An Optimized K-Nearest Neighbor Algorithm for Large Scale Hierarchical Text Classification

Xiaogang Han[1], Junfa Liu[1], Zhiqi Shen[2], and Chunyan Miao[1]

[1] School of Computer Engineering, Nanyang Technological University, Nanyang Ave, Singapore 639798
hanx0009@e.ntu.edu.sg
liujunfa@ict.ac.cn
ascymiao@ntu.edu.sg

[2] School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Ave, Singapore 639798
zqshen@ntu.edu.sg

Abstract. In this paper, we describe a large scale hierarchical text classification algorithm based on k nearest neighbor algorithm. Firstly, k-NN algorithm was performed to reduce the problem into a top-k examples based classification problem. Secondly, several critical category-neighbors features were identified and the weights of each of those features were estimated through cross-validation. Finally, the categories prediction algorithm utilizes the optimal parameters for the category-neighbors features to predict the categories for the testing documents. The experiments performed on three large datasets show that the classifier can gain high accuracy.

Keywords: hierarchical classification, k Nearest Neighbor, category-neighbors features

1 Introduction

Hierarchies are becoming even more important for the organization of text documents on the Web. There is a need for automated classification of new documents to the categories in the hierarchy to better search, recommend, and filter information for the user. Web hierarchies like Wikipedia [3] and ODP [4] provide categorization information for a wide range of topics. How to efficiently and accurately classify a new document onto categories in such hierarchies is a challenge for the data mining community.

In this paper, we study large scale hierarchical text classification problem based on Wikipedia and ODP data, and try to built a multiple stages model for hierarchical text classification. Our approach is motivated by the fact that the likelihood of categorizing a document into a category can be inferred from

[3] http://www.wikipedia.org
[4] http://www.dmoz.org/

the similarities between the document and each of the document in all the categories. Based on this assumption, an optimized k nearest neighbor classification algorithm is proposed. Firstly, k-NN algorithm was performed to reduce the problem into a top-k examples based classification problem. Secondly, several critical category-neighbors features were identified and the impact of each of those features were estimated through cross-validation. Finally, the categories prediction algorithm utilizes the optimal parameters for the category-neighbors features to predict the categories for the testing documents. The experimental results shows that our approaches gives promising results and can potentially be applied to various hierarchical text categorization tasks not limited to Web hierarchies.

The rest of the paper is structured as follows. Section 2 recaps the basic concepts and notations for our hierarchical classification algorithm. Section 3 develops our algorithm of example based classification. Section 4 presents the evaluation results. Section 5 discusses the related work. Finally, Section 6 concludes the paper and identifies possible future work.

2 Hierarchies and Classification

In this section, we defines the terms and notions that will be used in the description of the classification algorithm.

It is assumed that a hierarchy H is a collection of superordinate categories which is divided into collections of subordinate categories. Legal nodes for the classification task are only those leaf categories in the hierarchy which do not have any subordinate (child) categories. Each leaf category contains a collection of documents, which are always compiled by human experts. The structure of the hierarchy is shown in Figure 1.

Let $C = \{c_1, c_2, ..., c_m\}$ be the set of m leaf categories as the category space for the classification task. Let $D = \{d_1, d_2, ..., d_n\}$ be the collection of n documents in C, and let $T = \{t_1, t_2, ..., t_u\}$ be the set of u terms(words) appears in D. In our current approach, only the immediate superordinate categories of the leaf categories are used to infer the hierarchical information. We define these categories as the parent category space $P = \{p_1, p_2, ..., p_s\}$.

Based on these definitions, we introduce $Parents(c)$ to be the set of parent nodes of category c, as one category can belong to multiple parent categories. Let $Children(p)$ be the set of children categories of p. For instance, in Figure 1, $Parents(c_1) = \{p_1, \}$ and $Children(p_1) = \{c_1, c_2, c_3\}$. Furthermore, Let $D(c)$ be the set of documents in categories c, and let $T(d)$ be the set of terms in document d.

The hierarchical classification problem can be expressed as: given a new query document d, how to classify it into one or more of categories in C.

