8-2010

Text Classification with Imperfect Hierarchical Structure Knowledge

Thomas Ngo-Ye
University of Wisconsin-Milwaukee, linye@uwm.edu

Text Classification with Imperfect Hierarchical Structure Knowledge

Thomas Ngo-Ye
University of Wisconsin-Milwaukee
linye@uwm.edu

Abhijit Dutt
Pennsylvania State University, Beaver
aud18@psu.edu

ABSTRACT

Many real world classification problems involve classes organized in a hierarchical tree-like structure. However in many cases the hierarchical structure is ignored and each class is treated in isolation or in other words the class structure is flattened (Dumais and Chen, 2000). In this paper, we propose a new approach of incorporating hierarchical structure knowledge by cascading it as an additional feature for Child level classifier. We posit that our cascading model will outperform the baseline "flat" model. Our empirical experiment provides strong evidences supporting our proposal. Interestingly, even imperfect hierarchical structure knowledge would also improve classification performance.

Keywords

Hierarchical Text Classification, Imperfect Hierarchical Structure Knowledge, Cascading

INTRODUCTION

Text mining is used to uncover knowledge from unstructured data such as WebPages and text documents etc (Kuechler, 2007). One of the important applications of text mining is to classify documents (Fan, Wallace, Rich and Zhang, 2006). Most real world classification problems involve classes which are not flat. As an example, a web page could be classified as a sports page or a finance page. Within the category of sports there could be different sub-classes such as football, basketball, baseball etc. However in many cases the hierarchical structure is ignored and each class is treated separately or in other words the class structure is flattened (Dumais and Chen, 2000). In the text categorization literature, most researchers treat the classification structure as flat without considering the relationships among categories. Studies recognizing the hierarchical structure knowledge are relatively scarce (Qi and Davison, 2009).

It has been suggested that intelligent use of hierarchical structure information could improve classification accuracy significantly (Ruiz and Srinivasan, 2002). One possible explanation of the success of hierarchical model over the flat model is due to the divide-and-conquer principle (Dumais and Chen, 2000).

In this paper, we propose to exploit the hierarchical structure knowledge by cascading the knowledge as an additional feature down the chain. This cascading approach might lead to better classification accuracy when compared with a flat model. Our goal in this research is to demonstrate the efficacy of the above proposal. The rest of this paper is organized in the following way. First, we review existing research on hierarchical text classification. Second, we present our proposal of cascading hierarchical structure knowledge. Third, we describe our empirical experiments and we report the findings. Finally, we discuss the limitations and future directions and conclude with contributions and implications for practitioners.

LITERATURE REVIEW

Hierarchical Text Classification

Most existing studies of document classification model the categories only at one level or in other words a "flat" structure (Sun and Lim, 2001). This flat structure conceptualization does not represent real life classification systems. One example of common hierarchical classification systems is the "Library of Congress subject headings".

Hierarchical structure presents clear advantages in supporting tasks such as searching, browsing, and visualization (Cai and Hofmann, 2004). Automatic machine learning models could be applied to classify documents into a hierarchical structure and that could improve information retrieval tasks by reducing browsing and searching time needed for satisfying end user's information requirement. Automatically classifying texts into pre-defined taxonomy, is also important for knowledge management and content management in an organization, as many valuable corporate information is contained in unstructured texts (Cai and Hofmann, 2004).

Dumais and Chen (2000) conducted a study to apply support vector machines to hierarchical text classification. In their study hierarchical structure is used for two purposes. First, they trained the second-level classifier with two contrasting models – hierarchical case where the data are from the same top level parents, and flat case where data are from all categories. Second, they used both multiplicative scoring function and sequential Boolean function for combining scores from top and second-level classifiers. They found not only hierarchical model was more accurate than the baseline flat model but also it was more efficient in saving evaluation time (Dumais and Chen, 2000).

