International Journal of Computational Intelligence and Applications

A SCALABLE META-CLASSIFIER COMBINING SEARCH AND CLASSIFICATION TECHNIQUES FOR MULTI-LEVEL TEXT CATEGORIZATION

NANDITA TRIPATHI
Department of Computing, Engineering and Technology,
University of Sunderland,
St Peters Way, Sunderland, SR6 0DD,United Kingdom
(Nandita.Tripathi@research.sunderland.ac.uk)

MICHAEL OAKES
Research Institute of Information and Language Processing
University of Wolverhampton
MC Building, Stafford Street, Wolverhampton WV1 1LY, UK
(michael.oakes@wlv.ac.uk)

STEFAN WERMTER
Institute for Knowledge Technology, Department of Computer Science,
University of Hamburg,
Vogt Koelln Str. 30, 22527 Hamburg, Germany
(wermter@informatik.uni-hamburg.de)

Received 10th September 2015

Abstract - Nowadays, documents are increasingly associated with multi-level category hierarchies rather than a flat category scheme. As the volume and diversity of documents grow, so do the size and complexity of the corresponding category hierarchies. To be able to access such hierarchically classified documents in real time, we need fast automatic methods to navigate these hierarchies. Today's data domains are also very different from each other, such as medicine and politics. These distinct domains can be handled by different classifiers. A document representation system which incorporates the inherent category structure of the data should also add useful semantic content to the data vectors and thus lead to better separability of classes. In this paper we present a scalable meta-classifier to tackle today's problem of multi-level data classification in the presence of large datasets. To speed up the classification process, we use a search-based method to detect the level 1 category of a test document. For this purpose we use a category-hierarchy-based vector representation. We evaluate the meta-classifier by scaling to both longer documents as well as to a larger category set and show it to be robust in both cases. We test the architecture of our meta-classifier using six different base classifiers (Random Forest, C4.5, Multilayer Perceptron, Naïve Bayes, BayesNet and PART). We observe that even though there is a very small variation in the performance of different architectures, all of them perform much better than the corresponding single baseline classifiers. We conclude that there is substantial potential in this meta-classifier architecture, rather than the classifiers themselves, which successfully improves classification performance.

Keywords: Large scale datasets, Meta-classifiers, Multi-level classification, Text categorization, Parallel classifiers, Semantic representation of data.

1. Introduction

The easier access to online document collections has resulted in the overwhelming number of documents presently available along with a very wide variation in their content. To structure this content for easier accessibility, these documents are often arranged at multiple levels in a concept hierarchy. Hierarchies are not unique to the web. Documents collected for a specific purpose, e.g. collections of medical documents (MEDLINE), patent documents (WIPO) and news articles (RCV1) are all structured in a hierarchy. Similarly on the web, Yahoo! and the Open Directory Project (ODP) are two examples of systems which follow a structured document catalogue. The size and depth of data taxonomies is increasing with the current explosion of data. Taxonomies now consist of thousands of categories. An exhaustive study [1] found the Yahoo! directory to contain 292,216 categories in a 16-level hierarchy. This study was conducted on data collected in 2004.

Single classifiers do not take advantage of this hierarchical information. The hierarchy has to be flattened to a single level for the application of these classifiers. Flattening results in a huge number of categories which have to be differentiated by a single classifier. Single classifiers are not able to handle such a large number of categories. For example, the time complexity of an SVM is proportional to the number of categories [2]. This training time soon reaches unacceptable levels with the number of categories available in current systems. Furthermore, the information inherent in the hierarchy is lost during flattening and a single classifier is not able to focus on differences between categories at the lower level of a hierarchy. Therefore text classifiers which use this hierarchical information presented with the data are needed for further improvement in classification performance. This problem has been tackled in the literature with two different perspectives:Hierarchical Text Classification and Subspace Learning.

1.1. *Hierarchical Text Classification*

Several researchers [3], [4], [5], [6] have worked on hierarchical datasets extracted from various versions of the Reuters Corpus using different classifiers such as Naïve Bayes, SVM and Neural Networks. Other works include the application of Naïve Bayes on the UseNet and Yahoo datasets [7] and the use of SVM and kNN classifiers with the OHSUMED Corpus [8]. Liu et al [1] reported an evaluation of a hierarchy of SVMs on the complete Yahoo! taxonomy along with an analysis of the Yahoo! taxonomy itself. Ghazi et al [9] compared a flat Support Vector Machine (SVM) with a two-level and three-level hierarchy of SVMs for the classification of emotions in text using blog sentences and children's stories. All these studies concluded that the use of a structured topic hierarchy (even a partial one) along with dimensionality reduction resulted in a better classification performance than a flat category system.

Error Propagation and the Optimum Number of Levels: The use of many levels in hierarchical classification was actually found to degrade classification performances due to the effect of error propagation. A few researchers have proposed implementations of a single SVM multiclass classifier to deal with hierarchical information [10], [11]. However these studies were conducted on very small datasets with few levels of hierarchy (WIPO-alpha collection). This method seems inappropriate for scaling to very large datasets with a large number of categories and many hierarchy levels. A method for error reduction and correction has also been proposed for a hierarchical arrangement of classifiers [12]. This method requires a lot of prior information and computations at each node which would affect training times adversely.

