# Multi-Stage Rocchio Classification for Large-scale Multi-labeled Text data

Dong-Hyun Lee

Nangman Computing,
117D Garden five Tools, Munjeong-dong Songpa-gu, Seoul, Korea
dhlee347@gmail.com

**Abstract.** Large-scale multi-labeled text classification is an emerging field because real web data have about several millions of samples and about half a million of non-exclusive categories. But this is a challenging task in that it is hard for a single algorithm to achieve both performance and scalability at the same time. In this paper, we propose a Multi-Stage Rocchio Classification (MSRC) that exhibits comparable performance to state-of-the-art methods and also high scalability for large-scale multi-labeled text data. Our method was verified at the third Pascal Challenge on Large-Scale Hierarchical Text Classification (LSHTC3) [3].

**Keywords:** large-scale multi-labeled classification, scalability, rocchio classification

## 1 Introduction

Text classification has been intensively studied for decades as a key technology of data mining. However, real web data, unlike typical benchmark data sets such as Reuter-21578 and RCV1 [1], have about several millions of documents and about half a million of non-exclusive categories. Large-scale multi-labeled text classification is an emerging field for this reason. But this is a challenging task in that it is hard for a single algorithm to achieve both performance and scalability at the same time for this task.

In this paper, we propose a Multi-Stage Rocchio Classification method. Rocchio Classification is a simple method based on similarity between test document and label's centroids [2], but basically cannot support multi-labeled data. So we adapt this method in several stages within which the number of predicted labels grows. When the expected number of labels per document is small ($< 10$), our method is much faster than baseline methods such as K-Nearest Neighbor because the number of stages also can be small. Fortunately, most of multi-labeled datasets satisfy this condition. The performance of our method is much better than baseline methods and comparable to state-of-the-art methods for large-scale dataset, according to the third Pascal Challenge on Large-Scale Hierarchical Text Classification (LSHTC3) [3].

## 2 Methods

### 2.1 Term weighting with BM25

The BM25 is a bag-of-words retrieval function that ranks a set of documents. One of the most prominent instantiations of the function is as follows [2]. Given a query $Q$, containing keywords $q_1, \ldots, q_n$, the BM25 score of a document $D$ is

$$score(D,Q)=\sum_{i=1}^{n} IDF(q_i)\cdot\frac{f(q_i,D)\cdot(k+1)}{f(q_i,D)+k\left(1-b+b\frac{L}{L_{avg}}\right)} \tag{1}$$

where $f(q_i, D)$ is $q_i$'s term frequency in the document $D$, $L$ is the length of the document $D$ in words, and $L_{avg}$ is the average document length in the training corpus. $k$ and $b$ are free parameters. $IDF(q_i)$ is the IDF(inverse document frequency) weight of the query term $q_i$. It is usually computed as

$$IDF(q_i)=log\frac{N-n(q_i)+0.5}{n(q_i)+0.5} \tag{2}$$

where N is the total number of documents in the collection, and $n(q_i)$ is the number of documents containing $q_i$.

In this paper, the BM25 is taken as a term weighting scheme, not as a retrieval function or a similarity measure [4] in order to adapt to Rocchio Classification. For training documents,

$$wtf_{t,d}=\frac{(k+1)tf_{t,d}}{tf_{t,d}+k\left(1-b+b\frac{L_d}{L_{avg}}\right)}\cdot log\frac{N-df_t+0.5}{df_t+0.5} \tag{3}$$

For test documents,

$$wtf_{t,d}=\frac{(k+1)tf_{t,d}}{tf_{t,d}+k\left(1-b+b\frac{L_d}{L_{avg}}\right)} \tag{4}$$

where $tf_{t,d}$ is the term frequency of term t in document d, $wtf_{t,d}$ is the weighted version of $tf_{t,d}$, $df_t$ is the document frequency of term t in the training corpus, and others are the same as above. (In our experiments, $k = 1.5$ and $b = 0.75$)

## 2.2 Rocchio Classification

Rocchio Classification is a simple method to predict the label of test document. At first, the centroid of each category is computed as follows,

$$\overrightarrow{\mu_c} = \frac{1}{|D_c|} \sum_{d \in D_c} \overrightarrow{v_d} \tag{5}$$

where $c$ is the category (label), $D_c$ is the set of all training documents that belong to the category $c$ and $\overrightarrow{v_d}$ is the weighted vector space representation of document d.

