Large Scale Hierarchical Text Classification

Kushal Yarlagadda, Massand Sagar Sunil
Advised by : Prof. Amitabha Mukerjee

Abstract

Large Scale Hierarchical Text Classification methods using the hierarchy have largely been unsuccessful. Most of the methods have used flat classification methods for the same. Our work primarily focusses on using k-nearest neighbour class of algorithms for this task. We use a modified implementation of the algorithm proposed by [1] to do the classification.

Introduction

Today, there is a huge dependence on online sources of informations such as Wikipedia, Quora etc. However, the assignment of categories to documents on Wikipedia, as well as the assignment of question tags on Quora remains a largely manual process. The Large Scale Hierarchical Text Classification (LSHTC [2]) problem is an attempt at automation of this process.

A real-life application of the LSHTC problem can involve assigning one or more categories from amongst order of 10^5 categories to a specific document. Given that we have such a large number of categories to choose from, the training set would also be large, typically of the order of 10^6. Given the large number of categories as well as the large training set, the running time becomes almost as important a consideration as the accuracy.

Related Work

Large Scale Hierarchical Text Classification Challenges are held at [2] to accelerate research in this area. Papers on methods and techniques used by top-performers in the previous competitions were published. The current competition is hosted on *Kaggle* [3]. [4], [1], [5] are papers on different optimizations k-NN algorithm to classify the test documents(flat-classification strategies). Other approaches could be top-down approaches where classification of the test document is done at each internal node using an SVM(like in [6]) .

Problem Instance

The dataset available at [3] consists of 2.3 million documents for training.
Each document is of the form -

$$label_1, label_2, label_3 \ldots, label_c\ feature_1 : value_1\ feature_2 : value_2\ feature_n : value_n$$

Basically, each document is preprocessed and given to us in a vector form. The categories assigned to the document are $label_1$, $label_2$, $\ldots$, $label_c$. The features in the feature vector of the document are $feature_1$, $feature_2$, $\ldots$, $feature_n$. The values associated with each feature is the term frequency(TF) of the word/feature in the document.

It can be seen that the classification is multi-label and multi-class.

Preliminaries

TF-IDF

- Term Frequency(TF) is of a term/feature w for a document d is the frequency of the term/feature in the document.
- Inverse Document frequency(IDF) is an inverse measure a term/feature's redundancy.

$$IDF(w) = \log \frac{\text{Total documents}}{\text{No. of documents containing word } w}$$

  Lower the IDF value, higher the redundancy of the word(i.e. the word/feature is a common word like 'is','that' etc).

For a given feature w and document d,

$$TF - IDF(w, d) = TF(w, d) * IDF(w)$$

which gives a measure of the relevancy the feature w is to the document d.

BM25 - Similarity Measure

The BM25 similarity measure is used for measuring how similar a training document d_1 is to the test document d_2 [7].

$$BM25(d_1, d_2) = \sum_{w \in d_1 \cap d_2} TF_1(w, d_1) * TF_1(w, d_2) * IDF_1(w)$$

Basically, for each common feature we take into account the general importance of the feature(IDF_1), its frequency in both the documents($TF_1(w, d)$).

Also,

$$IDF_1(w) = \log \frac{N - N_0 + 0.5}{N_0 + 0.5}$$

where N is total number of documents and N_0 is number of documents containing feature w. Observe this only a slight modification of the IDF measure which we mentioned before.

$$TF_1(w, d) = \frac{(k_1 + 1)TF(w, d)}{k_1 + 1 - b + b\frac{|d|}{s}}$$

where, k_1, b training paramaters(to be tuned), $|d|$ is the length of the document d(total number of features with duplicity) and s is the average length of the training document. This idea is taken up from [1].

Observe that $TF_1(w, d)$ is directly related to $TF(w, d)$ measure for features corresponding to the same document.

Inverse Document Index

An inverse document index is a data-structure which stores the list of all the documents containing a a feature for each feature. This makes process of pruning the training set much more computationally efficient as we shall see.

Basic Work

Our work primarily consisted of two different k-nearest neighbour methods. We give the algorithms for the two methods below and give the results for both the methods:

- Basic K-nearest neighbours algorithm
- Enhanced K-nearest neighbours algorithm

Basic K-nearest neighbours algorithm [3]

- Find 3 features in the test instance with the highest Term Frequency-Inverse Document Frequency(TF-IDF) value.
- Construct a subset S of the training set which consists of atleast one of these three features.We do this using the inverse index data structure which stores the documents to which a specific feature belongs. This speeds up the operation considerably as we would now have, on average, ten thousand documents out of the two million originally present.
- Find five documents in the set S which are closest to the test document. The similarity measure used in this case is the following

$$similarity = \frac{N_1 + N_2 - 2C_f}{N_1 + N_2 - C_f}$$

where N_1 = Number of Features in Document 1(Training Document),
N_2 = Number of Features in Document 2(Test Document) and
C_f = Number of Common Features in the two documents

- Assign scores to each of the categories present as follows:

$$Score_i = Frequency_i$$

where $Score_i$ = Score assigned to i^{th} category,
$Frequency_i$ = Number of documents to which the i^{th} category belongs from amongst these five documents

- Output the top 3 categories in terms of the score.