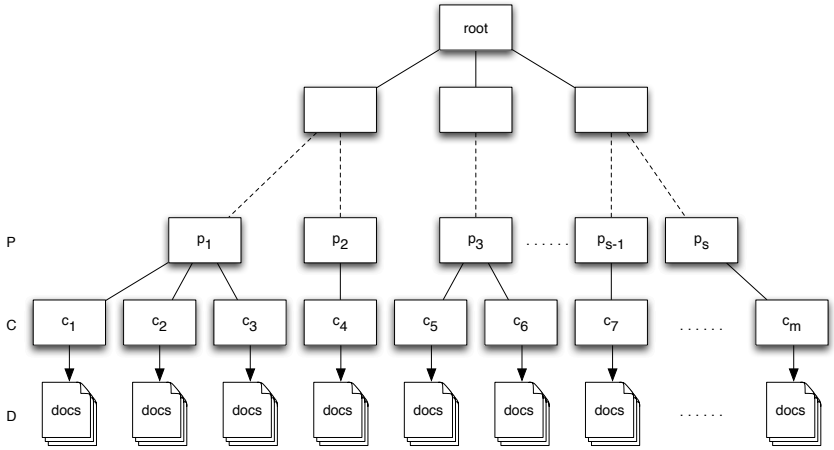

Fig. 1. Hierarchical Categorization Structure.

3 An Example Based Hierarchical Classification Algorithm

3.1 Overview

In text classification, k-NN is a method for classifying documents based on the majority vote of its most similar training examples (nearest neighbors). We introduce k-NN to calculate the top k most similar examples for each testing document. Rather than using majority voting or simple weighted voting method to predict the category label for the document, we perform detailed analysis on the critical features between the k nearest neighbors and each of the candidate categories and use cross-validation method to tune the weights for those features. Our algorithm is divided into two stages. First, the k nearest neighbor for each of the testing document is calculated. Second, by identifying the category-neighbors relationship features of the k neighbors, cross-validation is conducted followed by some further optimization. The overall algorithm workflow is shown in Fig. 2.

3.2 Calculating K-Nearest Neighbor

Preprocessing The first step is to prepare the training data and get the tuple set $DS = \{(d_i, c_j)|0 <= i < n, 0 <= j < m\}$ in which d_i is the term vector representation of text document and c_j is the category label.

Similarity Measure We use a variant of $TF \cdot IDF$ model [10] to represent the weighting of each term in a document, as this variant can improve the accuracy

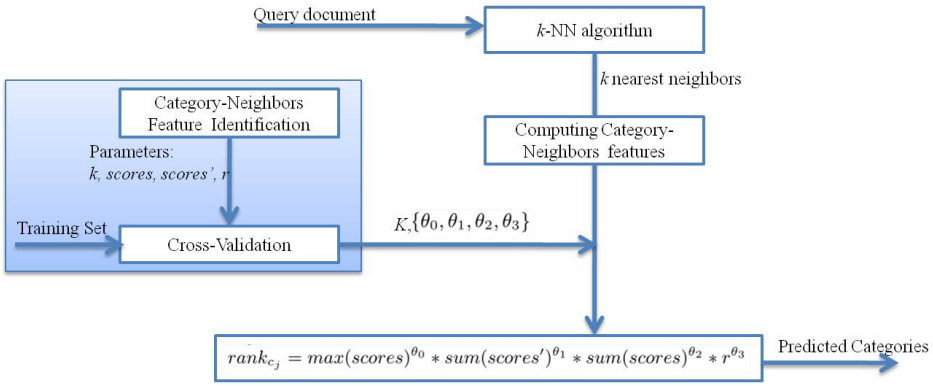

Fig. 2. Algorithm Workflow

significantly, compared to standard $TF \cdot IDF$ model. The $TF \cdot IDF$ variant is defined as:

$$\omega_{t,d} = log(tf_{t,d} + 1) \cdot idf_t \tag{1}$$

where $tf_{t,d}$ is the frequency count of term t in document d, n is the number of documents in the training corpus, and

$$idf_t = log(\frac{n}{n_t}) \tag{2}$$

is the *inverse document frequency* of term t, in which n_t is the number of documents containing the term.

