term vector and all context term vectors are calculated and added up. Since the context terms are extracted from document by name entity recognition, this feature also measures the similarity between the candidate term and the given document and will be the RRI based feature score of the candidate term.

4.3.3 Context Term, Semantic Predication, and Co-Occurrence Features

If a candidate term is also a context term, context term feature of that candidate term will be set to 1, otherwise 0. Another Boolean feature is based on semantic predications or subject-predicate-object triples extracted from the abstract/title text. If the candidate term participates as either the subject or object of a semantic predication, the semantic predication feature will be labeled as 1, otherwise 0. The co-occurrence feature is similar to the RI feature except instead of the RI score we aggregate the probability estimates of the candidate term given the context terms (see Kavuluru and He [12] for further details).

4.4 Results and Discussion

For learning-to-rank the training set is a set of 200 citations, named Small200, where their corresponding MeSH terms were recently assigned (from 2002 to 2009). There are two different test datasets, the NLM2007 and L1000 datasets with 200 and 1000 citations each. All these datasets are obtained from Huang et al. [10]. All the feature values were normalized to [0, 1] by dividing each value with the corresponding maximum value for that feature.

4.4.1 Comparison to Other Methods

We compared our results with those from NLM's MTI system and also with results from Huang et al [10] method. The comparison of the best results by Huang et al. with our results is shown in Table 4.2. In the table, our method outperforms Huang et al.'s method over all four measures for both datasets. Note that the relative difference is steady across the two datasets. There is a MAP score improvement of 1.8% for the NLM dataset and 1.5% for the L1000 dataset. The last row shows the best performance with only the neighborhood features, including both the top 10 and top 50 neighbors cutoff. The difference in performance measures when using all features (row 2) and neighborhood features is nearly twice that of the corresponding difference between Huang et al.'s and our method with all features, demonstrating that even though neighbor features are critical in learning to rank, the best performances is obtained with all the features.

Table 4.2: Comparison of micro measures with $N = 25$

	NLM2007 dataset				L1000 dataset			
	R_μ	P_μ	F_μ	MAP	R_μ	P_μ	F_μ	MAP
Huang et al	0.712	0.390	0.504	0.626	0.714	0.347	0.467	0.615
Our method	0.727	0.398	0.514	0.644	0.730	0.355	0.478	0.630
f^N_{10}, f^N_{50} only	0.696	0.382	0.493	0.620	0.698	0.339	0.456	0.603

4.4.2 Feature Study

To investigate the impact of different features, we performed a feature ablation study. Among six features in our experiment, we removed one feature each time from the full feature set and trained the model on Small200 and tested the performance on both datasets. The results are presented in details in Table 4.3, where a feature with a $\sim$ symbol next to it implies that it has been removed from the learning-to-rank framework.

Table 4.3: Feature ablation analysis of micro measures with $N = 25$

	NLM2007 dataset				L1000 dataset			
	R_μ	P_μ	F_μ	MAP	R_μ	P_μ	F_μ	MAP
all features	0.727	0.398	0.514	0.644	0.730	0.355	0.478	0.630
$\sim f^N_{10}$, $\sim f^N_{50}$	0.574	0.314	0.406	0.382	0.578	0.281	0.378	0.376
$\sim f^F$	0.719	0.393	0.508	0.637	0.719	0.349	0.470	0.621
$\sim f^R$	0.720	0.394	0.509	0.642	0.726	0.353	0.475	0.625
$\sim f^F \sim f^R$	0.697	0.382	0.493	0.621	0.697	0.339	0.456	0.603
$\sim f^A$	0.720	0.394	0.509	0.644	0.727	0.353	0.476	0.625
$\sim f^P$	0.720	0.394	0.509	0.644	0.730	0.354	0.477	0.627

In the table, the more significant performance drop occurs when the neighborhood features were removed, suggesting the two neighborhood features are the most significant contributors given their direct association with nearest neighbors from a large dataset. Note that neighborhood features alone do not achieve the best results if other features are removed. We note that the RRI feature and co-occurrence feature are important as observed by the performance drops when either is ablated. Also, the performance drop when either feature is dropped is smaller compared to the drop when both of them are removed showing that they both incorporate complementary information. Dropping both the co-occurrence and RI features leads to a loss of over 3% in recall, 2% in F-score, and nearly 3% in MAP for the bigger L1000 dataset.

