Distributed Representations for Automating MeSH Indexing

Scott A Longwell
Department of Bioengineering
Stanford University
Palo Alto, CA 94306
longwell@stanford.edu

Abstract

Manual MeSH indexing of the millions of journal articles cataloged in PubMed each year has become a daunting and expensive challenge for the National Library of Medicine. While the prospect of automated indexing is tempting, the requisite task of multilabel hierarchical classification is a difficult one. This article explores the possibility of generating distributed vector representations of both PubMed abstracts (AbsVecs) and MeSH terms (TermVecs) that can be used to quickly assign relevant terms to new articles in an unsupervised fashion.

1 Introduction

Since 1960, the National Library of Medicine (NLM) has manually indexed journal articles in the life-sciences with Medical Subject Headings (MeSH). Updated yearly, MeSH is a thesaurus of 27,883 descriptors organized into a directed acyclic graph (DAG) with 16 top-level categories (Figure 1). These descriptors help researchers navigate some 20 million articles cataloged in PubMed.

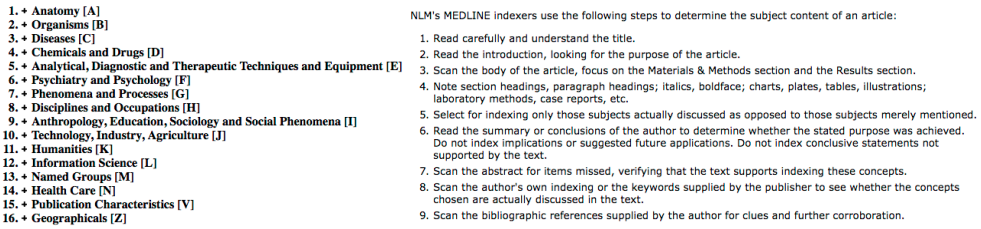

Figure 1: High-level MeSH categories; NLM guidelines for MeSH indexing. Images from

Manual MeSH indexing is held as the gold-standard, but it requires a team of expert indexers who are deeply familiar with the biomedical domain and MeSH terminology. As the body of biomedical literature grows exponentially, manual curation efforts have become increasingly expensive. Moreover, since MeSH is updated yearly, indexers do not re-index articles from prior years (although resources are available to link current MeSH terms with previous ones). Human curation also leaves ample room for subjectivity and inconsistency between indexers.

To encourage researchers to develop algorithms which aid and replace human MeSH

curation, the BioASQ initiative created a yearly challenge [1]. In this challenge, BioASQ provides millions of MeSH-labeled PubMed abstracts for competitors to train models on. Competitors can then use their models to predict MeSH labels for unlabeled abstracts, which BioASQ periodically releases in batches before they are manually indexed by the NLM.

Since MeSH labels are organized hierarchically, the challenge task is a variant of multilabel document classification known as hierarchical classification. A PubMed article may be indexed not just by the leaves of the MeSH DAG, but also most internal nodes. This hierarchical wrinkle is both a blessing and a curse: while it collapses the some 28,000 possible labels into a more manageable space, it complicates labeling strategies. Traditional metrics for flat multilabel classification, like accuracy, are binary "hit-miss" measures which fail to capture that closely related nodes share a greater similarity than distantly related nodes.

My approach to this task can generally be outlined as follows:

1) Generate distributed representations of abstracts (AbsVecs).

2) For each MeSH term, merge related AbsVecs to generate a composite representation (TermVecs).

3) For unlabeled examples, generate test AbsVecs (i.e. query expansion); for each test AbsVec, rank TermVecs by their distance via some metric (e.g. cosine similarity), and threshold/thin the ranked list to remove redundant nodes (i.e. nodes on the same path). Such an approach should minimize a cost that takes the unique challenges of hierarchical classification into account.

2 Background

2.1 Distributed representations of words and sentences

Word-to-vec (WV) is a neural-network like architecture with a single hidden layer created by Mikolov and coworkers to generate distributed representations of words [2]. It has two flavors. The skip-gram variant (WV-SG) takes a target word as input and attempts to predict its surrounding window of words (i.e. context). The continuous-bag-of-words (CBOW) model does the opposite: it tries to predict the target word from the target word's context (figure 2). Both models have two weight matrices (one to connect the input layer to the hidden layer and one to connect the hidden layer to the output layer). Assuming an orientation in which the rows of a weight matrix correspond to words in the vocabulary and columns correspond to elements of the hidden layer, each row of either weight matrix is a vector corresponding word in the vocabulary which, through several rounds of forward- and back-propagation, comes to capture the semantic meaning of the word. The resulting word vectors (WordVecs) can be averaged to crudely represent longer strings of text, and are also useful inputs for a variety of deep learning networks like RNNs, RNTNs, and LSTMs.