Wibowo and Williams (2002b) differentiated two approaches for hierarchical categorization – top down approach and combination schemes approach. The top down approach is intuitive. First, we assign a document to a child node of the root based on the similarity between the document and all children nodes of the root. After we make this labeling decision, we narrow our focus only to the chosen branch where the document is believed to belong. The categorization proceeds down the traverse path and we only concern the similarity to the relevant children notes at each step. Finally we reach a leaf node and assign it as the ultimate label to the document. The top down approach may lead to faster classification as it only consider a fraction of nodes at a given level and ignore irrelevant ones. Moreover, because of fewer categories to choose from, given the parent labeling is correct, the accuracy of child assignment might be higher. In the combination schemes approach, we need to calculate the similarity of a document to the classes in each level of the tree structure, and then combine the results from different levels to make a final assignment decision (Wibowo and Williams, 2002b). Dumais and Chen (2000)'s multiplicative scoring function and sequential Boolean function are examples of combining scores from different level classifiers.

Cascading Domain Knowledge to Enhance Data Mining

Domain or background knowledge about a problem has been used to improve efficiency of different data mining algorithms (Feelders, Daniels and Holsheimer, 2000; Hirsh and Noordewier, 1994). In traditional data mining research, domain knowledge is shown to be useful in enhancing machine learning model's performance. The output from a bank loan lending expert system is fed into a classifier and this addition of domain knowledge is proven to enhance classifier performance in terms of both lower misclassification cost and higher AUC. In essence this strategy of incorporating credit rating knowledge captured from an expert system into the learning process is equivalent to treating the expert system as a classifier and cascading it to another classifier (Sinha and Zhao, 2008). However, it is not intuitive how domain knowledge could be used in conjunction with text mining (Durfee, 2006).

OUR PROPOSAL OF CASCADING HIERARCHICAL STRUCTURE INFORMATION

In many situations there are valuable relationships among categories in a hierarchical structure. The first step in a hierarchical text classification is to define the concept hierarchy using domain knowledge (Chuang, Tiyyagura, Yang and Giuffrida, 2000). Between category members, a set inclusion relation, or so called IS-A relationship, is often encoded in a hierarchical structure (Ruiz and Srinivasan, 2002). For example, fish is an animal and trout is a fish.

Therefore, we can view the concept hierarchy embedded in the taxonomy structure, such as Yahoo! Directory, as a form of domain knowledge. In the flat model, the hierarchical structure information is discarded. We propose to use this hierarchical structure information, such as a web page is about sport, as domain knowledge and feed it to the flat model to enrich its feature set.

First we build a Parent level classifier, for example classifying a web page into categories such as sports or finance. Next we cascade the output from the Parent level classifier, which embed the domain knowledge of the hierarchical structure, to a Child level classifier. The newly generated information, for example web page category is sport, is used as a new feature and added to the existing conventional feature list of Child level classifier. We posit that this strategy of cascading hierarchical structure information as domain knowledge would enhance the accuracy of Child level classifier.

EMPIRICAL EXPERIMENT AND FINDINGS

Dataset Construction

Next we describe the dataset used in our empirical study. The dataset is available upon request from the first author. To empirically evaluate our proposed model of cascading hierarchical structure knowledge, we prefer to conduct a controlled text mining experiment. Simulation approach is often used for controlled experiment to sort out irrelevant factors and focus on the constructs of interest. Therefore, we adopt a numeric simulation study approach to imitate the distribution of words of texts for our hierarchical text classification problem. Without loss of generality, we construct the following general hierarchical structure.

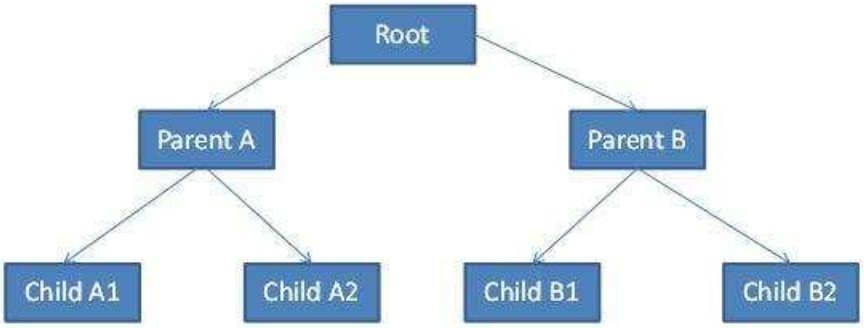

Figure 1. General Hierarchical Structure

Next we present the number of cases for each category for all levels. For our demonstration purpose we choose the total sample size of 1000. Each document case has actual category label of all levels. For example, one text might have label A and A1, meaning at the parent classification level, it belongs to class A and at child classification level, it belongs to class A1.