The vector space model for calculating the similarity between the query document d and a training document d_i can be expressed as:

$$cossim(d_i, d) = \frac{d_i \cdot d}{\|d_i\| \cdot \|d\|} = \frac{\sum_{k=1}^{N} \omega_{k,i}\omega_k}{\sqrt{\sum_{k=1}^{N} \omega_{k,i}^2}\sqrt{\sum_{k=1}^{N} \omega_k^2}} \tag{3}$$

Calculating K-Nearest Neighbor The algorithm for calculating top k nearest neighbors in D for a query document d in the testing set is depicted in **Algorithm** 1.

3.3 Cross-Validation

A known drawback of *majority voting* for k-NN classification is that there might be bias in the category distribution, as the categories with more documents tend to come up in the k nearest neighbors. Moreover, for hierarchical classification, the hierarchical relationships between different categories exposed rich information on the categorization of a document. Therefore the distribution features of

Algorithm 1 Calculating K-Nearest Neighbor

Require: query document d
  for $d_i \in D$ **do**
    $score_i = cossim(d_i, d)$
  end for
  Compute tuple set $DS = \{(c_i, score_i)|0 <= i < k\}$ containing the category label for the k nearest neighbors and the corresponding scores
  return DS

the k neighbors on the hierarchical categories can be explored to increase the accuracy of the classification.

In this subsection, we will explore the category-neighbors distribution features together with their weights and build a cross-validation model to tune their weights with the objective of maximizing the classification accuracy.

Category-Neighbors Feature Selection Let $DS = \{(c_i, score_i)|0 <= i < k\}$ denote the set of tuples representing the the category labels for the k nearest neighbors and the corresponding similarity scores for testing document d. Let $C' = \{c_i|c_i \in DS\}$ denote the set of category labels in DS. Based on the analysis on DS, we identified the following critical features:

- the choice of k;
  The best choice of the number of nearest neighbors, k, depends on the training data. This is one of the most important factors for the validation.
- the similarity scores for each k nearest neighbors in each categories, $scores_{c_i}$.
  As each category has different number of nearest neighbors (with different similarity scores), it is vital to identify the contribution of these scores for the classification. Additionally, we introduce $max(scores_{c_j})$ and $sum(scores_{c_j})$ to denote the maximum and sum of the scores in $scores_{c_i}$.
- All training documents $D(c_i)$ in each category c_i
  It is observed that the total number of training documents in a category will affect the assignment of d into the category. The ratio $r = \frac{|scores_{c_i}|}{|D(c_i)|}$, which is expressed as dividing the number of nearest neighbors in c_i by the total number of documents in c_i, is a critical features for the classification.
- nearest neighbors belong to the parent category of C_i, denoted as $DS' = \{(p_j(c_i), scores_{p_j(c_i)})|0 <= j < s\}$;
  In the current approach, we only explore the immediate parent categories of the leaf categories in the hierarchy, as it is assumed that the higher the hierarchy level, the less specific the category, which means the less contribution to the classification accuracy.

Cross-validation To tune the parameters for all the features identified in the previous step, cross-validation is performed by dividing up the training data into sub-training set and validating set. As it is observed that the testing data was

sampled evenly on each category, we perform the same sampling on the training data. The resulting validating set consists of one randomly selected document from each category in the training data, and the rest of the documents are divided into sub-training set. The documents in the sub-training set along with the corresponding category labels are used to make predictions. The validation algorithm then iteratively makes predictions for each document in the validating set. The resulting answers, the predicted category labels in our case, are compared to the category label of the validating documents. The overall correctness ratio is identical to the accuracy of current parameter estimation.

Let D_v denote the collection of documents in validating set, and $D_{sub-train}$ denote the collection of documents in the sub-training set. The validating algorithm is described in Algorithm 2.

