Structured Multi-Label Biomedical Text Tagging
via Attentive Neural Tree Decoding

Gaurav Singh
University College London
gaurav.singh.15@ucl.ac.uk

James Thomas
University College London
james.thomas@ucl.ac.uk

Iain J. Marshall
King's College London
iain.marshall@kcl.ac.uk

John Shawe-Taylor
University College London
j.shawe-taylor@ucl.ac.uk

Byron C. Wallace
Northeastern University
b.wallace@northeastern.edu

Abstract

We propose a model for tagging unstructured texts with an arbitrary number of terms drawn from a tree-structured vocabulary (i.e., an ontology). We treat this as a special case of sequence-to-sequence learning in which the decoder begins at the root node of an ontological tree and recursively elects to expand child nodes as a function of the input text, the current node, and the latent decoder state. In our experiments the proposed method outperforms state-of-the-art approaches on the important task of automatically assigning MeSH terms to biomedical abstracts.

1 Introduction

We consider the task of multilabel text annotation, where labels are drawn from an ontology. We are motivated by problems in biomedical NLP (Zweigenbaum et al., 2007; Demner-Fushman et al., 2016). Specifically, scientific abstracts in this domain are typically associated with multiple Medical Subject Heading (MeSH) terms. MeSH is a controlled, hierarchically structured vocabulary that facilitates semantic labeling of texts at varying levels of granularity. This in turn supports semantic indexing of biomedical literature, thus facilitating improved search and retrieval.[1]

At present, MeSH annotation is largely performed manually by highly skilled annotators employed by the National Library of Medicine (NLM). Automating this annotation task is thus highly desirable, and there have been considerable efforts to do so. The BIOASQ[2] challenge, in particular, concerns MeSH annotation, and competitive systems have emerged from this in past years (Liu et al., 2014; Tsoumakas et al., 2013); these constitute baseline approaches in the present work.
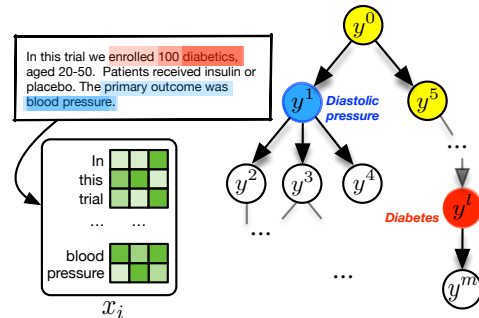

Figure 1: Illustration of the proposed Neural Tree Decoding (NTD) model. Input text is encoded, and a decoder then conditionally traverses the label tree to select all relevant nodes to apply, with node-wise attention induced over the input text.

More generally, MeSH annotation is a specific instance of multi-label classification, which has received substantial attention in general (Elisseeff and Weston, 2002; Fürnkranz et al., 2008; Read et al., 2011; Bhatia et al., 2015; Daumé III et al., 2017; Chen et al., 2017; Jernite et al., 2016). Our work differs from these prior efforts in that MeSH tagging involves *structured* multi-label classification: the label space is a tree[3] in which nodes represent nested semantic concepts, and the specificity of these increases with depth.

Past efforts in multi-label classification have considered hierarchical and tree-based approaches for tagging (Jernite et al., 2016; Beygelzimer et al., 2009; Daumé III et al., 2017), but these have not assumed a *given* structured label space; instead, these efforts have attempted to *induce* trees to improve inference efficiency. By contrast, we propose to explicitly capitalize on a known output structure codified here by the target ontology from which tags are drawn. We realize this by recursively traversing the tree to make (conditional) bi-

---

[1]This problem also resembles tagging clinical notes with ICD codes (Mullenbach et al., 2018).

[2]http://bioasq.org/

[3]Technically, MeSH comprises multiple trees, but we join these by insertion of an overarching root node.

nary tag application predictions.