Category	Number of cases
Child A1	250
Child A2	250
Child B1	250
Child B2	250
Parent A	500
Parent B	500
Root	1000

Table 1. Sample Size

In conventional text classification, simple Bag-of-Words are the most common features (Sebastiani, 2002). Therefore, we construct simulated Bag-of-Words features for above 1000 document cases.

Conceptually, those words, which are good predictors for a category, have the property that they tend to appear relatively more frequent in target category and less often in other categories. We call these words "defining features" of a class. For example, some potential defining features for "sports" category might be "sports", "game", "players", and "win". On the contrary, "non-defining features" are those words appearing less frequent in the target class, but more often in other classes.

Based on above logic, we construct the following feature set. The complete text collection, in our study (1000 document cases), uses the same dictionary or wordlist as feature, even if some words are absent in some documents. For the simplicity of presentation and model, we only use 5 words as defining features for a particular category. We

need to point out that the defining features for Parent class A and Child class A1 are usually not the same. The denotation of a defining feature for child A1 is WA1_*, where W means word, A1 is the category and wildcard * ranges from 1 to 5 to represent the first defining word, the second, etc.

Category	Defining features				
Parent A	WA_1	WA_2	WA_3	WA_4	WA_5
Parent B	WB_1	WB_2	WB_3	WB_4	WB_5
Child A1	WA1_1	WA1_2	WA1_3	WA1_4	WA1_5
Child A2	WA2_1	WA2_2	WA2_3	WA2_4	WA2_5
Child B1	WB1_1	WB1_2	WB1_3	WB1_4	WB1_5
Child B2	WB2_1	WB2_2	WB2_3	WB2_4	WB2_5
Random Noise	WN1	WN2	WN3	WN4	WN5

Table 2. Defining Features for Categories

A document is represented by above 35 words. Different documents have different realized values for the features based on the type of feature relative to the target class. For example, for a document belonging to A1, the values for WA1_1, WA1_2, WA1_3, WA1_4, and WA1_5 have relatively large values. Moreover, since the document belongs to A1, it also belongs to class A. The values of WA_1, WA_2, WA_3, WA_4, and WA_5 are relatively large. For the document, other features' values are relatively small.

While the first six rows of features are defining features for a category, in real world text, there are other words that are not defining features at all. We call them as "random noise", as they do not contribute to the classification task. We denote them as WN*, where N means noise.

To illustrate how the text collection is represented by our above proposed feature set, we present the following simplified conceptual view.

	Parent Class	Child Class	WA_*	WB_*	WA1_*	WA2_*	WB1_*	WB2_*	WN*
Document 1	A	A1	1	0	1	0	0	0	0.5
Document 2	A	A2	1	0	0	1	0	0	0.5
Document 3	B	B1	0	1	0	0	1	0	0.5
Document 4	B	B2	0	1	0	0	0	1	0.5

Table 3. Perfect Conceptual Representation

In above table we only show 4 typical documents, one for each child category. The * means the index ranges from word 1 to word 5. The value "1" in a cell indicates that the feature is a defining feature for the target class and it has higher chance of appearing in a document. The value "0" in a cell indicates that the feature is a non-defining feature for the target class and it has lower chance of appearing in a document. Finally the value "0.5" in a cell indicates that the feature is a random noise for the target class, and its chance of appearing in a document is random with the likelihood of 0.5.