Then, the predicted label of test document d is the one which has the smallest Euclidean distance between the document and the centroid as follows,

$$c^* = \underset{c}{argmin} \ \| \overrightarrow{\mu_c} - \overrightarrow{v_d} \| \tag{6}$$

where $c^*$ is the predicted label of the document d [2].

Euclidean distance is a well-known distance measure, but instead we use cosine similarity to take advantage of sparsity in documents. If the centroids are normalized to unit-length as follows,

$$\overrightarrow{\mu_c} = \frac{\sum_{d \in D_c} \overrightarrow{v_d}}{\left\| \sum_{d \in D_c} \overrightarrow{v_d} \right\|} \tag{7}$$

the predicted label of test document d can be computed as follows.

$$c^* = \underset{c}{argmin} \ \overrightarrow{\mu_c} \cdot \overrightarrow{v_d} \tag{8}$$

However, this method is only for single-label problems. For multi-labeled data, several labels which centroids are top ranked in similarity are also candidates for the predicted labels.

## 2.3 Rocchio Classification with a single threshold

The simplest adaptation of Rocchio Classification to multi-label problem is simply taking the labels which centroids have similarity greater than a single threshold value times the maximum similarity as follows.

$$C^* = \{ \ c \ | \ \overrightarrow{\mu_c} \cdot \overrightarrow{v_d} \ > \ th \cdot \overrightarrow{\mu_{c^*}} \cdot \overrightarrow{v_d} \ \} \tag{9}$$

where $C^*$ is the predicted label set of test document, $\overrightarrow{v_d}$ is the test document vector, $th$ is the single threshold value, and $c^*$ is the most probable label from (8).

Though this method is extremely fast, it ignores effects of label combination and it treats all categories in the same way. These drawbacks may degrade the classification performance of this method.

### 2.4 Rocchio Classification with Label Power-set (LP)

Label Power-set (LP) is a simple but effective problem transformation method from multi-label classification problem to single-label one. It considers each unique set of labels that exists in a multi-label training set as one of the classes of a new single-label classification task [7].

To apply Rochhio Classification to multi-label problem with Label Power-set, a centroid can be simply computed on each unique set of labels and the predicted labels of test document can be the label set which centroid has the largest cosine similarity with the test document. But this method cannot be scalable for large-scale data because the number of distinct label sets is too large, and the relatively few training examples per label set may degrade the classification performance.

### 2.5 Multi-Stage Rocchio Classification (MSRC)

We propose Multi-stage Rocchio Classification (MSRC) for efficiently adapting Rocchio Classification to multi-label problems. At first, we use a greedy algorithm to search for a predicted label set of a given test document through several stages. Then, we apply the termination condition at the end of each stage to predict the definite label set.

- *Greedy search algorithm for the predicted label set*

Instead of the whole label power-set, we search for the predicted labels from labels top-ranked in similarity one at a time. At the first stage, we take the first one which centroid has the largest similarity with test document. Next, we search the top K labels for the next one given the previous predicted labels. Until the termination condition is satisfied, this process is repeated through several stages as follows.

$$c_{n+1} = \underset{c^* \in C_{topK} - \{c_n, \cdots, c_1\}}{argmax} \vec{v}_d \cdot \vec{\mu}(c^*, c_n, \cdots, c_1) \tag{10}$$

where $c_{n+1}$ is the predicted label at the $(n+1)$th stage, $c_n, \ldots, c_1$ are the predicted labels at the previous stages, $\vec{\mu}(c^*, c_n, \ldots, c_1)$ is the centroid of the label set $(c^*, c_n, \ldots, c_1)$, and $C_{topK}$ is the set of K labels top-ranked in similarity. (K is the predefined parameter.)

- *Termination condition at the end of each stage*

To predict the definite label set instead of label ranking, searching process should finish if no more labels are needed at the end of each stage. To check whether more labels are needed or not, we compare similarities between test document and two centroids. The *including centroid* is normalized sum of training document vectors that belong to categories $c_1, \ldots, c_n$. It doesn't matter whether they belong to any other categories or not. The *same centroid* is normalized sum of training document vectors that belong only to categories $c_1, \ldots, c_n$. They should not belong to any other categories. Because the number of training examples for the *including centroid* is always greater than for the *same centroid*, there should be some penalty term for similarity with the *including centroid.*

$$th_n \vec{v}_d \cdot \vec{\mu}_{including}(c_1, \cdots, c_n) < \vec{v}_d \cdot \vec{\mu}_{same}(c_1, \cdots, c_n) \tag{11}$$

where $th_n$ ($< 1$) is the threshold parameter at the $n$th stage, $\vec{\mu}_{including}(c_1, \ldots, c_n)$ is the *including centroid,* and $\vec{\mu}_{same}(c_1, \ldots, c_n)$ is the *same centroid.*