Overall, hierarchical text classification research suggests that the popular divide-and-conquer strategy with the use of successive classifiers at different levels along with feature reduction is best suited for scaling up to a large number of documents as well as to a large number of categories. Training time is also significantly reduced by this method. To reduce the effect of error propagation, a small number of levels (two or three) should be used. The given hierarchy structure can be optimized by removing some intermediate levels between the root and the leaf nodes to create a two or three-level category hierarchy.

1.2. *Subspace Learning*

As the volume and diversity of data increases, the number of dimensions required to represent the data also increases drastically. Such high dimensions adversely affect classifier performances. Subspace Learning is a technique used in many fields to bring down the number of dimensions. Application areas of subspace learning include image processing, pattern recognition, computer vision, robotic vision, human gait analysis, object classification, document classification and multimedia classification to name a few. Several researchers have also applied subspace learning to the text domain [13], [14], [15], [16], [17]. Research in subspace learning is broadly divided into two main areas – *Feature subspace learning* which focuses on finding a reduced set of dimensions to represent the entire dataset and *Data subspace learning* which tries to find an optimal data subspace along with features corresponding to that subspace to improve overall classification performance. Feature reduction on the full data space is still an ongoing area of research.

In the multi-level text domain, we need to differentiate between similar subcategories within a larger category. As such, we need to focus more on smaller differences which would not be possible with a reduced feature set on the complete data. Hence data subspace learning is more suited to multilevel text categorization. However, *data subspace learning* research mostly concentrates on subspace clustering (also called projected clustering). Subspace clustering tries to find clusters present in different subspaces of a dataset. It is therefore a combination of a *search method* (to find the subspace) and a *learning method* (to find the clusters within the located subspace). There

are large hierarchical datasets available with associated category information. This information should therefore be used for text classification.

The current state of subspace research indicates that classification is more popular with feature subspace learning while clustering is more popular with data subspace learning. The analysis of our problem domain, however, suggests classification to be more appropriate due to the presence of class labels. There is therefore a need to develop classification methods which work on data subspaces. While two-level hierarchical classification may seem analogous to this situation, a major difference is the use of a classifier at the top level to detect the first level of categories in hierarchical classification. Data subspace methods, on the other hand, use search techniques to detect subspaces. This is very useful for improving classification and retrieval speeds. Hence the need for a fast multilevel classification system points to a *search-based* method to detect the subspace (first level category) followed by *classification* within the subspace with reduced dimensions to detect the second level of categories.

Partitioning of Vector Feature Space:

For dimension reduction, a partitioning of the feature space which corresponds to a partitioning of the underlying data is logically required. Tulyakov et al [18] have suggested that the ideal method would be to partition the feature space into regions related to different categories. This suggests that category information should be incorporated into feature vectors. Thus we have to look beyond the standard tf-idf [19] vectors and even beyond simple semantic enhancements such as grouping similar words based on some dictionary or thesaurus. A category-based vector system would further be useful to accommodate the inherent category structure of the data and thus add useful semantic content to the vector representation. Positioning similar categories close together in the feature space can lead to a spatial representation of the category hierarchy within the feature vector. This would enable different types of partitioning to access different levels of information.

2. Meta-classifier Framework for Two-Level Text Categorization

Section 1.1 has suggested that the optimum number of levels in a modified hierarchy should be two or three only. In this work, we concentrate on two-level text categorization as our data hierarchy could easily be reduced to two levels as explained later in section 3.1. However, this framework is also extensible to more than two levels.

2.1. *Two-Level Vector Representation*

Our meta-classifier will use a special vector representation called the Conditional Significance Vector [20] which represents a two-level category hierarchy within a single vector. This vector consists of ***(M+N)*** components out of which the first ***M*** represent the ***M*** level 1 (main) topics and the remaining ***N*** represent the ***N*** level 2 (sub) topics. The ***M*** level 1 topics can be considered as representing ***M*** subspaces of the full data space.

Within the ***N*** level 2 topics, the subtopics belonging to the same main topic are positioned consecutively in the vector space. This leads to a semantic division of the vector space into ***M*** groups, each group representing the subtopics of a specific main topic and therefore a *subspace*. Since the document significance vector represents the significance of the document for the different categories, the category with the maximum numerical significance value is the *most likely* to be the real category of a given document. Hence, the *Maximum Significance Value* is defined as a means to detect the relevant subspace (level 1 topic) of a new test document. The Conditional Significance Vector can also be recursively expanded for extending to more than two levels. The Full Significance Vector [20] which works with single level categories will be used as a baseline for comparison.