While above table captures the essence of perfect hierarchical text representation, in real world, the distribution of words relative to the target category is less clear and cut as highlighted above. The words' distribution has some inherent variations. To model the likelihood for a word appearing in a document, P (Word, Doc), we use the following heuristics.

First, if a word is a defining feature of a target class, on average, it is more likely to appear and we model it as a random variable following uniform distribution, P(Word, Doc) ~uniform(-1, 3). The mean is 1, implying presence.

Second, if a word is a non-defining feature of a target class, on average, it is less likely to appear and we model it as a random variable following uniform distribution, P(Word, Doc) ~uniform(-2, 2). The mean is 0, implying absence.

Finally, if a word is a random noise feature of a target class, on average, it is equally likely and unlikely to appear and we model it as a random variable following uniform distribution, P(Word, Doc) ~uniform(-2.5, 3.5). The mean is 0.5, implying totally random.

With above three rules, we populate values of feature set with Independently and Identically Distributed (iid) random numbers. In addition to above 35 features, for each document, we have the actual label of Parent class (A or B) and Child class (A1, A2, B1, or B2) as ground truth. The given ground truth of classification label is necessary for supervised learning.

Implementation of Proposed Models and Hypotheses

Next we describe the implementation of models driven by our proposal and hypotheses. First we establish the baseline scenario where no hierarchical structure information is used, in other words the flat model at Child classification level. The class attribute is Child classification with label A1, A2, B1, and B2. The information of Parent class label is withheld from the model.

In the second model, we presume to have the ground truth of Parents class label and therefore we cascade it as a new feature to above baseline model. We use the 35 features plus the actual Parent class label to predict the Child class label. We expect the classification accuracy will be higher compared to the baseline model, as the additional hierarchical structure knowledge may guide classifiers fine tune and narrow down the search space of potential hypotheses. The above argument has the same underlying rationale as ensemble learners could achieve better performance because of the reduction of the vast hypotheses space (Dietterich, 2000).

However, in real world, the ground truth of Parent class label is unknown or the knowledge is imperfect. In this case, we first build a classifier to classify documents at the Parent class level (whether a document belongs to A or B). Then we obtain the prediction from the trained model. The prediction might not be perfect accurate compared to the ground truth. However, we argue that even with such imperfect knowledge of hierarchical structure, it still might help the classification at Child level.

So in the third model, we first train a classification model for Parent level classification. Then we cascade down the prediction of Parent class label as an additional feature for the classification at Child level. We expect the imperfect information would overall help Child classification compared to the baseline. Moreover, it is straightforward to conceive that the performance upper bound for imperfect knowledge model is the performance of model with perfect knowledge or ground truth of Parent label.

Therefore we expect the following relationships from the experiment:

- **Hypothesis 1**: Performance(With Perfect Knowledge) > Performance(Baseline)
- **Hypothesis 2**: Performance(With Imperfect Knowledge) > Performance(Baseline)
- **Hypothesis 3**: Performance(With Imperfect Knowledge) <= Performance(With Perfect Knowledge)

To implement our proposed models, we construct different datasets for each model. For the baseline model, the dataset looks like:

(WA_*,WB_*,WA1_*,WA2_*,WB1_*,WB2_*,WN*) → Child Class

For the second model with perfect knowledge, the dataset looks like:

(WA_*,WB_*,WA1_*,WA2_*,WB1_*,WB2_*,WN*,Actual Parent Class Label) → Child Class

For the third model with imperfect knowledge, in the first stage we build model to classify Parent Class with dataset:

(WA_*,WB_*,WA1_*,WA2_*,WB1_*,WB2_*,WN*) → Parent Class

Then for each document, we obtain the Prediction value – Predicted Parent Class Label. We find the predicted Parent class label is not 100% accurate. Thus it represents imperfect knowledge of hierarchical structure. Next we report the error rate of our trained model for predicting Parent class label.

