Interaction Matching for Long-Tail Multi-Label Classification

Sean MacAvaney,[1] Franck Dernoncourt,[2] Walter Chang,[2] Nazli Goharian,[1] Ophir Frieder[1]
[1] IR Lab, Georgetown University, [2] Adobe Research
{sean,nazli,ophir}@ir.cs.georgetown.edu
{franck.dernoncourt,wachang}@adobe.com

Abstract

We present an elegant and effective approach for addressing limitations in existing multi-label classification models by incorporating interaction matching, a concept shown to be useful for ad-hoc search result ranking. By performing soft n-gram interaction matching, we match labels with natural language descriptions (which are common to have in most multi-labeling tasks). Our approach can be used to enhance existing multi-label classification approaches, which are biased toward frequently-occurring labels. We evaluate our approach on two challenging tasks: automatic medical coding of clinical notes and automatic labeling of entities from software tutorial text. Our results show that our method can yield up to an 11% relative improvement in macro performance, with most of the gains stemming from labels that appear infrequently in the training set (i.e., the long tail of labels).

Introduction

Multi-label text classification (i.e., the task of assigning a variable number labels to a piece of text) is a classic task with a variety of practical applications. For instance, a clinical report could be tagged with medical codes, describing a patient's diagnoses (e.g., *Lyme disease*) and procedures (e.g., *clipping of aneurysm*). Since both the note and the medical codes are required in the clinical process, a system that can automatically derive these medical codes from notes would be a valuable time-saving measure. As another example, software tutorial text can be semantically labeled with the tools that are used to accomplish the task. These labels could provide additional support to users trying to replicate a tutorial, or be used to improve search engines by indexing on these labels.

Text labeling approaches rely on machine learning techniques to rank a given set of labels for a piece of text. For example, supervised FastText (Joulin et al. 2017) learns dense label and document term representations. At inference time, it compares the label representations to the representation of an unseen document to produce label scores. Others have employed conceptually similar yet more sophisticated approaches, such as using convolutional neural networks and attention mechanisms in a similar fashion (Mullenbach et al. 2018; Liu et al. 2017). One limitation of these approaches is that the models fail to effectively rank infrequently-occurring labels due to inadequate variability in the training data. These approaches are also unable to handle labels that do not occur in training data (e.g., extremely rare or new labels).

We present a text labeling approach that uses soft n-gram interaction matching, an approach inspired by recent work in ad-hoc ranking (Pang et al. 2016; Hui et al. 2018). This allows handling of labels that have meaningful natural language names, while not necessarily occurring frequently in training data. It is common to have label names in multi-labeling tasks, as these are used by humans to manually perform the labeling (e.g., medical codes have descriptions). Our approach, which handles infrequent labels, can be combined with existing labeling techniques that handle frequently-occurring labels. We show that our approach is effective at two tasks, each with a large number of labels: automatic medical coding of clinical reports (1,159 labels), and automatic labeling of tools in software tutorials (831 labels).

In summary, our contributions are: (1) an approach for extending multi-label classification models, based on soft n-gram interaction matching; (2) an evaluation on two datasets, showing that our approach can be effectively combined with other leading classification approaches; and (3) an analysis demonstrating our capacity to identify long tail labels, even those without training samples.

Background & Related Work

Multi-label text classification. This is a well-studied task with a multitude of prior work. Among the most notable recent efforts are supervised FastText (Joulin et al. 2017), which learns embeddings for labels that can be compared to document representations. Earlier work by Kim (2014) showed that a simple convolutional neural network (CNN) with dynamic pooling can be effective for text classification. Berger (2015) shows that recurrent neural networks (RNNs) can also be used for classification. Johnson and Zhang (2015) uses n-gram indicator variables fed into a deep neural network to make classification decisions. Yen et al. (2016) addresses training data sparsity by enforcing heavy regularization penalties, but falls short of handling extremely infrequent labels. Liu et al. (2017) attempts to