The contribution of this work is a neural sequence-to-sequence (*seq2seq*) model (Bahdanau et al., 2014) for structured multi-label classification. Our approach entails encoding the input text to be tagged using an RNN, and then *decoding into the ontological output space*. This involves a tree traversal beginning at the root of the tree. At each step, the decoder decides whether to 'expand' children as a function of a hidden state vector, node embeddings, and induced attention weights over the input text. This approach is schematized in Figure 1. Expanded nodes are added to the predicted tag set. This process is repeated recursively until either leaf nodes are reached or no children are selected for expansion. This neural tree decoding (NTD) model outperforms state-of-the-art models for MeSH tagging.

2 Model

Overview. Our model is an instance of an encoder-decoder architecture. For the encoder, we adopt a standard Gated Recurrent Unit (GRU) network (Cho et al., 2014a), which yields hidden states for the tokens comprising an input document. The decoder network consumes these outputs and begins at the root of the ontological tree. It induces an attention distribution over encoder states, which is used together with the current decoder state vector to inform which (if any) of its immediate children are applicable to the input text (Figure 1). This decoding process proceeds recursively for all children deemed relevant. Below we provide more in-depth technical detail regarding the constituent modules.

The encoder (ENC) consumes as input a raw sequence of words, here composing an abstract. These are passed through an embedding layer, producing a sequence of word embeddings x (for clarity we omit a document index here), which are then passed through a GRU (Cho et al., 2014b) to obtain a sequence of hidden vectors $h = \{h_0, \cdots, h_{|x|-1}\}$, where $h_t = \text{GRU}(x_t, h_{t-1})$.

These are then passed to our neural tree decoder, which is responsible for tagging the encoded text with an arbitrary number of terms from the label tree, i.e., sequences in the structured output space. This module traverses the label space top-down, beginning at the root, thus exploiting the concept hierarchy codified by the tree structure.

At each step in the decoding process, the decoder will be positioned at a particular node in the tree n. Children — immediate descendents — of this node are then considered for expansion in turn, based on a hidden state vector s_n, and a context vector c_n. Both of these are initialized to zero vectors and recursively updated during traversal, i.e., as nodes are selected for expansion (and hence added to the predicted tag set). More specifically, the context vector that informs the decision to expand node v in the label hierarchy from its parent node n is a weighted sum of the encoder hidden states h, where weights reflect induced attention over inputs, conditioned on n. That is:

$$c_n = \sum_j \alpha_{nj} h_j \qquad (1)$$

where

$$\alpha_{nj} = \frac{exp\{a(s_n, h_j)|\theta_n\}}{\sum_l exp\{a(s_n, h_l)|\theta_n\}} \qquad (2)$$

and a is a simple multi-layer perceptron (MLP), with node-specific parameters θ_n. Here both sums range over the length of the input text.

Given c_n, we then estimate the probability that child label v is applicable to the current input text as a function of the decoder state vector (s_n), the current context vector (c_n) and the decoder parameters. In particular, this is realized via a standard linear layer with sigmoid activations, parameterized by a weight matrix W comprising independent weight vectors for each output node v. Thus the score for a particular output node v is $\sigma(W_v \cdot [s_n, c_n])$, where W_v denotes the weight vector for output node v.

Pseudocode for the training and decoding procedures are presented in Algorithm 1. In the NODELOSS function, n denotes a particular node. The set of hidden vectors induced by the encoder (corresponding to the inputs) are denoted by h, s is the hidden state of the decoder, and y is the reference label (this encodes a path in the output tree). We assume the decoder, DEC, consumes input representations, a node index and a hidden state and yields a context vector for n, c_n and an updated state vector s_n; in our case the latter is implemented via a GRU. The advantage of using an RNN during decoding is that this allows the exploitation of learned, distributed hidden representations of partial tree paths, which inform node-wise attention and subsequent predictions.