Classifier	Error rate
NaiveBayes	0.025
J48	0.006
SMO	0.028
Jrip	0.014
IBk3	0.038

Table 4. Error Rate for Predicting Parent Class Label

Then we cascade this imperfect knowledge down for Child classification with dataset:

(WA_*,WB_*,WA1_*,WA2_*,WB1_*,WB2_*,WN*, Predicted Parent Class Label) → Child Class

In above datasets, the features to the left of "→" are predictor variables and the variable to the right of "→" is the dependent variable.

Experiment Configuration, Findings and Analysis

We select Weka (Witten and Frank, 2005) as a tool for our text mining task. Weka includes a large variety of classification algorithms and experiment facility to compare classifiers across multiple datasets. We choose the following five classifiers: Naïve Bayes, J48, SMO, JRip, and IBk. Above five classifiers have diversified theoretical background and represent different approaches to classification.

Naive Bayes is a robust and efficient algorithm for text classification. J48 is Weka's implementation of C4.5 decision tree, a widely used classifier. SMO is sequential minimal optimization algorithm, a support vector machine classifier (Platt, 1998). JRip is Weka's implementation of Repeated Incremental Pruning to Produce Error Reduction (RIPPER), a propositional rule learner (Cohen, 1995). IBk is K-nearest neighbors classifier (Aha and Kibler, 1991). We set the K to arbitrary value 3, or 3 nearest neighbors as in (Ngo-Ye and Dutt, 2009; Sinha and Zhao, 2008).

For each classifier, we compared three models (Baseline, With Perfect Knowledge, and With Imperfect Knowledge) as outlined above. So we setup five experiments, one classifier a time. Within each experiment, we have three datasets: "Baseline", "WithParentLabel", and "WithParentPred*". We should point out that "Baseline" and "WithParentLabel" datasets maintain the same across 5 experiments. However for "WithParentPred*" or "with imperfect Knowledge", we have different dataset for different classifier, as the prediction from different classifier varies. For example, if we use Naïve Bayes to predict Parent class label, we should also use Naïve Bayes to predict Child class label to be consistent. So the "*" denote the index ranging from NB, J48, SMO, JRip, to IBk3.

Within each individual experiment of a classifier, we conduct 10 runs for averaging and within each run, we use 10 fold cross validation for reliable estimation. Therefore for one individual experiment, we have 3 datasets X 1 classifier X 10 runs X 10 fold cross validations = 300 observations of performance. We aggregate the performance for each dataset with 100 observations. Next we report the mean value (and standard deviation) of accuracy of each algorithm for each dataset.

	Baseline	WithParentLabel	WithParentPred
NaiveBayes	86.91(3.26)	89.19(2.96)*	86.97(3.23)
J48	85.99(3.36)	91.89(3.00)*	91.28(3.08)*
SMO	85.16(3.67)	88.21(3.26)*	85.68(3.60)
Jrip	82.72(4.00)	89.51(3.22)*	88.03(3.57)*
IBk3	71.85(4.51)	80.96(3.60)*	77.31(3.92)*

Table 5. Accuracy Mean (Standard Deviation) of Classifiers on Datasets

We perform paired T test between WithParentLabel and Baseline, and between WithParentPred and Baseline. The symbol "*" denotes that a model is significantly more accurate than the baseline model at significant level of 0.05. From above table, we find that model with perfect knowledge performs more accurately than the baseline model across all classifiers. Comparing model with imperfect knowledge with Baseline model, in three out of five classifiers, imperfect knowledge model outperforms Baseline model. For the rest two (Naïve Bayes and SMO), the result is not statistically significant, but the mean difference is positive.

So overall, we find strong support for our hypothesis 1, which states Performance(With Perfect Knowledge) > Performance(Baseline). We also find reasonable support for our hypothesis 2, which states Performance(With Imperfect Knowledge) > Performance(Baseline). Moreover, when we perform paired T test to compare model with imperfect knowledge and model with perfect knowledge, we find perfect knowledge model performs better in some situations. In other situations, their accuracy means are not statistically different. So we obtain strong support for our hypothesis 3, which states Performance (With Imperfect Knowledge) <= Performance(With Perfect Knowledge).