More recently, Mikolov and coworkers built upon WV to generate more precise vector representations of longer strings of text; this model is commonly known as paragraph-to-vec (PV) [3]. Like WV, PV comes in two variants: (PV-DM), which is analogous in structure to WV-CBOW, and PV-CBOW. These approaches are similar to their WV counterparts, except that a paragraph matrix is included. Assuming a similar orientation, rows of this matrix correspond to vector representations of documents. As the model is trained through rounds of forward- and back-propagation, the paragraph vectors come to represent how the broader document context modifies the meaning of the word vectors that comprise it. In essence, these paragraph vectors act like a document-wide "memory".

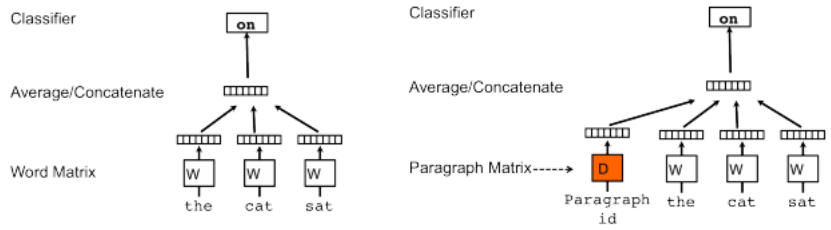

Figure 2: WV-CBOW model; PV-DM model. Images taken from Mikolov, et al. [3]

2.2 Hierarchical clustering scoring

As mentioned in the introduction, there are unique challenges to hierarchical clustering relative to traditional clustering. For example, given a correct label of "cancer" and a predicted label of "lung-cancer", a binary "hit-miss" metric like accuracy would give no credit to the prediction. However, a more appropriate metric would take into account that "lung-cancer" is closely related to "cancer" and view the prediction more favorably. These challenges are further discussed in detail by [4], and summarized in figure 3. Two promising metrics are MGIA and FLCA.

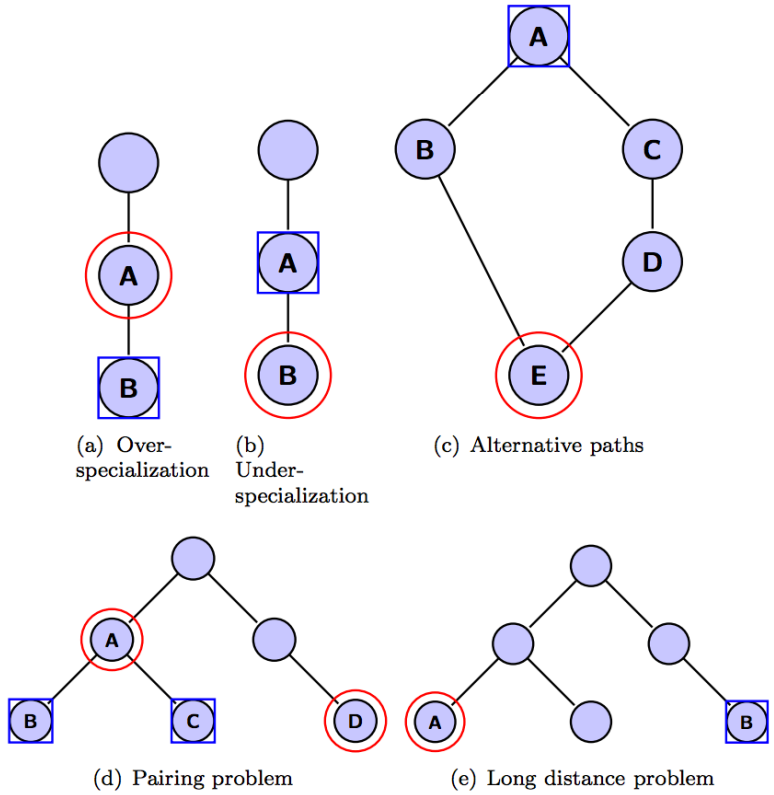

Figure 3: Five types of unique scoring challenges in hierarchical classifcation. While a flat classification metric would score all of these misses equally, more advanced metrics such as MGIA and FLCA would score, for example, (a) and (b) as better classifications than (e). Image from Kosmopoluos, et al. [4]