	Create	another	new	document	(	I	chose	600	x	400	px	for	width	and	height	)	,	select	the	brush	tool	,	and	open	the	brush	preset	panel	.
brush	0.3	0.3	0.2	0.1	0.1	0.3	0.1	0.2	0.1	0.2	0.0	0.2	0.3	0.4	0.3	0.2	0.2	0.1	0.3	1.0	0.2	0.2	0.4	0.3	0.3	1.0	0.0	0.1	0.3
tool	0.6	0.4	0.4	0.5	0.2	0.3	0.2	0.2	0.3	0.1	0.0	0.5	0.1	0.4	0.2	0.3	0.4	0.4	0.4	0.2	1.0	0.4	0.4	0.2	0.4	0.2	0.1	0.4	0.4
Score	0.4	0.4	0.3	0.1	0.2	0.2	0.1	0.2	0.1	0.1	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.2	0.3	**1.0**	0.3	0.3	0.3	0.3	0.3	0.6	0.2	0.3	0.1

Figure 1: Example similarity matrix and interaction matching scores for label “brush tool”.

address label sparsity by using shared intermediate representations from a CNN. Gehrmann et al. (2018) interpreted CNNs' classification by defining a salience score for each token of the sentence input. Others have shown that using attention can further improve performance, and improve explainability of decisions in the medical domain (Mullenbach et al. 2018; Xie et al. 2018). Jain et al. (2019) shows that millions of labels can be practically handled using a pruning strategy. These approaches are limited by the variability of labels in the training data, or exact term matching. None of these approaches allow for soft label text matching, allowing for soft matching of unseen labels.

Soft n-gram interaction matching. *Interaction-focused ranking models* formulate document ranking as a learning-to-rank problem over a term similarity matrix between a query and a document. One successful approach to learn these patterns is by applying square convolutional kernels over the term similarity matrix (e.g. Pang et al. (2016); Hui et al. (2018)), a process called *soft-ngram interaction matching*. We propose an approach inspired by these methods for multi-label text classification; we use the approach to rank labels for a given segment of text, rather than documents for a query. Due to the large number of labels, we use a fixed sequence, rather than allowing the model to learn out-of-order n-grams. We also normalize the scores based on the label length. In our preliminary work, we found this a necessary optimization for scalability, without impacting performance; this is not necessary in document ranking models because the same query is used for each document, and thus has the same length.

Methodology

Notation. Let T be a sequence of tokens in a document, and let $\mathbf{L}$ be a collection of labels. For multi-label text classification, a score $V^{L_i,T}$ is generated for each label $L_i \in \mathbf{L}$. The labels are ranked by this score, optionally employing a score threshold and/or a maximum count to select the labels.

For our task, we assume each label consists of a sequence of tokens representing its name in natural language: $L_i = \{L_{i,1}, L_{i,2}, ..., L_{i,|L_i|}\}$. This is a reasonable assumption, given that the labels are usually produced for humans, who will often need a name to reason about the label. For our method to be most effective, the names should have terms that may have approximate matches in the text. For instance, given the procedure label of *clipping of aneurysm*, an approximate match found in the text might be *clip the aneurysm*. Our method uses the term similarity matrix $S^{L_i,T} \in \mathbb{R}^{|L_i|,|T|}$ between label L_i and text T as input:

$$S^{L_i,T}_{j,k} = \cos(embed(L_{i,j}), embed(T_k)) \tag{1}$$

where $\cos(\cdot)$ is the cosine similarity score and $embed(\cdot)$ returns the token's word embedding. Note that each similarity score here is a unigram match; our model operates over this matrix to perform n-gram matching. An example similarity matrix is shown in Figure 1.

Interaction matching model. Inspired by recent interaction-focused ranking models in information retrieval, we apply square convolutions over the similarity matrix to produce soft n-gram matching scores. Unlike document ranking models, however, we impose a single fixed sequential convolution kernel over the labels. In other words, we use the identity matrix $I_{|L_i|}$ as a convolutional kernel. We then take the maximum scores from each kernel and normalize them by the length of the label $|L_i|$. This step is not taken in the document ranking models but necessary in this context because multiple labels of different lengths are being matched over the same document. Note that since the convolutional kernel matches similarity scores, exact matches are not required; this is what makes the n-grams ‘soft'. More formally, our method generates a label score for each document position $P^{L_i,T} \in \mathbb{R}^{|T|}$ for document T and label L_i:

$$P^{L_i,T} = \sigma\left(\frac{I_{|L_i|} \star S^{L_i,T}}{|L_i|}\right) \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid activation function and $\star$ performs 2-dimensional convolution. For simplicity, we assume $\star$ applies padding where necessary.