The only common feature between our approach and Huang et al. method is the k-nearest neighborhood based feature and we utilize different k values compared to Huang et al.'s method. Beside neighborhood features, their method adds sophisticated features that use an additional training dataset of nearly 14,000 citations and the corresponding MeSH term sets to obtain probability estimates $P(t|I)$ of the probability of a MeSH term t given the title and abstract text of the instance I. These estimates rely on the distributions of individual tokens of the preferred name of the

MeSH term and those present in abstract and title text of the citation. In contrast, we use implicit terms associations that lead to higher performance instead of relying on the associations between the words in the citation and candidate terms.

4.4.3 Manual Qualitative Error Analysis

The moderate recall in these experiments suggests that our model can be further optimized in theory. Under such consideration, we performed a manual qualitative errors analysis of false negatives (FNs) after applying our methods on the small NLM2007 dataset. Only 15% of the citations have the 100% recall; in the rest, several false negatives occur when a more specific term or a more generic term of the correct term is included in the top 25 terms. For example, when “Social Behavior” was the correct term for a citation, but we had the generic term “Behavior” as one of the predicted terms. When “Doppler Ultrasonography” was the correct term, our method extracted a more specific term “Duplex Doppler Ultrasonography”. Since MeSH is inherently hierarchical in nature, using such information [6] in our framework might help improve the accuracy. Also, terms that are not specific but are used to characterize the study discussed in the paper are often missed by our methods. Examples of such terms are “sex factors”, “time factors”, and “follow-up studies”. A manual examination of a citation where we missed “sex factors” shows no indication of stratification or analysis based on the sex of the patient. This is probably discussed in the full text but “sex factors” was ranked 98th for this particular citation. To address this problem, we may build a binary classifier for each term which falls in the false positive category as a complement to the learning to rank approach.

Chapter 5 Prediction of Diagnosis Codes

Diagnosis codes are regularly extracted from EMRs by trained coders for reimbursement and data quality purposes. In this chapter, we present the application of our methods to automatically predict diagnosis codes based on textual narratives in patient EMRs. Please see Chapter 1 for an introduction to the ICD-9-CM terminology which forms the label set for this problem. Unlike in the previous chapter where k-NN was used to identify the candidates, here we simply use the top 200 codes as candidates when all codes are ranked based on binary classifier scores when the corresponding logistic regression models are run for all possible codes.

5.1 EMR Dataset

Our datasets is the set of all EMRs of the UKY medical center in-patient visits with discharge dates in the 2011-2012 two year period. The dataset is properly approved for retrospective research by the UKY IRB. The dataset has a total of 71,461 EMRs containing a total of 921,000 physician authored documents and 7,485 unique ICD-9 codes. The average size of words per EMR is 5303 and the average number of documents per EMR is 13. The average number and median number of codes per EMR are 9.76 and 8, respectively. Given that a huge number of the ICD-9 codes occur less than 10 times, we only consider predicting codes at the fourth digit level; for instance, we mapped 800.21 to 800.2 and kept these codes that are already codes at third or fourth digit level. After this process, we obtain a set of ICD-9 codes including 4,723 codes. Even when truncated to 4 digits, there were still many codes that had too few examples to apply supervised methods. Thus, we restricted the set of ICD-9 to 1231 codes, each of which occur at least 50 EMRs. This subset of ICD-9 codes contributed to the 76% of the total diagnoses in the dataset.

5.2 Code Prediction Framework and Features for Learning to Rank

Our method of extracting ICD-9 codes from EMR records has the following three steps

1. We transform the multi-label classification problem into multiple binary classification problems. Assuming that all codes are independent, for each code we identify the positive examples and the rest of the EMRs are set as negative examples. We build multiply binary classifiers one for each of 1231 codes. The main features of the binary classifiers are unigram and bigram counts with the popular *tf-idf* transformation. The used logistic regression (LR) as the main classifier from the scikit-learn Python library [18].