Algorithm 2 Cross-validation

INPUT: parameter list $\{\theta_0, \theta_1, \theta_2, \theta_3\}$
for $d_i \in D_v$ **do**
  c_{target}: the target category label for d_i
  $rank_{c_j}$: the score for category C_j
  for $c_j \in C'$ **do**
    $rank_{c_j} = max(scores_{c_j})^{\theta_0} * sum(scores_{p_j(c_i)})^{\theta_1} * sum(scores_{c_j})^{\theta_2} * r^{\theta_3}$
  end for
  $ranks_{c \in C'} = \{rank_{c_j} | 0 <= j < k\}$
  $c_i = argmax_{c \in DS}(ranks)$
  if $c_i == c_{target}$ **then**
    $matched+ = 1$
  end if
end for
$accuracy = matched/|D_v|$
return accuracy

Optimization For the practical application of the validating model, we further scale the scores for the categories and their parent categories as:

$scores'_{c_j} = \{log(1 + score) \mid score \in socres_{c_j}\}$,
$scores'_{p_j(c_i)} = log(sum(|children(p(c_i))|))$

and optimize the evaluation function as follows to gain maximum accuracy.

$$rank_{c_j} = max(scores'_{c_j})^{\theta_0} * sum(scores'_{p_j(c_i)})^{\theta_1} * sum(scores'_{c_j})^{\theta_2} * r^{\theta_3} \quad (4)$$

We perform the validation with a range of weights for the selected parameters. The weights with the highest accuracy are selected as the optimal weights. Finally, we popularize the weights on the evaluation function and apply it to C' to predict the target categories for each of the testing document.

3.4 Multiple Category Classification

As each document can be assigned to multiple categories in the hierarchy, it is required that the classification method can efficiently scale to multiple categories case. In our approach, as a ranked list of categories $ranks_{c \in C'}$ can be gained from the previous subsection, a straightforward solution is to select top-N categories as the predicted categories for the testing document d. The problem is how to decide N for each document d.

Let $avgCats$ denote the average number of categories per document for the hierarchy, which is a known constant value.

For the ranked list of categories $ranks_{c \in C'} = (rank_{c'_1}, rank_{c'_2}, ...rank_{c'_i}, ...)$ gained in the prediction procedure, rather than selecting c'_1, the category with the maximum rank score, we iterate the list and decide whether to include c'_i based on evaluating $rank_{c'_i}/rank_{c'_1} > \alpha, (0 < \alpha < 1)$. Then we calculate the predicted average number of categories per document, denoted as $avgCats_{predicted}$. By iteratively adjusting α to minimize $error = avgCats_{predicted} - avgCats$, the α with minimum $error$ is selected as the best threshold value for multiple category classification.

4 Evaluation and Results

4.1 Evaluation Metrics

The evaluation of the hierarchical text classification algorithm is performed on three categorization datasets, provided by the 2nd edition of the Large Scale Hierarchical Text Classification Pascal Challenge [5]. The first dataset, based on Dmoz, contains 27,875 categories, in which most documents belong to only one category and each category has only one parent category. The second and third datasets, both based on Wikipedia, contain 36,504 categories and 325,056 categories, in which a document may belong to multiple categories and each category can have multiple parent categories.

The metrics used for evaluating the classification algorithms include accuracy, example-based F-measure, label-based macro F-measure, label-based micro F-measure and multi-label graph-induced error [8].

4.2 Experimental Results

The experimental results for all of the three tasks compared to k-NN baseline algorithm with regards to various measures are shown in Table 1. In the table, task 1,2 and 3 are corresponding to Dmoz, Wikipedia small, and Wikipedia Large dataset respectively.

It can be observed that the proposed algorithm can obtain promising accuracy. In particular, the accuracy for Dmoz dataset is relatively higher than Wikipedia datasets, which may attribute to the fact that the classification for