To generate a label score for the entire text, we perform max pooling: $V^{L_i,T} = \max_{j=1}^{|T|} P^{L_i,T}_j$. The use of max pooling allows for the soft n-gram to match anywhere in the document and for the score not to be influenced by document length (as opposed to average pooling, for instance). Furthermore, the $\arg\max$ yields an interpretable grounding of the model's decision within the text and can be used to aid in the explanation of the model's decision. At inference time, all labels in $\mathbf{L}$ are ranked using this method. This approach is trivially parallelizable and easily handles datasets with thousands of labels. An example of interaction matching scores are shown in Figure 1. In this example, the exact match is given a perfect matching score of 1.

The model's structure allows the interaction model to easily incorporate new labels to be introduced after training simply by adding to $\mathbf{L}$. In our experiments, we train the model by back-propagating errors to the word embeddings. We recognize that our model is ineffective for labels

Medical Coding	**Software Tutorial**
Report: ...status post tracheostomy for paradoxical vocal cord motion with asthma discharge medications fenofexadine mg po q day calcium carbonate grams po t i d percocet one po q to hours prn pain... **ICD-9 labels:** Other diseases of upper respiratory tract ; Asthma ; Diabetes mellitus	**Tutorial:** Create another new document (I chose 600 x 400 px for width and height), select the brush tool, and open the brush preset panel. **Tool labels:** File > New ; Brush Tool

Figure 2: Example de-identified clinical report excerpt from MIMIC-III, including top-level ICD-9 labels, and sentence from the tutorial dataset with labeled software tools.

	MIMIC-III	Tutorials
# records	112k	40k
# labels	1,159	831
avg. record length	709.3	21.7
avg. labels per record	7.6	1.2

Table 1: Dataset characteristics.

Model	MacroP	MacroR	MacroF1	MicroF1
Bi-RNN	0.1585	0.1486	0.1534	0.5890
+ interaction (ours)	0.1831	0.1233	0.1473	0.5970
CNN	0.1628	0.1562	0.1594	0.5894
+ interaction (ours)	0.1822	0.1745	0.1783	0.6072
CAML	0.2582	0.2235	0.2396	0.6520
+ interaction (ours)	**0.2720**	**0.2310**	**0.2498**	**0.6546**

Table 2: Medical coding performance on MIMIC-III. Our results on the macro metrics for CNN and CAML are significant at $p < 0.05$.

that do not match the text. Thus, we suggest incorporating our method into existing multi-label text classification approaches, which can learn to effectively match labels that frequently occur in the training data. We train the two models jointly, combining them by taking the maximum score for each label.

Experiments

We test our approach on two tasks: medical coding and software tool extraction from online tutorials. While both are multi-label classification tasks, the data characteristics are different for each task, demonstrating that our approach is generally applicable. Examples are given in Figure 2.

Medical coding

Dataset. We first evaluate on the MIMIC-III dataset (Johnson et al. 2016), a large, de-identified, and publicly-available collection of medical records; Each record in the dataset includes ICD-9 codes, which identify diagnoses and procedures performed.[1] Each code is partitioned into sub-codes, which often include specific circumstantial details. We treat the parent (top-level) codes as labels to be identified based on the patient's discharge note. The dataset consists of 112k clinical reports records and 1,159 top-level ICD-9 codes (labels). See Table 1 for further dataset characteristics and Figure 2 for an excerpt of a report with labeled codes. We use the same train/dev/test split used by (Mullenbach et al. 2018), with 1,632 development and 3,372 testing reports. We train word embeddings on MIMIC-III using word2vec (Mikolov et al. 2013), matching the setting of (Mullenbach et al. 2018). Note that this does not preclude the matching of terms unseen in training data; trivially, a larger unlabeled corpus could be employed for training embeddings or binary matching could be used for out-of-vocabulary terms.