2.3 Prior work

This is the fourth iteration of the BioASQ MeSH indexing task. There have been several successful entries [1], many of which outperform the NLM's version of an automated indexer, the Medical Terminology Indexer (MTI). Performance results on the most recent test batch (batch 3, week 3) are shown in figure 4. The MeSHLabeler submissions, which take advantage of article metadata, are frequently top performers [5]. See http://participants-area.bioasq.org/results/4a/ for more.

System	Acc.	LCA-F
auth1	0.4184	0.4694
auth2	0.4289	0.4690
BioASQ Filtering	0.4075	0.4634
CSX-1	0.2702	0.2964
d33p	0.1270	0.1907
Default MTI	0.4201	0.4707
iria-1	0.2567	0.3557
LABDA baseline	0.3249	0.3959
LABDA ElasticSearch	0.1819	0.3158
LargeElasticLABDA	0.1812	0.3150
MeSHLabeler	0.4747	0.5113
MeSHLabeler-2	0.4735	0.5024
MeSHLabeler-3	0.4621	0.4989
MTI First Line Index	0.4088	0.4636
UCSDLogReg	0.3420	0.3781

Figure 4: BioASQ Task 4a, Test batch 3, week 3 results. LCA-F is a hierarchical measure of performance. Image from the BioASQ participants area webpage.

3 Approach and experiment

3.1 Inputs

3.1.1 MeSH hierarchy

The 2016 version of MeSH has some 27.9 K terms organized into a DAG. BioASQ provides this DAG as a parent child list. Inspection of this graph through the python networkx library revealed the properties shown in figure 5. The longest path was determined to be 17.

```
Type: DiGraph                    Summary of shortest paths to 18714 leaves:
Number of nodes: 27900           AVG: 5.75
Number of edges: 37796           MIN:  2
Average in degree:   1.3547      MAX:  13
Average out degree:  1.3547
```

Figure 5: Summary of properties of the MeSH DAG.

3.1.2 Dataset

BioASQ provides two versions of the 2016 training set, both in JSON. The first is a 20.9 GB set with 12.2 M mesh-labeled abstracts (along with other information such as title, journal, and pubmed id). The second is a 8.5 GB subset with 4.9 M abstracts from a limited list of journals. The unlabeled test sets provided on BioASQ use the same limited list of journals. While initial efforts went into working with the complete dataset, the difficulties involved with working with the larger dataset led to the use of the limited one.

Analysis of the number of MeSH terms used to label each training example revealed an average of 13 labels with considerable variation (figures 6 and 7). This variation confounds efforts to predict labels, as the number of labels is not known in advance. Moreover, brute-force approaches to minimize cost over all possible numbers of labels

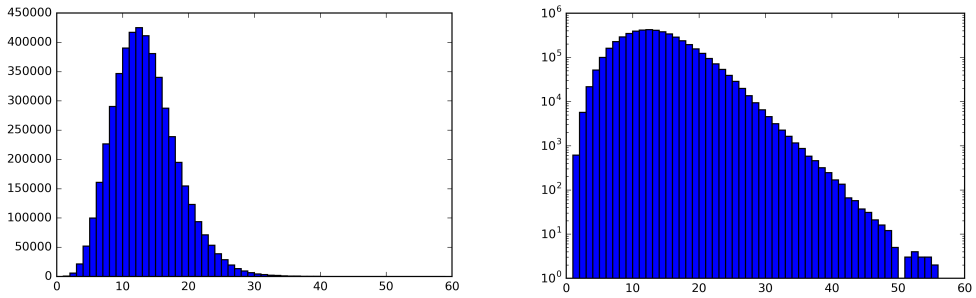

Figure 6: Distribution of MeSH label counts for the limited 8.5 GB dataset of abstracts (standard scale; log scale).

```
count:  4917245
min max: 1 60
avg:  13.01
med:  13.0
var std:  23.27 4.82
```

Figure 7: Corresponding summary statistics for figure 6 (MeSH label counts).