2. We use the learning-to-rank technique and rank the top 200 ICD-9 codes based on the binary classifier output scores and other features similar to those used for

MeSH term prediction. After experimenting with various algorithms we selected the random forest based learning to rank algorithm from RankLib library used for our experiments on MeSH term prediction.

3. Finally, we use linear regression to predict an appropriate number of labels for each EMR instead of a choosing a constant number. We select top few labels based on this predicted number from the reranked list of candidates. The features used for this model are exactly those used to build the binary classifiers.

The main feature for a candidate term is the [0, 1] output score of the corresponding binary classifier. UKY medical center has a total of 2 million visits since 2011 and each visit has a set of diagnosis codes attached to that EMR. Thus, we can capture the explicit connection between codes by co-occurrence and implicit connections using RI. Since we assume all codes are independent in transforming the multi label diagnosis extraction problem to multiple single label problems, it is necessary and essential to reconsider the connections between labels in this step. We derive the co-occurrence feature based and RI feature from this output code set database exactly like we did in the MeSH prediction task. The context code set features based on codes directly extracted using NER is also used in this task.

5.3 Experiments, Results, and Discussion

From the full dataset of over 71,000 EMRs, we randomly selected 2000 EMRs for validation and 3000 for testing, and all other records were used for training. The linear regression function, the learning-to-rank ranking function are all obtained using the validation dataset. Finally, we test our model on the test set with group truth code sets coming from the coder assigned codes. The results of our experiments are displayed in Table 5.1.

Table 5.1: UKLarge test set scores (with binary relevance as the transformation)

Calibrator	Learner	Example-Based			Micro			Macro		
		P	R	F	P	R	F	P	R	F
RCut=8	LR	0.434	0.489	0.399	0.434	0.387	0.410	0.824	0.181	0.176
	LR+L2R	0.443	0.501	0.408	0.443	0.395	0.417	0.799	0.190	0.189
Linear Regression	LR	0.482	0.461	0.442	0.513	0.421	0.463	0.859	0.176	0.180
	LR+L2R	0.491	0.469	0.450	0.524	0.430	0.472	0.844	0.184	0.188
Correct Count	LR	0.484	0.481	0.481	0.482	0.482	0.482	0.835	0.197	0.198
	LR+L2R	0.486	0.486	0.486	0.495	0.495	0.495	0.821	0.211	0.212

The first two rows of the above table show the evaluation scores if the top 8 codes after reranking were selected to be assigned to the EMRs. The middle two rows are obtained when the regression function to predicate the correct number of labels was applied. Over the first four rows, our best performance is in row 4 with the best micro and macro F-score. Given the linear regression model may not perfectly predict the correct number of codes for an EMR, we obtained scores assuming a perfect calibrator that is, assuming we know the correct number of labels for each EMR. These scores are shown in final two rows of the table. Regardless of the calibrator, we see that using learning to rank always improves all metrics including precision, recall and F-score. The usage of learning to rank lead to a 1% improvement (or a 3.4% relative improvement) in the micro F-score although having a better calibrator could add an additional 2% improvement. In all experiments, example-based and micro F-score are close to each other, but the macro F-score is substantially lower owing to the very low code level recall for several codes with very few training examples.

Chapter 6 Conclusion

In the thesis, we focused on the multi-label classification problem and its two applications in biomedical informatics: MeSH term assignment and diagnosis code extraction. For the MeSH assignment, we used the k-NN approach to identify candidates and used a learning to rank approach to rerank candidates based on a set of features which include RI scores of a candidate term with the context terms derived from the abstract. We obtained new state-of-the-art results on two public datasets using our approaches. The influence of different features is analyzed and false negatives errors are investigated. Our future work will focus on improving the recall. In theory, we can get a nearly 90% recall from the neighbors but our recall is only 73%. For ICD-9 code extraction, we built multiple binary classifiers one for each code to identify the top 200 candidates based on classifier score output. These candidates were ranked according to a learning to rank approach using features similar to those used in the MeSH task. Even in this task we observed that the performance improves when using the RI and co-occurrence features over the baseline. In our future work, we will employ a combination of k-NN and binary classifier approach and use the corresponding features in the learning to rank framework.