Algorithm 1 RECURSIVETREEDECODING

1: **function** NODELOSS(n, h, s, y)
2:     $l_n \leftarrow 0$
3:     $c_n, s_n \leftarrow$ DEC(h, n, s)
4:     **for each** child $v \in$ children(n) **do**
5:         $\hat{y}_v \leftarrow \sigma(W_v \cdot [s_n, c_n])$
6:         $p_v \leftarrow \propto$ depth in tree
7:         $B_v \sim$ Ber(p_v)
8:         **if** B_v **then**
9:             $l_n \leftarrow l_n + \mathcal{L}(\hat{y}_v, y)$
10:        **if** $\hat{y}_v > \tau$ **then**
11:            $l_n \leftarrow l_n +$ NODELOSS(v, h, s_n, y)
        return l_n

12: **function** TRAIN(x, y, α, epochs)
13:     $\theta \leftarrow$ INIT(θ)
14:     $e \leftarrow 0$
15:     **while** $e <$ epochs **do**
16:         **for each** instance $x_i \in x$ **do**
17:             $h_i \leftarrow$ ENC(x_i)
18:             $s_0 \leftarrow \mathbf{0}$
19:             $l_i \leftarrow$ NODELOSS(ROOT, h_i, s_0, y_i)
20:             $\Delta\theta \leftarrow$ BACKPROP(l_i)
21:             $\theta \leftarrow \theta + \alpha\Delta\theta$
22:         $e \leftarrow e + 1$
        return θ

Incurring loss for all nodes along the path specified by y would place a disproportionate amount of emphasis on correctly applying terms that are 'higher' in the ontology, as loss will be propagated for the initial predictions concerning the application of these and then also, due to recursive application, for all of their children (and so on). Thus we only incur (and hence backpropagate) loss for a node v stochastically, according to a Bernoulli distribution B with parameter p_v. We set p_v to be proportional to the depth of node v in the tree such that we are likely to incur larger loss for deeper (rarely occurring) nodes. We operationalize this as: $p_v = min(1, 0.5 + \frac{m}{f_v})$, where m is the count corresponding to the least frequently observed node in the training corpus and f_v is the count for node v. In Section 4 we demonstrate the benefit of this approach.

At train time we use *teacher forcing* (Williams and Zipser, 1989) during decoding. That is, we revert the model back to the correct (training) tree subsequence when it goes off-course, and continue decoding from there. We have elided this detail from the pseudocode for clarity.

3 Experimental setup

Below we describe experimental details concerning our implementation, datasets and baselines. Code and data to reproduce our results is available at https://github.com/gauravsc/NTD.

3.1 Implementation Details

We limited the vocabulary to the $50,000$ most frequent words. Word embeddings were initialized to pre-trained vectors induced via word2vec, trained over a large set of abstracts indexed on PubMed.[4] Ontology node embeddings were pre-trained using DeepWalk (Perozzi et al., 2014), fit over PubMed.

3.2 Dataset

Our dataset comprises abstracts of articles describing randomized controlled trials (RCTs) from PubMed along with their MeSH terms. The MeSH annotations were manually applied by professionals at the National Library of Medicine (NLM). The label space underlying MeSH terms is codified by a publicly available ontology.[5]

We split this dataset into disjoint sets for training/development and final evaluation (Table 1). We further separated the former into train, validation and development test subsets, to refine our approach. For our final evaluation we used a held-out set of 10,000 abstracts that were not seen in any way during model development and/or hyperparameter tuning. We performed extensive hyperparameter tuning for the baseline models to ensure fair comparison; details regarding this tuning are provided in the Appendix.

3.3 Baselines

We compare our proposed approach to three baselines, including two prior winners of the annual BioASQ challenge, which includes an automated MeSH annotation task. However, it is important to note that we used a different (and considerably smaller) dataset in the current work, as compared to the corpus used in the BioASQ challenge.