Additional analysis of some 154 million labels in the complete 20.9 GB dataset found a bimodal distribution of shortest paths (figures 8 and 9). This finding suggests that an intelligent search algorithm should be able to achieve higher performance relative to brute-force sampling of all 27.9 K possible MeSH terms—many of the terms are only

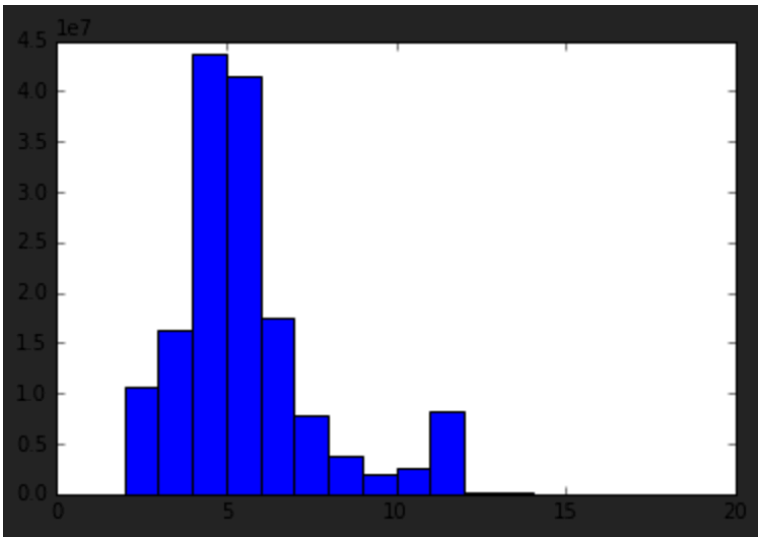

Figure 8: Distribution of shortest path lengths for all 154,100,126 labels encountered in the complete 20.9 GB training set. The distribution appears to be bimodal.

```
count:  154100126
min max: 2 13
avg:  5.05
med:  5.0
var std:  4.48 2.12
```

Figure 9: Summary statistics for the shortest path lengths for all 154,100,126 labels encountered in the complete 20.9 GB training set.

3.1.3 Word vectors

Initial word vectors (WordVecs) were provided as a resource from BioASQ and described at http://participants-area.bioasq.org/info/BioASQword2vec/. Briefly, ~10.9 million abstracts were striped of punctuation and converted to lower case. The resulting corpus had ~9.4 million unique words. The Gensim implementation of WV was then applied to the corpus, which yielded 1.7 million 200-dimensional WordVecs (each corresponding word had a minimum of 5 mentions in the corpus). The corresponding 3.5 GB space-delimited file had one vector per line; a corresponding file with a sorted list of words in the vocab indexed these vectors.

3.2 Processing

3.1.1 Creating AbsVecs

There are many possible avenues to generating vector representations of abstracts (AbsVecs). The simplest of these is to take a uniform average of the WordVecs for each word in the abstract, which is ultimately the approach that was pursued in this paper. This approach gives very coarse vectors because words that are simply mentioned more often will tend to dominate the average, resulting in low signal-to-noise. An improvement on this approach is to use a similar but weighted average of WordVecs. Because the task seeks to index documents with broad terms that capture what the document is about, a term-

frequency inverse document-frequency (TF-IDF) weighting that effectively identifies keywords makes appropriate sense. Ultimately, an approach such as PV or RNNs to better estimate the weights on the WordVecs would likely yield even better composite representations.

To generate AbsVecs for each training example, raw abstract text was stripped of punctuation and converted to lowercase. The resulting words were looked up in the BioASQ provided vocab to obtain a word index, which in turn was used to locate the corresponding WordVec. The WordVecs for each word in a given abstract were then averaged to create an AbsVec (one per training example). Like the comprising WordVecs, each AbsVec was 200-dimensional.

3.1.2 Creating TermVecs

As with generating AbsVecs, there are many possible ways to create vector representations of MeSH terms (TermVecs). Again, the simplest possibility to generate a TermVec (and the one that was pursued in this paper) is to take all of the AbsVecs associated with a given term and to average them. This approach has a disadvantage in that it will not produce a TermVec for any term not used in the training corpus. A better approach would be to merge the average of a term's AbsVecs the average of its children's TermVecs. A simple 50-50 weighting structure, would be the first thing to try herealthough a neural network could also. Indeed, the hierarchical structure of the MeSH resembles that of a Recursive Neural Network, so it appears likely that an RNN-like arch.

Like the comprising AbsVecs, each TermVec was 200-dimensional.