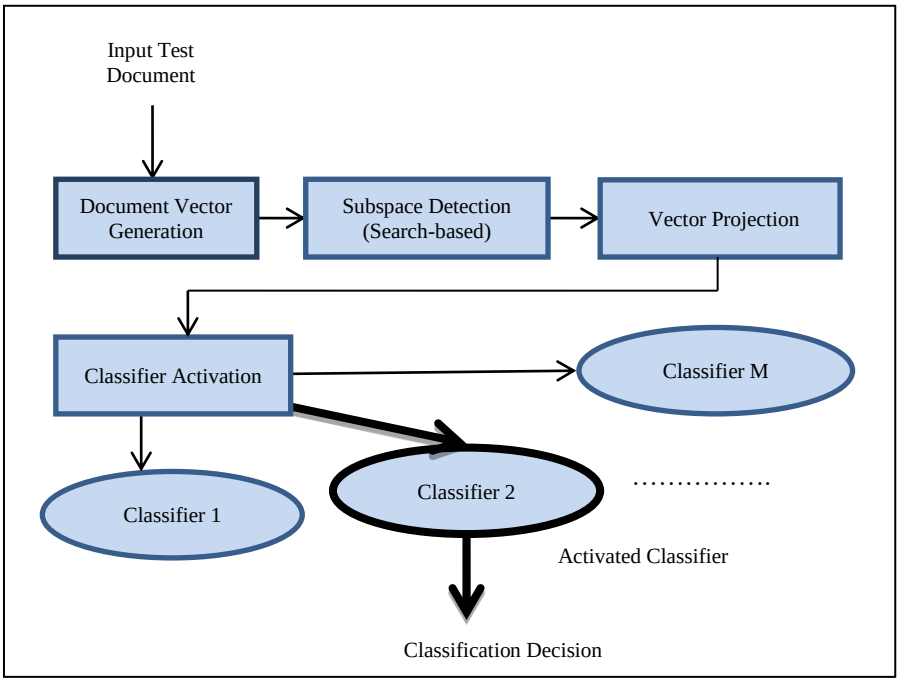

Fig. 1: Meta-Classifier Framework for Two-Level Text Categorization

2.2. *The Meta-Classifier Framework*

Fig. 1 shows the proposed meta-classifier framework for two-level text categorization. The last stage of the meta-classifier consists of ***M*** classifiers for the ***M*** level-1 topics or subspaces of the dataset. These ***M*** classifiers are different instances of a single type of classifier, i.e. the implementation will consist of either ***M*** Naïve Bayes classifiers or ***M*** MLP classifiers or ***M*** BayesNet classifiers etc. During the training phase, the Conditional Significance document vectors are generated for the training documents. This training data vector set is then divided into ***M*** separate training data subsets according to the ***M*** level-1 topics. The relevant feature vector subset is extracted for each subspace. These training data subsets with the relevant feature subsets and the associated document

subtopic (level-2) labels are then used to train the corresponding base classifiers associated with the different subspaces. Fig. 1 shows the path followed by a test document in this framework. Firstly the two-level Conditional Significance Vector is generated for the test document. The relevant subspace of a test vector is detected using the Maximum Significance Value. This method *searches* for the main topic having the maximum numerical value among the level-1 vector components. This is followed by the vector projection phase where only the vector components corresponding to subtopics of this subspace (main topic) are extracted. The classifier trained on this subspace is then activated for level-2 classification of the test vector. The predicted subtopic labels of the test vectors are then compared with their *actual* subtopic labels for the calculation of classification performance.

In this meta-classifier framework, each base classifier trains on less data with reduced dimensions. This is expected to reduce the training time of each classifier thus impacting the overall training time. Classification performance is also expected to improve as each base classifier deals with a smaller variation in data.

3. Experimental Methodology

3.1. *Experimental Dataset*

The datasets used for our experiments were the well-established Reuters RCV1 benchmark and the LSHTC dataset drawn from the Open Directory Project (ODP) for the ECIR 2010 challenge.

3.1.1. *Reuters Corpus (RCV1):*

The Reuters Corpus [21] is a well-known test bench for text classification experiments. We used the Reuters Corpus Volume 1 (RCV1) which is a collection of 806,791 news items written by Reuters journalists in 1996 - 1997. We extracted two datasets of 10,000 items each from the Reuters Corpus – Reuters Headlines which consist of one line of text with about 3 – 12 words and Reuters Full Text which consisted of headlines along with the news body text. The topic codes in the Reuters Corpus are organized into four hierarchical groups CCAT, ECAT, MACT and GCAT. In the Reuters original category hierarchy, the leaves are not all at the same level. The CCAT, ECAT and MCAT groups are complete till level 3 while the GCAT group is complete only till level 2. *This leads to the restriction of a two-level hierarchy for our system*. We removed the intermediate level 2 categories in CCAT, ECAT and MCAT connecting the level 3 categories (C11, E12, M14, etc) directly to the corresponding level 1 categories CCAT, ECAT and MCAT. The categorization below these levels was discarded. Fig. 2 shows both the original as well as the modified Reuters hierarchy used in this work.

As a representative test, ten thousand headlines along with their topic codes were extracted from the Reuters Corpus. These headlines were chosen so that there was no overlap at the first level classification. Each headline belonged to only one level 1 category.