LSSI (Tsoumakas et al., 2013) use an approach that involves predicting both the number of terms and which to apply to a given abstract. They use linear models for both tasks, which operate over TF-IDF representations of abstracts. Specifically, they train a regressor to predict k, the number of MeSH terms to be applied to an abstract. Simultaneously, a binary linear SVM is trained independently for each MeSH term appearing in the train set. At test time, these SVMs provide scores for each term and the top $\hat{k}$ terms are applied, where $\hat{k}$ is the estimate from the aforementioned regressor.

---

[4]A repository of biomedical literature.
[5]https://meshb-prev.nlm.nih.gov/treeView

Train	20000
Validation	4000
Dev test	18884
Test (held-out)	10000
Mean MeSH terms per article	15.33
Total unique MeSH terms	27892
Unique MeSH terms in dataset	3781

Table 1: Dataset statistics.

Method	Precision	Recall	F1	SD
LSSI	0.326	0.293	0.309	1.518
UIUC	0.236	**0.388**	0.291	1.433
DML	0.378	0.223	0.275	1.516
NTD-d	**0.434**	0.235	0.299	1.209
NTD-s	0.425	0.265	**0.327**	**1.130**

Table 2: Results on the held-out test dataset. SD refers to *semantic distance*, defined in Eq. 3.

UIUC (Liu et al., 2014) uses a learning-to-rank model to identify the top MeSH terms for an abstract from a candidate set of terms, which is obtained from the nearest neighbours of the abstract. Additionally, one SVM classifier is trained for each of the MeSH terms (similar to the above approach), and scores for each are used to obtain additional terms to be added to the candidate set. In the end, a threshold (tuned on the validation set) is used to select the final set of terms to be assigned.

Finally, we consider a deep multilabel classification model **DML** (Rios and Kavuluru, 2015) that takes as input unstructured abstracts and activates the output nodes corresponding to the relevant MeSH terms. In brief, embedded tokens are fed through a CNN to induce a vector representation, which is then passed on to the dense output layer. Finally, this is passed through a sigmoid activation function. Note that this model exploits the same pre-trained word embeddings as our model does.

3.4 Evaluation metrics

We first evaluate model performance via output node-wise precision, recall and F1 measure. However, these metrics are overly strict in the sense that a model will be penalized equally for all mistakes, regardless of whether they are nearby or far from the target in the label tree. This is problematic because whether to apply a specific MeSH term or its immediate parent may be somewhat subjective in practice. To quantify this, and to explore the extent to which explicitly decoding into the target label space yields improved predictions, we also consider a measure that we refer to as *semantic distance* (SD):

$$ \text{SD} = \frac{1}{|\mathcal{Y}|} \sum_{u \in \mathcal{Y}} \min_{v \in \hat{\mathcal{Y}}} dist(u, v) \qquad (3) $$

where $\mathcal{Y}$ and $\hat{\mathcal{Y}}$ are the sets of target and predicted terms respectively, and *dist* is a function that returns the shortest distance between two nodes in

the label ontology tree. The idea is that this penalizes less for 'near misses'. Thus if a model fails to apply a particular tag t, but does apply one near to t in the label tree, then it is penalized less.[6] We hypothesize that our model will improve results markedly with respect to this metric, given our exploitation of the tree structure.

As in the case of recall, SD can be 'gamed': one can achieve a perfect score by predicting that all nodes apply to a given abstract. Thus this is only meaningful alongside complementary metrics like F1.

4 Results

Results on the test set (which was completely held out during development) are reported in Table 2. The proposed Neural Tree Decoding model with stochastic backpropagation (NTD-s) bests the most competitive baseline (LSSI) in F1 score by over 2 points.

To explore the effect of backpropagating loss from nodes in proportion to their depth in the ontology, we also include results for a deterministic variant that does not do this, NTD-d. This version does not perform as well, demonstrating the utility of the proposed training approach.