3.1.3 Predicting MeSH labels

Regrettably, while I was able to produce a basic set of training AbsVecs and TermVecs, I failed to generate a set of validation or test AbsVecs within the constraints of the project timeline. Given additional time, I would have liked to have *a)* performed 3-fold cross-validation when generating TermVecs from the training AbsVecs (such code to generate 3-fold TermVecs from AbsVecs is included in the submission), and *b)* generated AbsVecs from unlabeled test data to make predictions for submission to BioASQ.

To make a prediction for an unlabeled test AbsVec, I would have calculated a list of TermVecs ranked and sorted on their distance to the AbsVec (e.g. via cosine similarity or Euclidean distance). If necessary, a threshold could be applied to the ranked list (e.g. top 10%). Then, given a fixed number of labels, I would have attempted to find the pruning of that ranked list that gave the highest combined while enforcing the NLM-dictated constraint that any predicted term have no predecessors (ancestors) that are also predictions (e.g. "cancer" would not be a valid label if "lung-cancer" was a superior prediction). In the case of validation, the cost of the predicted terms could be established using accuracy (flat metric) and F-LCA (hierarchical metric).

4 Conclusions

Progress on the project was greatly hampered by the size of the datasets involved (20.9 GB for the complete dataset, 8.5 GB for the 3.5 GB for the WordVectors). Downloading and decompressing the data proved alone proved to be nontrivial, as BioASQ used a compression protocol not documented on their website. Over the course of the project, a significant amount of time was also invested into working with the large 20.9 GB dataset. After struggling with the size of the dataset (which would not fit in RAM), an effort was made to organize data and resulting calculations into a sqlite database for faster querying; however, while the database was successfully created, this amounted to a dead end effort with little useful speedup, and ultimately the decision was made to use the smaller, more limited 8.5 GB dataset. Future work would greatly benefit from better organization and access to the large datasets.

Nonetheless, the work above has produced a set of AbsVecs (each the average of comprising

WordVecs) and TermVecs (each the average of comprising AbsVecs) for the limited training set data provided by BioASQ. The usefulness of these two resources remains to be evaluated. As mentioned, both the generation of the AbsVecs and the TermVecs could easily be improved beyond the by using smarter weighting (e.g. by using TF-IDF for AbsVecs or propagation of TermVecs through the DAG) or by using a neural architecture to learn appropriate weights. More generally, an integrated learning model that was trained to give task specific (rather than general purpose) AbsVecs and TermVecs based directly on predictive performance (e.g. measured by F-LCA) is also likely to give better results; however, it remains unclear to me how to adapt deep networks to the unique challenges of multilabel hierarchical classification (at least in an efficient way that takes advantage of the hierarchical structure).

There are several additional possibilities for improvement. The MeSHLearner model established that other abstract metadata can providing invaluable features for better classification [5]. This finding makes tremendous sense—for instance, certain journals (e.g. "Cancer") are likely to have very common MeSH annotations. Rather than comparing representations of abstracts against representations of terms, it may ultimately be more helpful to conceptualize embedding abstracts as some representation of a DAG and attempt to reconcile the predicted DAG with some pruning of the full MeSH DAG structure (inspiration from [6]).

Acknowledgments

The author thanks the CS224d teaching assistants for their hard work this quarter as well as Dr. Socher for his excellent lectures and guidance.

References

[1] G. Tsatsaronis, et al: *An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition*. Apr. 30, 2015. BMC Bioinformatics.

[2] T. Mikolov, et al: *Efficient estimation of word representations in vector space*. Version 3. Sept. 7, 2013. arXiv:1301.3781v3. [cs.CL]

[3] Q. Le & T. Mikolov: *Distributed representations of sentences and documents*. Version 2. May 22, 2014. arXiv:1405.4053v2 [cs.CL]

[4] A. Kosmopoulos, et al: *Evaluation measures for hierarchical classification: a unified view and novel approaches*. Version 2. Jul. 1, 2013. arXiv: 1306.6802v2 [cs.AI]

[5] K. Liu, et al: *MeSHLabeler: improving the accuracy of large-scale MeSH indexing by integrating diverse evidence*. Jun., 2015. Bioinformatics.

[6] S. Bengio, J. Weston, & D. Grangier: *Label embedding trees for large multi-class tasks*. 2010. Advances in Neural Information Processing Systems 27, 163-171.