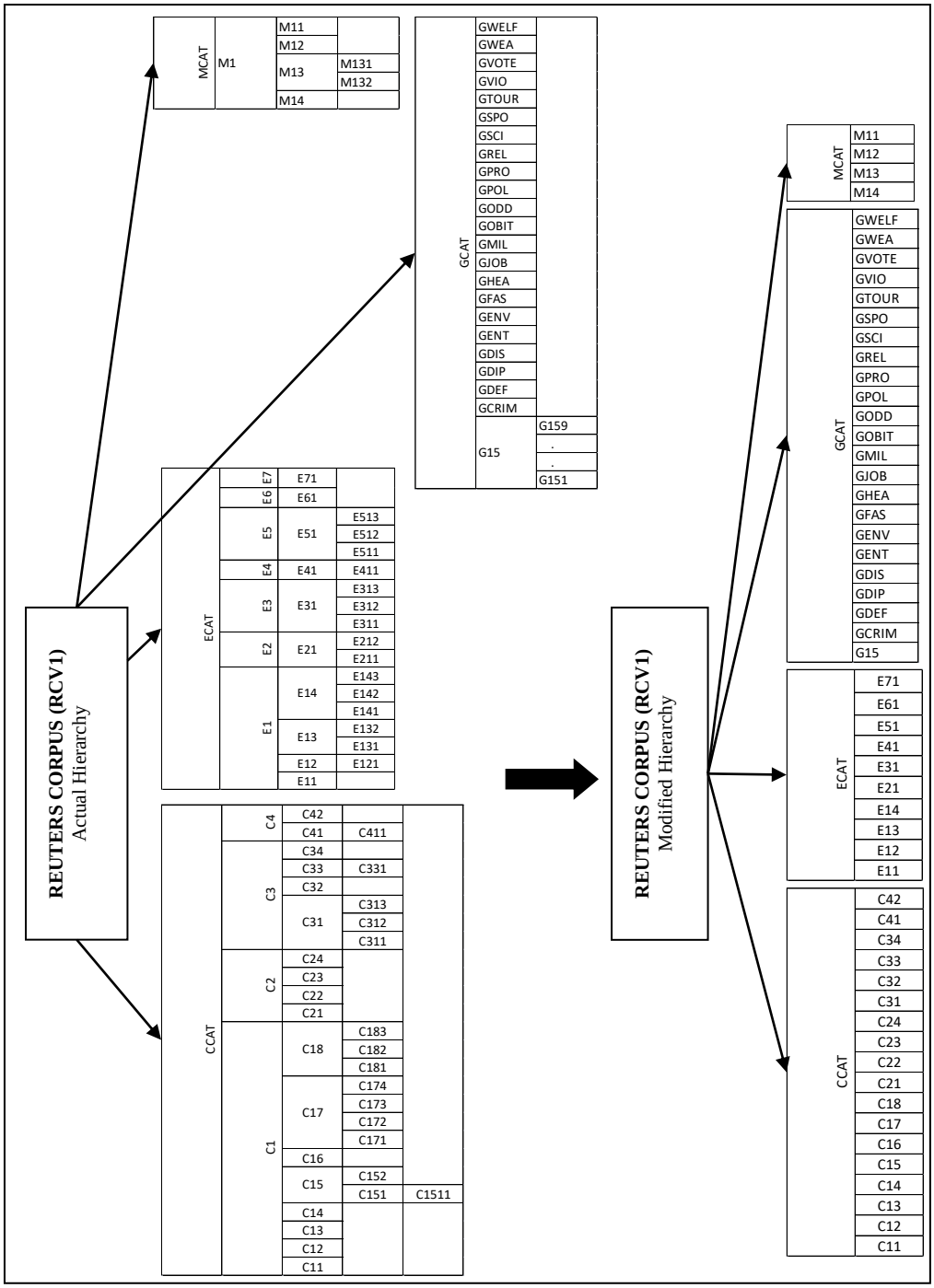

Fig 2: Reuters Corpus Hierarchy

Since most news items had multiple level 2 subtopic categorizations (as per the modified hierarchy), the first subtopic was taken as the assigned subtopic. Our assumption here was that the first subtopic used to tag a particular news item was the one most relevant to it. Thus each news item had two labels associated with it – the main topic (Level 1) label and the subtopic (Level 2) label. A total of 50 subtopics were used in our dataset. Pre-processing was done to separate hyphenated words to avoid such combinations being interpreted as new words rather than a sequence of known words. Dictionaries with term frequencies were generated using the TMG toolbox [22] and were then used to generate the Full Significance Document Vector and the Conditional Significance Document Vector representation for each document.

3.1.2. *LSHTC Corpus:*

To test the ability to scale-up to a larger hierarchy, we used the Large Scale Hierarchical Text Classification (LSHTC) [23] competition data from the LSHTC website (http://lshtc.iit.demokritos.gr) as our second corpus. This challenge was part of the European Conference on Information Retrieval (ECIR) 2010. The LSHTC data was constructed by crawling the web pages that are found in the Open Directory Project (ODP) located at www.dmoz.org and translating them into feature vectors. These vectors were called content vectors. Two datasets were put up for the LSHTC competition – the large_lshtc_dataset and a smaller dataset named as the dry-run_lshtc_dataset. The directory of each dataset consisted of a *cat_hier.txt* file describing the category hierarchy of the dataset and data folders for four tasks (Task1 – Task4). Task1 contained only crawl data while the data for task 2, task 3 and task 4 contained crawl data and RDF data. We used the data from the dry-run task1 training folder as our LSHTC corpus. There were 10 level 1 main topics and 158 level 2 subtopics in this dataset. As this provided a sufficient comparison with the Reuters Corpus, we did not take the lower level categories of this corpus into account. There were no overlapping topics at any level in this corpus. The content vectors of this dataset were then used to generate the Full Significance Document Vector and the Conditional Significance Document Vector for each document.