The metrics reported thus far do not account of the structure in the output space. We thus additionally report results with respect to the the semantic distance (SD) metric (Eq. 3). We observe a marked performance increase of ~21% over the best performing baseline. This is intuitive given that we are explicitly decoding into the label tree structure, and demonstrates the ability of our model to learn the ontological structure, thereby predicting semantically appropriate terms.

---

[6]This metric is equivalent to the sum of two metrics ("divergent path to gold standard" and "divergent path to prediction") defined in (Perotte et al., 2013).

5 Conclusions, Discussion & Limitations

We developed a neural attentive sequence tree decoding model for structured multilabel classification where labels are drawn from a known ontology. The proposed method can decode an input text into a tree of labels, effectively using the structure in the output space. We demonstrated that this model outperformed SOTA approaches for the important task of tagging biomedical abstracts with Medical Subject Heading (MeSH) terms on a modestly sized training corpus. Code and data to reproduce these results are available at https://github.com/gauravsc/NTD.

One limitation of our model is that it is comparatively slow, due to having to traverse the tree structure during decoding. Prediction speed may not be a major issue in practice, as articles on PubMed could be batch tagged nightly as they arrive. However, slow decoding also means lengthy training (see Appendix, section A.2 for details). For this reason we have here used a modest training set of ∼20k abstracts, which is smaller than corpora used in prior work on this task. Given the relative expressiveness of our model, we expect it to benefit substantially from additional training data, moreso than the simpler baseline architectures. But at present this is only a conjecture.

In future work we thus hope to apply this model to larger datasets, and to address the efficiency issue. Concerning the latter, sibling subtrees may be traversed in parallel, conditioned on the hidden state of their parent. Another promising direction would be to move to convolutional encoder and decoder architectures, designing the latter in a way similarly capitalizes on the label space tree structure.

6 Acknowledgements

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Alina Beygelzimer, John Langford, Yuri Lifshits, Gregory Sorkin, and Alex Strehl. 2009. Conditional probability tree estimation analysis and algorithms. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*.

Sheng Chen, Akshay Soni, Aasish Pappu, and Yashar Mehdad. 2017. Doctag2vec: An embedding based multi-label learning approach for document tagging. In *Proceedings of the Workshop on Representation Learning for NLP*. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014a. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014b. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Hal Daumé III, Nikos Karampatziakis, John Langford, and Paul Mineiro. 2017. Logarithmic time one-against-some. In *International Conference on Machine Learning*.

D Demner-Fushman, N Elhadad, et al. 2016. Aspiring to unintended consequences of natural language processing: A review of recent developments in clinical and consumer-generated text processing. *IMIA Yearbook*.

André Elisseeff and Jason Weston. 2002. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems*.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. 2008. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2).

Yacine Jernite, Anna Choromanska, and David Sontag. 2016. Simultaneous learning of trees and representations for extreme classification, with application to language modeling. *arXiv preprint arXiv:1610.04658*.

Ke Liu, Junqiu Wu, Shengwen Peng, Chengxiang Zhai, and Shanfeng Zhu. 2014. The fudan-uiuc participation in the bioasq challenge task 2a: The antinomyra system. *Risk*, 129816.

James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *North American Chapter of the Association for Computational Linguistics*.

Adler Perotte, Rimma Pivovarov, Karthik Natarajan, Nicole Weiskopf, Frank Wood, and Noémie Elhadad. 2013. Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association*, 21(2).

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3).

Anthony Rios and Ramakanth Kavuluru. 2015. Convolutional neural networks for biomedical text classification: application in indexing biomedical articles. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Health Informatics*. ACM.

Grigorios Tsoumakas, Manos Laliotis, Nikos Markantonatos, and Ioannis Vlahavas. 2013. Large-scale semantic indexing of biomedical publications at BioASQ. In *BioASQ Workshop*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2).

Pierre Zweigenbaum, Dina Demner-Fushman, Hong Yu, and Kevin B Cohen. 2007. Frontiers of biomedical text mining: current progress. *Briefings in bioinformatics*, 8(5).