[5] `http://lshtc.iit.demokritos.gr/`

Table 1. Evaluation results for three datasets

Task	Algorithm	Acc	EBF	EBP	EBR	LBMaF	LBMaP	LBMaR	LBMiF	LBMiP	LBMiR	MGIE
1	Our algorithm	0.3866	0.3871	0.3882	0.3866	0.2266	0.4081	0.2595	0.3867	0.3882	0.3852	2.8232
	k-NN baseline	0.1073	0.1075	0.1078	0.1073	0.0375	0.1633	0.0413	0.1074	0.1078	0.1070	4.2891
2	Our algorithm	0.3621	0.4217	0.4785	0.4362	0.2133	0.3974	0.2600	0.3756	0.3759	0.3752	4.3635
	k-NN baseline	0.2491	0.3175	0.2829	0.4163	0.1757	0.2522	0.2353	0.2978	0.2508	0.3665	5.7007
3	Our algorithm	0.3367	0.4116	0.4961	0.4157	0.1833	0.3866	0.2275	0.3345	0.3697	0.3055	4.3920
	k-NN baseline	0.2724	0.3471	0.3627	0.3869	0.1486	0.3033	0.1769	0.3015	0.3256	0.2808	4.2883

Acc = Accuracy
EBF = Example Based F1-measure
EBP = Example Based Precision
EBR = Example Based Recall
LBMaF = Label Based Macro F1-measure
LBMaP = Label Based Macro Precision
LBMaR = Label Based Macro Recall
LBMiF = Label Based Micro F1-measure
LBMiP = Label Based Micro Precision
LBMiR = Label Based Micro Recall
MGIE = Multi-label Graph Induced Error

Dmoz is a single category classification problem while the classification for Wikipedia datasets are multiple categories classification problems. On the other hand, the accuracy for Wikipedia small dataset is relatively higher than Wikipedia large dataset, which may attribute to the fact that the more the number of categories and number of documents, the harder to distinguish the feature space of each category. Another trend can be observed from the table is that KNN baseline performs better for Wikipedia datasets than Dmoz dataset, which probably due to the fact that the top categories obtained from KNN matches the accuracy metric for multiple categories classification better than single category case.

4.3 The impact of individual features on the accuracy

As described in Section 3.3, we observed that there are five critical distribution features affecting the classification accuracy. They are (1) the number of nearest neighbors, k; (2) the maximum score of the nearest neighbors in category c_i, $max(scores_{c_i})$; (3) the scaled aggregation of score of the nearest neighbors in category c_i, $sum(scores'_{c_i})$; (4) the scaled aggregation of score of the nearest neighbors in category $Parents(c_i)$, $sum(scores'_{p_j(c_i)})$; and (5) the ratio which divides the number of nearest neighbors in c_i by the total number of documents in c_i, r, respectively.

In this subsection, we analyze the impact of each of those features using the validation data to better understand the critical features for hierarchical text classification. To simplify the analysis, we will take Dmoz dataset as an example for the illustration.

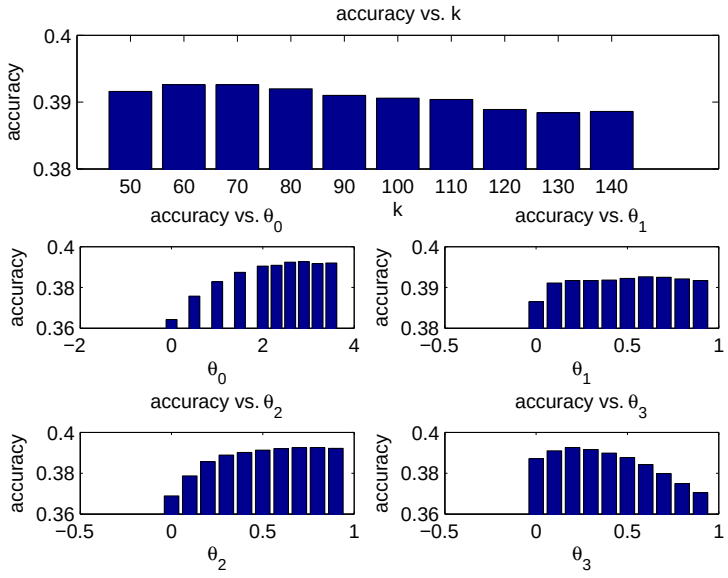

Fig. 3. The impact of individual features on the accuracy for Dmoz dataset

As shown in Figure 3, each of the features identified in our approach has different level of impact on the accuracy of the algorithm [6]. In particular, the best number of nearest neighbors, k, is 70, for Dmoz dataset. The figure also shows that $max(scores_{c_i})$ significantly affects the algorithm accuracy, as the accuracy changes rapidly as θ_0 grows.