3.2. *The Experimental Environment*

The performances of different classifiers have been compared for single level classification by several researchers [24] [25]. However, to the best of our knowledge, an exhaustive study comparing the performance of various classifiers for two-level text classification has not been carried out. We propose a meta-classifier which can therefore be implemented with any classifier. The Waikato Environment for Knowledge Analysis or WEKA [26] is an open source machine learning environment containing a wide variety of classifiers. It was developed by the University of Waikato, New Zealand. We chose six different classifiers in WEKA to serve as base classifiers for the purpose of evaluating our framework. The classification algorithms were Random Forest, C4.5, Multilayer Perceptron, Naïve Bayes, BayesNet and PART. Random Forests [26] are a combination of tree predictors such that each tree depends on the values of a random

vector sampled independently. C4.5 [27] is an inductive tree algorithm with two pruning methods: subtree replacement and subtree raising. The Multilayer Perceptron [28] is a neural network which uses backpropagation for training. BayesNet [29] implements Bayes Network learning using various search algorithms and quality measures. A PART [30] decision list uses C4.5 decision trees to generate rules. Naive Bayes [31] is the simplest form of Bayesian network, in which all attributes are independent given the value of the class variable.

3.3. *Experimental Methodology*

We compare the performance of our meta-classifier framework with the corresponding single classifier as the baseline. After this, we evaluate the scaling performance of our framework under two different criteria – scaling up to longer documents with the same category set and scaling up to a much larger category set. For these experiments, we generated a number of datasets from the test corpora as follows.

3.3.1. *Dataset Generation*

Two different vector representations, the Full Significance Vector and the Conditional Significance Vector, were generated for our data. The Conditional Significance Vector dataset set was then processed further to generate multiple subspace-based training sets.

a) Full Significance Vector

The document Full Significance Vector consists of $M+N$ vector components where the first M vector components represent the level 1 main topics. The first M vector components are deleted leaving a vector with N components representing the N subtopics (N=50 for Reuters and N=158 for LSHTC). The order of the data vectors was then randomised and divided into two sets – training set and test set. The Reuters datasets were divided into 9000 training and 1000 test vectors while the LSHTC dataset was divided into 4000 training and 463 test vectors. We use the Full Significance Vector as a baseline for our experiments as it shows a superior performance to the popular tf-idf vectors [20].

b) Projected Conditional Significance Vectors for Multiple Classifiers

Here, the order of the Conditional Significance document vectors was randomised and divided into two sets – training set and test set (9000 Training/1000 Test for Reuters and 4000 Training/463 Test for LSHTC). The training set was then divided into M sets (4 for Reuters and 10 for LSHTC) according to the main topic labels. For each of these sets, only the relevant subtopic vector entries (e.g. ***C11, C12***, etc for ***CCAT*** in Reuters; ***A01, A02***, etc for ***A*** in LSHTC) for each main topic were retained. Thus in the Reuters Corpus, the CCAT category training dataset had 18 columns for 18 subtopics of CCAT. Similarly the ECAT training dataset had 8 columns, the GCAT training dataset had 20 columns and the MCAT training dataset had 4 columns. These 4 training sets were then used to train the 4 parallel classifiers of the Reuters meta-classifier. In the LSHTC dataset, the main

category ***A*** training dataset had 19 columns, the main category ***B*** training dataset had 36 columns and so on. The 10 training sets were then used to train the 10 parallel classifiers of the LSHTC meta-classifier.

The main category of a test data vector was determined by searching for the maximum significance vector entry in the first ***M*** columns representing the ***M*** main categories. After this, the entries corresponding to the subtopics of this predicted main topic were extracted along with the *actual* subtopic label and given to the classifier trained for this predicted main category. Hence this meta-classifier combines a *search* at the first level followed by *classification* at the second level to determine the subtopic of a test document

3.3.2. *Performance Evaluation Metrics*

In this work, we use Classification Accuracy (the proportion of correctly classified instances to the total number of instances) which is a popular performance evaluation metric for multiclass classification setting such as ours. It has been used by several researchers [3], [7] instead of precision/recall/F-measure which are inherently binary classification measures.

As datasets become larger and larger and involve thousands of categories, the training and classification (test) times also increase. This affects the real-time performance of a classifier. Thus, we also measure the training times and the classification or test times to study the time efficiency of our framework. In order to compare the performance of the framework on two different datasets, we use what we call *improvement measures* rather than absolute metric values. We compare the improvement obtained in classification accuracy and training/test times over the corresponding baselines for the two datasets. We call these *Classification Accuracy Improvement* and *Training/Test Time Speedup.* We also discuss the reasons for these improvements in the results section.

We conducted experiments in Weka with six different classification algorithms as base classifiers in the meta-classifier framework. Single classifiers using the Full Significance Vector on the complete dataset were used as baselines for these experiments. Classification Accuracy, Training Time and Test Time were recorded for each experimental run. The average of ten runs with different classifier parameter values for was taken for comparing the classifier performances.