Bibliography

[1] AMA. International classification of diseases, 9th revision, clinical modification: physician ICD-9-CM. *American Medical Association*, 1, 2004. 2

[2] A. Aronson. Effective mapping of biomedical text to the UMLS Metathesaurus: the metamap program. *Proceedings of the AMIA Symposium*, pages 17–21, 2001. 13

[3] A. Aronson, O. Bodenreider, H. Chang, S. Humphrey, J. Mork, S. Nelson, T. Rindflesch, and W. Wilbur. The NLM indexing initiative. *Proceedings of the AMIA Symposium*, pages 17–21, 2000. 13

[4] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. 4, 9

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *In Proceeding of ICML*, pages 89–96, 2005. 9

[6] L. Cagliero and P. Garza. Improving classification models with taxonomy information. *Data and Knowledge Engineering*, 86:85–101, 2013. 17

[7] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to Rank: From Pairwise Approach to Listwise Approach. *ICML*, 2007. 9

[8] T. Cohen, R. Schvaneveldt, and D. Widdows. Reflective random indexing and indirect inference: a scalable method for discovery of implicit connections. *Journal of Biomedical Informatics*, 43(2):240–256, 2010. vi, 4, 6, 7, 8, 12

[9] G. Cooper and R. Miller. An experiment comparing lexical and statistical methods for extracting MeSH terms from clinical free text. *Journal of the American Medical Informatics Association*, 5(1):62–75, 1998. 13

[10] M. Huang, A. Neveol, and Z. Lu. Recommending mesh terms for annotating biomedical articles. *Journal of the American Medical Informatics Association*, 18(5):660–667, 2011. 13, 15

[11] W. B. Johnson and J. Lindenstrauss. Extension of lipschitz mapping to hilbert space. *Contemp Math*, 26:189–206, 1984. 5

[12] R. Kavuluru and Z. He. Unsupervised Medical Subject Heading Assignment Using Output Label Co-occurrence Statistics and Semantic Predications. *Springer*, 7934:176–188, 2013. 12, 15

[13] W. Kim, A. Aronson, and W. Wilbur. Automatic MeSH term assignment and quality assessment. *Proceedings of the AMIA Symposium*, pages 319–23, 2001. 13

[14] T. K. Landauer and S. T. Dumais. A solution to Platos problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997. 4

[15] T. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 17:285p, 2011. 4

[16] D. Metzler and W. Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10:257–274, 2007. 9

[17] A. Nvol, J. Mork, and A. Aronson. Automatic indexing of specialized documents:using generic vs. domain-specific document representations. *Bioinformatics*, pages 183–90, 2007. 13

[18] F. Pedregosa, G. Varoquaux, and A. Gramfort. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 18

[19] S. Robertson and H. Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. 9

[20] S. Sohn, W. Kim, D. Comeau, and W. Wilbur. Optimal training sets for bayesian prediction of MeSH assignment. *Journal of the American Medical Informatics Association*, 15(4):546–53, 2008. 13

[21] V. Vasuki and T. Cohen. Reflective random indexing for semi-automatic indexing of the biomedical literature. *Journal of Biomedical Informatics*, 43(5):694–700, 2010. 13

[22] J. Xu and H. Li. AdaRank: a boosting algorithm for information retrieval. *In Proceeding of SIGIR*, pages 391–398, 2007. 9

[23] Y. Yang and C. Chute. A linear least square fit mapping method for information retrieval from natural language texts. *14th international conference on computational linguistics. Nantes*, p:446–453, 1992. 13

[24] Y. Yang and C. Chute. An application of expert network to clinical classification and MEDLINE indexing. *Proceedings of the eighteenth annual symposium on computer applications in medical care*, pages 157–161, 1994. 13

Vita

- Name:

  Yuan Lu

- Education:

  Bachelor of Science in Chemical Physics, Sept. 2006 to July 2010
  University of Science and Technology of China Hefei, China

- Publication:

  Ramakanth Kavuluru and Yuan Lu, Leveraging Output Term Co-occurrence Frequencies and Latent Associations in Predicting Medical Subject Headings, Data and Knowledge Engineering 94 (B): 189-201 (2014)