Baselines & training. We compare our approach to the state-of-the-art attention-based CAML (Mullenbach et al. 2018) network for medical coding, along with a convolutional neural network text classifier network (Kim 2014) and a bi-directional GRU network as baselines, using the implementations provided by (Mullenbach et al. 2018). These represent strong baselines for this task. We combine our approach with each baseline, training them jointly as described in Section . We train all neural models optimizing cross entropy loss with the Adam optimizer (Kingma and Ba 2015) (learning rate of 10^{-4}). We select a threshold for all labels using MacroF1 performance on the dev set.

Results. Our results on MIMIC-III are presented in Table 2. In line with prior work on the dataset, we measure the performance in terms of micro- and macro-averaged F1 score. Since our focus is improving the long tail of infrequently-occurring labels, we also include macro-averaged precision and recall. Our approach outperforms the state-of-the-art CAML method in terms of macro precision, recall and F1 by 3–5% (relative improvement, significant at $p < 0.05$). The performance improvement on the weaker CNN baseline is even more pronounced, achieving an 8–11% improvement on the macro metrics (also significant at $p < 0.05$). Interestingly, our approach improves the precision for the bi-directional RNN, at the expense of recall. We attribute this to the interaction matching technique that is inherently high-precision. The improvements in the micro metrics are less pronounced, showing that our approach primarily benefits the long tail.

Error analysis. We often found that our method was able to match infrequent labels where CAML had failed. For instance, in one report, our method labeled all three codes correctly (including one that occurs in only 0.5% of training data), while the unmodified CAML method found two of the three correctly, but also mistakenly included a third, completely unrelated label (occurs in about 0.1% of training data). We observed cases where general codes were not

[1] https://mimic.physionet.org/; https://www.cdc.gov/nchs/icd/icd9.htm

	Rank			
	FT	+Inter.	Sentence	Label (Training count)
(a)	2	1	Go to Filter > Texture > Craquelure. Change the Crack Spacing to 13, the Crack Depth to 3, and the Crack Brightness to 8.	`Filter > Texture > Craquelure` (2)
(b)	>10	1	Now in your new layer, using the Radial Gradient Tool, drag a red gradient over the whole document.	`Radial Gradient Tool` (2)
(c)	2	7	Set the duration of frame 2 to .05 seconds	`Animation > Go To > Time` (3)
(d)	>10	>10	Now it is time to create a path P. If we have our path the right click of the mouse and stroke path with brush set 10px hardness of 100%.	`3D > Repousse > Selected Path` (0)

Table 3: Example label rankings from the long tail of labels.

Model	Dev Set	Test Set
FastText	0.5569	0.5444
+ interaction (ours)	↑ **0.6225**	↑ **0.5789**
XML-CNN	0.5882	0.5916
+ interaction (ours)	**0.5910**	**0.5923**
BERT	0.4142	**0.4240**
+ interaction (ours)	↑ **0.4570**	0.4160

Table 4: Average precision (macro-averaged by label) performance on the Tutorial dataset. Significant results that are significant at $p < 0.01$ are indicated by ↑.

matched effectively by either model. For instance, *Other diseases of lung* is difficult to match by both models because it involves more advanced reasoning (i.e., the condition affects the lung, and there isn't another label).

Software tutorial labeling

Dataset. We also evaluate our approach on a collection of software tutorials, labeled with the tools used to complete each step (by sentence). This dataset is collected from online tutorials and manually labeled by sentence with a large collection of software tools (831 in total). The dataset consists of 40k sentences, with an average length of 21.7 tokens and an average number of 1.2 tools labeled per record. An example labeled tutorial sentence is given in Figure 2. Note that the tool mentions can be either explicit (*brush tool* → `Brush Tool`) or implicit (*Create a new document* → `File > New`). The dataset will be made available for validation of our results. We use a random 90/5/5% train/dev/test set split.