4. Results and Analysis

4.1. *Evaluation of the Meta-Classifier Framework on Basic Metrics*

In the first step, we evaluated our meta-classifier framework on the three basic metrics of Classification Accuracy, Training Time and Test Time. The baseline used for comparison in each case was the corresponding single classifier using the Full Significance Vector format. Our results showed that the performance of the meta-classifier framework was better than the corresponding single classifier baselines for all the different base

classifiers for all the three datasets. Classification Accuracy was increased and training times as well as test times were reduced in all cases.

Table 1: Classification Accuracy %								
	Reuters Headlines			**Reuters Full Text**			**LSHTC**	
	Meta	**Single**		**Meta**	**Single**		**Meta**	**Single**
Naïve Bayes	92.10	85.9		69.60	62.8		80.13	67.17
BayesNet	92.30	71.4		67.30	55.3		77.32	44.28
C4.5	92.10	91.2		70.30	62.4		79.70	54.21
Random Forest	93.40	86.7		71.25	62.81		78.10	36.93
PART	92.00	89.2		69.30	65.6		79.91	60.26
MLP	93.97	79.53		72.28	47.01		82.92	21.02
t-test p-value	**0.038195**			**0.018636**			**0.006258**	
Results significant in all datasets (p-value less than 0.05)								

The results also showed that the classification accuracies achieved by the various meta-classifier implementations were quite similar to each other for a given dataset whereas there was a wide variation in the classification accuracies of the corresponding single classifiers used for the same dataset. The average classification accuracies for the meta-classifiers were 92.64% for Reuters Headlines, 70.01% for Reuters Full Text and 79.68% for LSHTC. This shows that the choice of base classifier does not matter much and the performance of our framework is ***classifier-independent***. Our results also showed that the improvement achieved over the baselines was different for the three datasets. The individual classification accuracy values for all cases are given in Table 1. These results are shown to be statistically significant by the t-test [32] p-values included in the table.

4.2. *Scaling Performance of the Meta-Classifier Framework with the Improvement Metrics*

For scaling performance, we compared the ***improvement metrics*** (Classification Accuracy Increase, Training Time Speedup and Test Time Speedup) for two different types of scale-up, i.e. scaling to longer documents as well as scaling to a larger taxonomy. In the case of scaling to longer documents, we compared the improvement metrics of the Reuters Headlines dataset with the Reuters Full Text dataset which has longer documents but the same number of categories (4 main and 50 subtopics). For scaling to a larger taxonomy, we compared the improvement metrics of Reuters Headlines dataset with those of the LSHTC dataset which has a much larger taxonomy (10 main and 158 subtopics).

Scaling to Longer Documents
(Reuters Headlines vs. Reuters Full Text)

Scaling to a Larger Taxonomy
(Reuters Headlines vs. LSHTC)