The use of a centroid for a label set instead of a single label can improve the classification performance. However, the larger label set we use a centroid for, the fewer number of training examples there are to compute the centroid. If the number of stages S should be large ($> 10$) due to the large number of labels per document, this method will not be good at performance as well as scalability in this dataset.

## 2.6 Computational complexity of Multi-Stage Rocchio Classification

Our method is divided into two parts: Computing centroids and selecting most-probable label set through several stages.

Computing a set of centroids which represent labels is $O(C_{avg}DL_{avg})$. Time complexity of adding training documents belong to each category is $O(CD_{avg}L_{avg})$ where $C$ is the number of categories, $D_{avg}$ is the average number of training documents belong to each category, $L_{avg}$ is the average number of words per document since we compute only non-zero entries. Total time complexity of cosine normalization of the centroids is also $O(CD_{avg}L_{avg})$ since the average number of non-zero entries of a centroid is $O(D_{avg}L_{avg})$. Overall, time complexity of this phase is $O(C_{avg}DL_{avg})$ since $C_{avg}D = CD_{avg}$ where $C_{avg}$ is the average number of labels per document (Label Cardinality), $D$ is the number of training documents.

Before entering stages, picking K labels top-ranked in its centroid's similarity with a test document is $O(KCL_{avg})$. In $n$ th stage, computing centroids of $\vec{\mu}(c_1^*, c_n, \ldots, c_1) \sim \vec{\mu}(c_K^*, c_n, \ldots, c_1)$ is $O\big(KD_{avg}(C + L_{avg})\big)$ since searching for training documents belong to $c^*, c_n, \ldots, c_1$ is $O(CD_{avg})$, and the number of these documents is less than $D_{avg}$. Dot products between a test document and the centroids are $O(KL_{avg})$. Checking termination condition in each stage is $O\big(D_{avg}(C + L_{avg})\big)$ since computing *same centroid* and *including centroid* is $O\big(D_{avg}(C + L_{avg})\big)$, and dot products are $O(L_{avg})$. Overall, time complexity of all stages is $O\big(SKD_{avg}(C + L_{avg}) + KCL_{avg}\big)$ where S is the total number of stages.

Eventually, total time complexity of our method is linear in the size of the training documents $D$ and in the number of categories $C$. However, it is not $O(CD)$ but $O(C_{avg}D + C)$. This fact makes our method scalable for large-scale multi-labeled dataset because $C$ and $D$ are also very large in the dataset but $C_{avg}$ is usually much smaller than $C$.

## 3 Experiments

The third Pascal Challenge on Large-Scale Hierarchical Text Classification (LSHTC3) provides a great platform for us to test our method [3]. We applied our method to Wiki Medium and Wiki Large data sets and got online feedback from the oracle server.

Table 1 shows some statistics of Wiki Medium and Wiki Large datasets. $C_{avg}$, $D_{avg}$, $L_{avg}$ are very small and not changed much as the size of the training documents. This makes our approach efficient especially for these datasets.

**Table 1.** some statistics of data sets

| *Data set* | *# Train docs(D)* | *# Test docs* | *# Categories(C)* | *# Words* | $C_{avg}$ | $D_{avg}$ | $L_{avg}$ |
|---|---|---|---|---|---|---|---|
| Wiki Medium | 456,886 | 81,262 | 36,504 | 346,299 | 1.84 | 23.09 | 47.05 |
| Wiki Large | 2,365,436 | 452,167 | 325,056 | 1,617,899 | 3.26 | 23.74 | 42.53 |

Experiments were carried out with Intel® Core$^{TM}$ i5-2500 CPU @ 3.30GHz and 16 GB memory. The numbers of stages $S$ is 4, the number of labels K top-ranked in similarity for greedy search is 8, and the threshold of termination condition $th_n$ is 0.8 for all stages. These values are optimal according to our pilot experiments for a validation subset of the datasets. Though large S and K improve the classification performance a little bit, time costs also increase too much.