DISCUSSIONS

In our simulation study, some values of features are negative, because the lower bound of uniform distribution is a negative number. However, in text mining, the weighting of a word is usually non-negative, whether using binary representation of presence or absence, or word count, of TFIDF. Therefore we conduct a second experiment where we add a constant number, 2.5, to all values to make them non-negative. The new result is almost identical to the original one.

Our empirical experiments lend strong support to our proposal that incorporating hierarchical structure knowledge – cascading Parent class label as an additional feature, enhances classification at Child level. The results are robust and consistent for both perfect knowledge scenario and for imperfect knowledge scenario.

Our proposed cascading model is different from other approaches reported in hierarchical text classification literature. Top down approach and schemes combination approach are the two major methods for hierarchical text classification (Dumais and Chen, 2000; Wibowo and Williams, 2002b). In our proposed approach, we take a two-stage approach and simply pass down the predicted Parent class label to the Child classification model. Therefore our approach is a straightforward extension of conventional "flat" model, which ignores the hierarchical structure knowledge. We demonstrate that our proposed methodology is viable and outperforms baseline "flat" model. Our model is also intuitive and it is easy to implement. These issues make our approach attractive to practitioners.

LIMITATIONS AND FUTURE DIRECTIONS

We empirically demonstrate our approach only with two-level hierarchical classification. However, the method can easily be adapted in multilayer hierarchical structure situation.

We simulate the representation of documents with only 35 words as the feature set. In real world text mining application, a text corpus can easily have thousands of words in the pooled feature set. We leave it to the future study to test whether our results would be applicable in that scenario.

In our current study we use uniform distribution to generate random numbers to represent word distribution in a text. In future, other probability distributions, such as normal distribution, Poisson distribution, or power law distribution could be used.

Moreover, in current study, we only evaluate performance using the most common measure in data mining, accuracy. In the future, we plan to consider other additional measures appropriate for text mining such as precision, recall, and F measure. Furthermore, the misclassification cost is not considered explicitly here. In the future work, cost-sensitive learning such as ROC analysis (Provost and Fawcett, 2001) can be applied.

Another potential extension of our work is to empirically compare our cascading knowledge approach with the traditional top down approach and schemes combination approach for hierarchal text classification.

Finally, we could further test our proposed model on real world hierarchal text collection. In fact a new project is undergoing using web crawler to collect WebPages from Internet. We are looking forward to validate our proposal on a realistic dataset.

CONTRIBUTIONS AND IMPLICATIONS FOR PRACTITIONERS

Our study makes contribution to the hierarchal text classification literature by proposing and validating a new approach to incorporate hierarchal structure knowledge for text mining. We demonstrate that even with imperfect knowledge of hierarchal structure, our proposed model is still compared favorably with the baseline "flat" model.

Our proposed methodology provides an alternative for practitioners facing hierarchal text classification problem. Our model is not only efficient, but also easy to understand and implement.