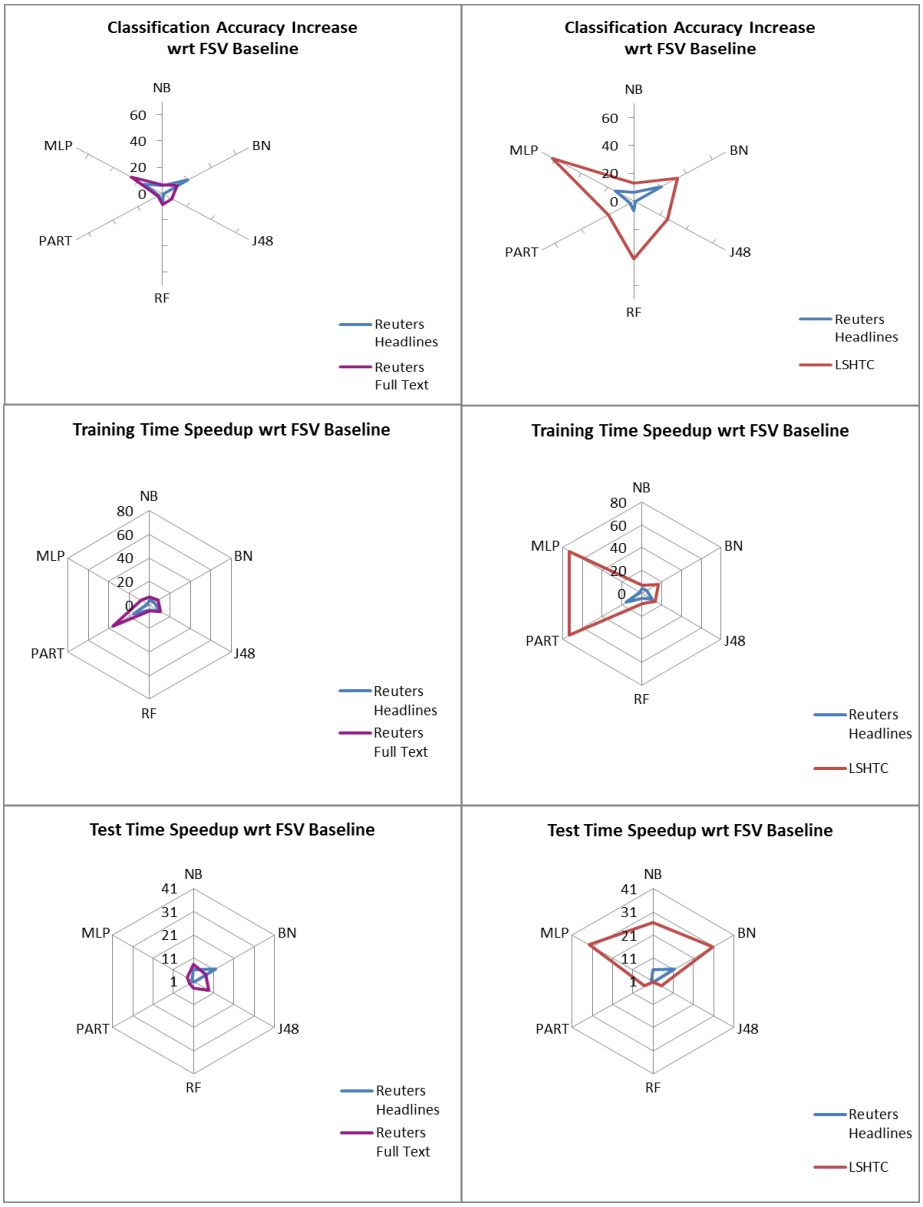

Fig. 3: Scaling Performance of the Meta-Classifier Framework

The charts in Fig. 3 show these improvement metrics for both types of scale-up. For scale-up to longer documents, the improvement observed with Reuters Full Text was similar to that of the Reuters Headlines. For the BayesNet (BN) - based meta-classifier

framework, the Reuters Headlines dataset showed more improvement in terms of Classification Accuracy Increase and Test Time Speedup while with the PART-based framework, Reuters Full Text showed a higher Training Time Speedup. Overall the improvement metric values are similar in both datasets and as such we can state that our method scales up well to processing longer documents.

The scale-up to a larger taxonomy showed a very significant difference. As can be seen in the charts in Fig. 3, the larger taxonomy (LSHTC) showed much more improvement than Reuters Headlines. All the three metrics, Classification Accuracy Increase, Training Time Speedup and Test Time Speedup are much greater for LSHTC than for Reuters Headlines. Hence the performance of our framework improves with increasing complexity of data.

For an analysis of this unexpected result (improved performance with increasing complexity of data), we look at the architecture of our meta-classifier framework for the two different taxonomies. For the Reuters taxonomy with 4 main topics, the framework architecture consists of 4 classifiers to deal with these 4 main topics or subspaces. In the case of the LSHTC taxonomy, there are 10 main topics and the framework architecture now has 10 classifiers to deal with these 10 main topics. Thus more classifiers are combined in a more complex hierarchy resulting in a good performance output. However, in the case of the corresponding baseline classifiers, the single classifier dealing
with a more complex hierarchy performs less well and as such the ***improvement*** observed with respect to the baselines is greater in the case of more complex data.

5. Conclusion

Our experimental results show that our high-level architecture is effective in improving learning at the subtopic level (level 2). This improvement is almost independent of the type of base classifier used. Even though there is variation in the performance of various multiple classifier architectures, they are very close to each other and all of them are much better than the baseline single classifiers over the full data space. This suggests that it is the general architecture of maximum significance-based subspace learning which improves the performance. For instance, an elementary classifier such as Naïve Bayes can be used with this architecture as can a more complex classifier such as the MLP. The strength of our architecture is based on the method of classifier combination rather than the classifiers themselves. Thus, it is a powerful meta-classifier whose performance is even more useful when the underlying data has a more complex category hierarchy. This is supported by a greater improvement observed over the baselines in the LSHTC Corpus with 10 main and 158 subtopics than in the Reuters Corpus with 4 main and 50 subtopics. A major implication of this result for the field of text classification and classifier theory is that further improvements in classification performance can be obtained by combining various classifiers and by focusing on improving the feature vector representation. Vector representations that incorporate semantic information produce better results than general vectors like TF-IDF. Thus feature engineering and classifier combinations will be our

focus of future work for improved classification performance. We also intend to explore the extension of this work for multi-label categorization.

6. References

[1] T.Y. Liu, Y. Yang, H. Wan, H.J. Zeng, Z. Chen and W.Y. Ma, Support Vector Machines Classification with A Very Large-scale Taxonomy, *SIGKDD Explorations*, vol. 7(1), pp. 36-43, 2005.

[2] T. Li, S. Zhu and M. Ogihara, Using discriminant analysis for multi-class classification: an experimental investigation, *Knowledge and Information Systems*, vol. 10(4), pp. 453-472, 2006.

[3] D. Koller and M. Sahami , Hierarchically classifying documents using very few words, in *Proceedings of the Fourteenth International Conference on Machine Learning (ICML-97)*, Nashville, Tennessee, pp. 170-178, 1997.

[4] A. S. Weigend, E. D. Weiner and J. O. Pedersen, Exploiting Hierarchy in Text Categorization, *Information Retrieval*, vol. 1(3), pp. 193-216, October 1999.

[5] F. Fukumoto and Y. Suzuki, Manipulating Large Corpora for Text Classification, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 196-203, Philadelphia, 2002.

[6] R. Wetzker, W. Umbrath, L. Hennig, C. Bauckhage, T. Alpcan and F. Metze, Tailoring Taxonomies for Efficient Text Categorization and Expert Finding, in *IEEE International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 459-462, 2008.