Enhanced K-nearest neighbours algorithm [1]

Overview of the algorithm

- We first find the top 3 features in the test instance with the highest TF-IDF value.
- We construct a subset S of the training set which consists of atleast one of these three features using the inverse index data structure mentioned above.
- For this document set S, we assign scores to all the documents based on the BM25 similarity measure as mentioned above.We use a validation set to obtain the parameters in the BM25 similarity measure.

- We then choose the top k documents from these on the basis of their scores. k is a parameter which we obtain by the use of a validation set.
- We then assign scores to each of the categories associated with these k documents as follows:[2]

$$Score(D) = \sum_{i=1}^{k} \gamma(S_b, D) * BM25(S_a, S_b)^{\alpha}$$

  where D is the category to which the score is being assigned, $\gamma(S_b, D)$ denotes whether D is a label of S_b, where S_b is the training document and S_a is the test document and $BM25(S_a, S_b)$ denotes the BM25 similarity and α is a parameter which is on the validation set.
- We then use a DS-cut [1] thresholding strategy to obtain the final labels for the document set.

DS-cut thresholding strategy [1]

In a Distinctive Cut Thresholding strategy(DS-cut), we first divide the scores of all the labels by the maximum score of a label. Then, we set the threshold 1 for the 1^{st} category(thus always including the first category). We then tune the thresholds for subsequent categories using a validation set.

Validation

We mention all through the previous algorithm that there are some parameters which need to be tuned. We have separated a small validation set of 3000 documents from the training set and tuned our parameters on this set.

Tuning parameter k

k is number of nearest neighbour documents we take into consideration. We tuned k in the range from 10 to 60.

Table 1. Tuning parameter k

Value of k	**Accuracy measure**
10	0.0775379
20	0.0775379
30	0.108278
40	0.12799
50	0.133764
60	0.133764

So, based on these value we set parameter k to 50.

Tuning parameter α

Parameter α is used to assign scores to the candidate categories. Since it appears in the exponent of the score the value will be small.

After tuning the parameter value of α was set to 0.75.

Table 2. Tuning parameter α

Value of α	Accuracy measure
0.50	0.155411
0.75	0.157766
1.00	0.155544
1.25	0.153619
1.50	0.141275
1.75	0.136831
2.00	0.128835

Tuning threshold parameters

These parameters decide which of the labels are assigned to the test document after the calculation of the scores using the DS-cut strategy. We used the following steps to tune these parameters - we first set an empirical lower bound on the value a threshold can take to 0.4.

We then tuned the model on various values of the thresholds and the maximum number of thresholds.

After tuning the maximum number of labels we take was set to 3 and the threshold values were set to $1, 0.6, 0.4$.

We assigned empirical values for parameters $k_1 = 6, b = 0.75$

Results

Accuracy

First we implemented the basic K-NN algorithm. The algorithm gave very poor results(has a lot of false positives). This may be because we treat all the k nearest documents as equals and so many categories have the same frequency when assigning final labels.

We tested the enhanced K-NN algorithm on some 10,000 test documents. This set of test documents is extracted from the training set. We then measured the cosine similarity between the final label vector predicted and the actual label vector. The actual formula for the measuring the similarity is -

$$A(l_p, l_a) = \frac{\sum_{l \in l_p \cup l_a} l_p(l) * l_a(l)}{\sqrt{|l_a| * |l_p|}}$$

where, $l_p(l) = 1$ if label l is present in label vector l_p(same with l_a). The average measure we got for the accuracy(measure of the cosine similarity) obtained was 18.88%.

Efficiency

With the use of the inverse document index data structure we were able to reduce the computation time per test document considerably. The average time taken per test document for the enhanced K-NN algorithm is $0.275s$.

Conclusions and Future Work

The use of hierarchy in the classification to achieve good accuracy is a major problem yet unsolved. The accuracy we feel can be improved a lot with a better tuning techniques for the parameter and using atleast some information from the hierarchy.

Acknowledgments

We would like to thank Prof. Amitabha Mukerjee for the emphasis he put on the project work in the course.

References

1. Wang Xl, Zhao H, Lu Bl (2011) Enhanced k-nearest neighbour algorithm for large-scale hierarchical multi-label classification. In: Proceedings of the Joint ECML/PKDD PASCAL Workshop on Large-Scale Hierarchical Classification, Athens, Greece. volume 5.

2. (2014). Large scale hierarchical text classification - pascal challenge. URL `http://lshtc.iit.demokritos.gr/`.

3. (2014). Large scale hierarchical text classification. URL `http://www.kaggle.com/c/lshtc`.

4. Han X, Liu J, Shen Z, Miao C (2011) An optimized k-nearest neighbor algorithm for large scale hierarchical text classification. In: Joint ECML/PKDD PASCAL Workshop on Large-Scale Hierarchical Classification. pp. 2–12.

5. Han X, Li S, Shen Z A k-nn method for large scale hierarchical text classification at lshtc3.

6. Liu TY, Yang Y, Wan H, Zeng HJ, Chen Z, et al. (2005) Support vector machines classification with a very large-scale taxonomy. ACM SIGKDD Explorations Newsletter 7: 36–43.

7. Robertson S, Walker S, Jones S, Hancock-Beaulieu M, Gatford M (1996) Okapi at trec-3. pp. 109–126.