Baselines. Since no specialized systems exist for this dataset, we use supervised FastText (Joulin et al. 2017), XML-CNN (Liu et al. 2017), and BERT (Devlin et al. 2020) as baseline approaches for the tutorial dataset. FastText is trained for 100 epochs with 1–3 word n-grams, and XML-CNN and BERT are trained using default settings. We initialize the embeddings for interaction matching using 100-dim GloVe embeddings (Pennington, Socher, and Manning 2014), and fine-tune them during the training process.

Results. We present our results on the tutorial dataset in Table 4. We use Average Precision (AP, macro-averaged by label). This evaluation emphasizes correctness along the long tail (as opposed to a micro average). When applied to FastText, our approach improves the test set performance by 6% (relative improvement, significant at $p < 0.01$). The FastText model achieved a perfect AP score of 1.0 for 55 of the labels found in the test set (meaning it was ranked highest among all labels whenever it appeared), whereas the interaction variant had a perfect score for 69 labels. This is a 25% improvement, most of which came from the less frequent half of the labels. The least frequent quarter saw an even bigger change, from 11 perfect scores to 26 (136%), four of which had no training samples. Our approach also slightly improves the performance on XML-CNN, though the results are not significant at $p < 0.01$. Interestingly, the XML-CNN model appears to hamper performance in the development set. Finally, the BERT model underperforms FastText and XML-CNN. This is likely in part due to the small amount of training data available for many labels.

Qualitatively, we find that the interaction approach improves in situations in which there are similar terms/phrases in the long tail. Examples are given in Table 3. Specifically, in cases in which there is similar text to a label in the sentence, the interaction approach is beneficial (a) and (b). We acknowledge that the interaction mechanism can occasionally have false matches (c), and it does not improve performance when there is no similar text (d).

Conclusion

We presented an approach to enhance existing multi-label classification techniques that employs soft n-gram interaction matching. We demonstrated that the approach is effective at identifying labels in the long tail, which are underrepresented with current state-of-the-art classification approaches. We also showed that the approach can effectively label items that do not appear at all in the training data.

References

Berger, M. J. 2015. Large Scale Multi-label Text Classification with Semantic Word Vectors.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2020. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

Gehrmann, S.; Dernoncourt, F.; Li, Y.; Carlson, E. T.; Wu, J. T.; Welt, J.; Foote Jr, J.; Moseley, E. T.; Grant, D. W.;

Tyler, P. D.; et al. 2018. Comparing deep learning and concept extraction based methods for patient phenotyping from clinical narratives. *PloS one* 13(2): e0192360.

Hui, K.; Yates, A.; Berberich, K.; and de Melo, G. 2018. Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval. In *WSDM*.

Jain, H.; Balasubramanian, V.; Chunduri, B. R.; and Varma, M. 2019. Slice: Scalable Linear Extreme Classifiers Trained on 100 Million Labels for Related Searches. In *WSDM*.

Johnson, A. E. W.; Pollard, T. J.; Shen, L.; wei H. Lehman, L.; Feng, M.; Ghassemi, M. M.; Moody, B.; Szolovits, P.; Celi, L. A.; and Mark, R. G. 2016. MIMIC-III, a freely accessible critical care database. In *Scientific data*.

Johnson, R.; and Zhang, T. 2015. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *HLT-NAACL*.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of Tricks for Efficient Text Classification. In *EACL*.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Liu, J.; Chang, W.-C.; Wu, Y.; and Yang, Y. 2017. Deep Learning for Extreme Multi-label Text Classification. In *SIGIR*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.

Mullenbach, J.; Wiegreffe, S.; Duke, J.; Sun, J.; and Eisenstein, J. 2018. Explainable Prediction of Medical Codes from Clinical Text. In *NAACL-HLT*.

Pang, L.; Lan, Y.; Guo, J.; Xu, J.; Wan, S.; and Cheng, X. 2016. Text Matching as Image Recognition. In *AAAI*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*.

Xie, P.; Shi, H.; Zhang, M.; and Xing, E. P. 2018. A Neural Architecture for Automated ICD Coding. In *ACL*.

Yen, I. E.-H.; Huang, X.; Ravikumar, P.; Zhong, K.; and Dhillon, I. S. 2016. PD-Sparse : A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification. In *ICML*.