[7] A. McCallum, R. Rosenfeld, T. Mitchell and A. Y. Ng, Improving Text Classification by Shrinkage in a Hierarchy of Classes, in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, Madison, Wisconsin, USA, pp. 359-367, 1998.

[8] Y. Yang, J. Zhang and B. Kisiel, A Scalability Analysis of Classifiers in Text Categorization, in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval (SIGIR '03)*, pp. 96-103, Toronto, Canada, 2003.

[9] D. Ghazi, D. Inkpen and S. Szpakowicz, Hierarchical versus Flat Classification of Emotions in Text, in *NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pp. 140-146, Los Angeles, 2010.

[10] L. Cai and T. Hofmann, Hierarchical Document Categorization with Support Vector Machines, in *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM'04)*, pp 78-87, 2004.

[11] X. Qiu, W. Gao and X. Huang, Hierarchical Multi-Class Text Categorization with Global Margin Maximization, in *Proceedings of ACL-IJCNLP 2009 Conference Short Papers*, Singapore, pp. 165-168, 2009.

[12] F. Gao, W. Fu, Y. Zhong and D. Zhao, Large-Scale Hierarchical Text Classification Based on Path Semantic Vector and Prior Information, in *International Conference on Computational Intelligence and Security (CIS'09)*, pp. 54-58, Beijing, 2009.

[13] H. Schutze and C. Silverstein, Projections for Efficient Document Clustering, in *SIGIR 1997*, Philadelphia, pp. 74-81, 1997.

[14] K. Torkkola, Linear Discriminant Analysis in Document Classification, in *IEEE ICDM Workshop on Text Mining*, 2001.

[15] J. Yan, Q. Cheng, Q. Yang and B. Zhang, An Incremental Subspace Learning Algorithm to

Categorize Large Scale Text Data, in *Proceedings of the 7th Asia- Pacific Web Conference (APWeb 2005), LNCS 3399*, pp. 52-63, Shanghai, 2005.

[16] R. Salakhutdinov and G. Hinton, Semantic Hashing, *International Journal of Approximate Reasoning*, vol. 50(7), pp. 969-978, 2009.

[17] M. J. Gangeh, M. S. Kamel and R. P. Duin, Random Subspace Method in Text Categorization, in *IEEE 20th International Conference on Pattern Recognition (ICPR)*, pp. 2049-2052, Istanbul, 2010.

[18] S. Tulyakov, S. Jaeger, V. Govindaraju and D. Doermann, Review of Classifier Combination Methods, *Studies in Computatonal Intelligence (SCI)*, vol. 90, pp. 361-386, 2008.

[19] C. Manning, P. Raghavan and H. Schutze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[20] N. Tripathi, S. Wermter, C. Hung and M. Oakes, Semantic Subspace Learning with Conditional Significance Vectors, in *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 3670-3677, Barcelona, July 2010.

[21] T. Rose, M. Stevenson and M. Whitehead, The Reuters Corpus Volume 1 - from Yesterday's News to Tomorrow's Language Resources, in *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC- 02),*pp. 827-833, 2002.

[22] D. Zeimpekis and E. Gallopoulos, TMG : A MATLAB Toolbox for Generating Term Document Matrices from Text Collections, in *Book Chapter in Grouping Multidimensional Data: Recent Advances in Clustering*, J. Kogan and C. Nicholas, Eds., Springer, 2005.

[23] A. Kosmopoulos, E. Gaussier, G. Paliouras and Aseervatham, The ECIR 2010 Large Scale Hierarchical Classification Workshop, in *SIGIR Forum*, pp. 23-32, 2010.

[24] S. Ramasundaram and S. Victor, Algorithms for Text Categorization : A Comparative Study, *World Applied Sciences Journal* , vol. 22 (9), pp. 1232-1240, 2013.

[25] C. Aggarwal and Z. ChengXiang, A Survey of Text Classification Algorithms, in *Mining Text Data*, C. Aggarwal and Z. ChengXiang, Eds., Springer, pp. 163-222, 2012.

[26] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. Witten, The WEKA Data Mining Software: An Update, *ACM SIGKDD Explorations Newsletter*, vol. 11(1), pp. 10-18, July 2009.

[27] J. Quinlan, *C4.5 : Programs for Machine Learning*, San Mateo, CA.: Morgan Kaufmann Publishers, 1993.

[28] D. Rumelhart, J. McClelland and the PDP Research group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, Cambridge, MA: MIT Press..

[29] F. Pernkopf, Discriminative learning of Bayesian network classifiers, in *Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications*, 2007.

[30] E. Frank and I. Witten, Generating Accurate Rule Sets Without Global Optimization, in *Machine Learning: Proceedings of the Fifteenth International Conference*, J. Shavlik, Ed., Morgan Kaufmann Publishers, 1998.

[31] D. Lewis, Naive Bayes at Forty: The independence assumption in information retrieval, in *Proc. of the 10th European Conference on Machine Learning*, Chemnitz, Germany, 1998.

[32] R. Chhikara and L. Singh, An Improved Discrete Firefly and t-Test based Algorithm for Blind Image Steganalysis, in *Proc. of* ISMS '15, Washington, DC, USA, pp 58-63, 2015