4.4 Computation Time

The experimental environment for our evaluation is shown in Table 2.

Table 2. Experimental Environment

OS	Windows Server 2003
Memory	8 GB
CPU	Intel Xeon 2.67 Ghz × 4

The computation time for our algorithm is divided into training time, during which the k-NN algorithm runs; validating time, during which the parameter

[6] When estimating the impact of each features, the optimal parameters are set for the other features

estimation for each feature runs; and testing time, during which the prediction algorithm runs. The summary of the computation time for all of the three datasets are shown in Table 3.

Table 3. Computation Time

Task	Training	Validating	Testing
DMOZ	19763s	202s	119s
Wikipedia Small	6525s	551s	173s
Wikipedia Large	48411s	-	4536s

5 Related Work

Example based classification, especially k-NN algorithm is very popular for text classification [9] [3]. However, how to identify critical features in the training data to make better prediction is always a concern when applying k-NN to practical data, especially hierarchical classification problems. Selecting the features based on the data [5] [2] is very important for robust learning model. The hierarchical relationships [1] [7] between different categories in the hierarchical category space expose extra information about the categorization of new document [4]. However, how to utilize the semantic relationships between hierarchical categories is still an open problem.

One of the most important problem for k-NN algorithm is to compute the similarity between two documents. $TF \cdot IDF$ model [6] is the most widely used weighting scheme to represent feature vectors. Dependent on the specific problems, various variant of $TF \cdot IDF$ model are developed, such as the one described in [10], which is the weighting scheme used in our algorithm.

6 Conclusion and Future Work

6.1 Conclusion

In this paper, we proposed a two stages hierarchical text classification algorithm based on k-NN algorithm. In the first stage, k-NN algorithm was performed to reduce the problem into a top-k examples based classification problem. In the second stage, several critical category-neighbors features were extracted and the weights of each of those features were estimated through cross-validation. Finally, the categories prediction algorithm utilizes the optimal parameters for the category-neighbors features to predict the categories for the testing documents.

We found that to calculate the similarity between two feature vectors, the $TF \cdot IDF$ variant developed in [10] can gain better performance than the standard $TF \cdot IDF$ model.

We also found that for hierarchical classification problem, the hierarchical relationships expose extra information for the categorization of new document. Integrating such information together with category-neighbors features identified from the feature space can improve the classification performance significantly.

6.2 Future Work

There are still lots of work can be done with the proposed classification model. In the current model, we only utilized two levels of hierarchical relationships information. It is possible to identify the impact of more higher level hierarchical relationships for the classification. Furthermore, a more general ensemble methods can be developed to scale different features into the same space. Finally, the current dataset only provides single terms (unigrams) based features. The classification performance can be improved if word sequences (n-grams) information is available.

References

1. Dumais, S., Chen, H.: Hierarchical classification of web content. In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 256–263 (2000)
2. Forman, G.: An extensive empirical study of feature selection metrics for text classification. The Journal of Machine Learning Research 3, 1289–1305 (2003)
3. Larkey, L., Croft, W.: Combining classifiers in text categorization. In: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 289–297 (1996)
4. Mladenic, D., Grobelnik, M.: Feature selection for classification based on text hierarchy. In: Proceedings of the Workshop on Learning from Text and the Web (1998)
5. Rogati, M., Yang, Y.: High-performing feature selection for text classification. In: Proceedings of the eleventh international conference on Information and knowledge management. pp. 659–661 (2002)
6. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval* 1. Information processing & management 24(5), 513–523 (1988)
7. Sun, A., Lim, E.: Hierarchical text classification and evaluation. In: Proceedings IEEE International Conference on Data Mining. pp. 521–528 (2001)
8. Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multi-label classification. IEEE Transactions on Knowledge and Data Engineering (2010)
9. Yang, Y., Chute, C.: An example-based mapping method for text categorization and retrieval. ACM Transactions on Information Systems 12(3), 252–277 (1994)
10. Yang, Y., Liu, X.: A re-examination of text categorization methods. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 42–49 (1999)