Table 2 shows the performance results for Wiki Medium dataset. "Acc" means accuracy. "LBMiF" means Label based micro F1-measure. "HF" means Hierarchical F1-measure [5]. "Best in LSHTC3" means the best results of each evaluation measure in LSHTC3. "Arthur in LSHTC2" means the results of "arthur" who is one of the best participants in LSHTC2 [4]. In comparison to Rocchio Classification with a single threshold ("Single threshold RC"), our adaption of Rocchio Classification to multi-label problems has a definite effect. Moreover, our method outperforms the best method in LSHTC2 and achieves a 3x speed-up. But our results are worse than the best results in LSHTC3.

Table 3 shows the performance results for Wiki Large dataset. Our results are comparable to the best results of each evaluation measure in LSHTC3. Especially our result of label based micro F1-measure is the best in LSHTC2 and LSHTC3. And our method outperforms the best method in LSHTC2 and, furthermore, achieves a 4x speed-up. Our method is more suitable for larger dataset as can be shown in these results. Because of the fewer number of training documents to make a centroid for a larger label set, the larger size of training examples can always improves the classification performance.

**Table 2.** Experimental results for Wiki Medium dataset

| *Algorithm* | *Acc* | *LBMiF* | *HF* | *Time per doc* | *Total time* |
|---|---|---|---|---|---|
| Single threshold RC | 0.3666 | 0.3885 | 0.6640 | 0.0040 sec | 321 sec |
| Our method(MSRC) | 0.3974 | 0.4326 | 0.6783 | 0.0093 sec | 752 sec |
| Best in LSHTC3 | **0.4382** | **0.4939** | **0.7092** | - | - |
| Arthur in LSHTC2 | 0.3739 | 0.3904 | - | 0.026 sec | 2160 sec |
| kNN Baseline | 0.2491 | 0.2979 | 0.5609 | - | - |

**Table 3.** Experimental results for Wiki Large dataset

| *Algorithm* | *Acc* | *LBMiF* | *HF* | *Time per doc* | *Total time* |
|---|---|---|---|---|---|
| Single threshold RC | 0.3242 | 0.3175 | 0.5195 | 0.044 sec | 5.57 hrs |
| Our method(MSRC) | 0.3614 | **0.3698** | 0.5602 | 0.090 sec | 11.3 hrs |
| Best in LSHTC3 | **0.3806** | 0.3452 | **0.5787** | - | - |
| Arthur in LSHTC2 | 0.3467 | 0.3484 | - | 0.37 sec | 46.8 hrs |
| kNN Baseline | 0.2724 | 0.3016 | 0.5193 | - | - |

Though we participated as “dhlee” in LSHTC3, our method in this paper is revised one. Unnecessary heuristics are removed, Tf-idf is replaced by BM25, and greedy search algorithm is added.

## 4 Conclusion

In this paper, we propose a Multi-Stage Rocchio Classification (MSRC) to solve large-scale multi-labeled hierarchical text classification problems. Although a relatively simple adaptation of Rocchio Classification to multi-label problem, eventually our method exhibits comparable performance to state-of-the-art methods and also high scalability especially in Wiki Large dataset. This was verified at LSHTC3.

## References

1. Lewis, D. D.; Yang, Y.; Rose, T.; and Li, F. RCV1: A New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research, 5:361-397, (2004)
2. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. (2008)
3. Third Pascal Challenge on Large Scale Hierarchical Text classification (LSHTC3). ECML/PKDD Discovery Challenge (2012) : http://lshtc.iit.demokritos.gr/

4. Xiao-lin Wang, Hai Zhao, and Bao-liang Lu: Enhance K-Nearest Neighbour Algorithm for Large-scale Multi-labeled Hierarchical Classification. Second Pascal Challenge on Large Scale Hierarchical Text classification (2011) : http://lshtc.iit.demokritos.gr/LSHTC2_CFP
5. Kiritchenko, S.: Hierarchical text categorization and its application to bio-informatics. Ph.D. thesis, University of Ottawa Ottawa, Ont., Canada. (2006)
6. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: An ensemble method for multilabel classification. In: Proceedings of the 18th European Conference on Machine Learning (ECML2007), Warsaw, Poland (2007) 406–417
7. Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas: Mining multi-label data. In Oded Maimon and Lior Rokach, editors, Data Mining and Knowledge Discovery Handbook, chapter 34, pages 667.685. Springer, 2nd edition. (2010)