REFERENCES

1. Aha, D., and Kibler, D. (1991) Instance-based learning algorithms. *Machine Learning , 6*, 37-66.
2. Apte, C., Liu, B., Pednault, E. P. and Smyth, P. (2002) Business applications of data mining,. *Communications of the ACM , 45* (8), 49-53.
3. Cai, L., and Hofmann, T. (2004) Hierarchical Document Categorization with Support Vector Machines. *CIKM*, Washington, DC, 78-87.
4. Chen, H., and Dumais, S. (2000) Bringing order to the Web: Automatically categorizing search results. *The SIGCHI Conference on Human Factors in Computing Systems*, New York, NY: ACM Press, 145–152.
5. Choi, B., and Peng, X. (2004) Dynamic and hierarchical classification of web pages. *Online information Review , 28* (2), 139-147.
6. Chuang, W. T., Tiyyagura, A., Yang, J. and Giuffrida, G. (2000) A Fast Algorithm for Hierarchical Text Classification. In M. M. Y. Kambayashi (Ed.), *DaWaK 2000, LNCS 1874*, Springer-Verlag Berlin Heidelberg, 409-418.
7. Cohen, W. W. (1995) Fast Effective Rule Induction. *Twelfth International Conference on Machine Learning,* Morgan Kaufmann, 115-123.
8. Davidov, D., Gabrilovich, E. and Markovitch, S. (2004) Parameterized Generation of Labeled Datasets for Text Categorization Based on a Hierarchical Directory. *SIGIR'04*, Sheffield, South Yorkshire, UK, 250-257.
9. Dhillon, I. S., Mallela, S. and Kumar, R. (2002) Enhanced Word Clustering for Hierarchical Text Classification. *KDD*, Alberta, CA: Edmonton, 191-200.
10. Dietterich, T. G. (2000) An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. (D. Fisher, Ed.) *Machine Learning , 40*, 139-157.
11. Dumais, S. and Chen, H. (2000) Hierarchical classification of web content. *The 23 rd. ACM Int. Conf. on Research and Development in Information Retrieval*, Athens, Greece, 256-263.
12. Durfee, A. V. (2006) Text Mining Promise and Reality. *The Twelfth Americas Conference on Information Systems (AMCIS)*, Acapulco, Mexico, 1449-1460.
13. Fan, W., Wallace, L., Rich, S. and Zhang, Z. (2006) Tapping into the Power of Text Mining. *Communications of the ACM , 49* (9), 77-82.
14. Feelders, A., Daniels, H. and Holsheimer, M. (2000) Methodological and practical aspects of data mining. *Information and Management , 37*, 271-281.
15. Hearst, M. A. (2006) Clustering versus Faceted Categories for Information Exploration. *Communications of the ACM , 49* (4), 59-61.
16. Hirsh, H. and Noordewier, M. (1994) Using background knowledge to improve inductive learning of DNA sequences. *IEEE Conference on AI for Applications*, 351-357.
17. Kroenke, D. (2007) *Experiencing MIS* (1st ed.). Prentice Hall.
18. Kroenke, D. M. (2006) *Database Processing Fundamentals, Design, and Implementation* (10th ed.). Upper Saddle River, New Jersey: Pearson Prentice Hall.
19. Kuechler, W. L. (2007) Business application of unstructured text. *Communications of the ACM , 50* (10), 86-93.
20. Levine, D. M., Berenson, M. L. and Stephan, D. (1998) *Statistics for Managers Using Microsoft Excel.* Upper Saddle River, New Jersay: Prentice Hall.

21. Liu, T.-Y., Yang, Y., Wan, H., Zeng, H.-J., Chen, Z. and Ma, W.-Y. (2005) Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations , 7* (1), 36–43.
22. Mandreoli, F., Martoglia, R. and Tiberio, P. (2005) Text clustering as a mining task. In A. Zanasi (Ed.), *Text Mining and its Applications to Intelligence, CRM and Knowledge management.*
23. Ngo-Ye, T. and Dutt, A. (2009) A Study on Efficacy of Ensamble Methods for Classification Learning. *Thirtieth International Conference on Information Systems (ICIS)*, Phoenix,Arizona, 1-10.
24. Peng, X. and Choi, B. (2002) Automatic Web Page Classification in a Dynamic and Hierarchical Way. *The IEEE International Conference on Data Mining (ICDM)*, Los Alamitos, CA: IEEE Computer Society Press, 386–393.
25. Platt, J. (1998) Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. S. Smola (Ed.), *Advances in Kernel Methods - Support Vector Learning.*
26. Provost, F. and Fawcett, T. (2001) Robust Classification for Imprecise Environments. *Machine Learning , 42*, 203-231.
27. Qi, X. and Davison, B. D. (2009) Web Page Classification: Features and Algorithms. *ACM Computing Surveys , 41* (2), 12:1-12:31.
28. Rousu, J., Saunders, C., Szedmak, S. and Shawe-Taylor, J. (2006) Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research , 7*, 1601-1626.
29. Ruiz, M. E. and Srinivasan, P. (2002) Hierarchical Text Categorization Using Neural Networks. *Information Retrieval , 5*, 87–118.
30. Sebastiani, F. (2002) Machine Learning in Automated Text Categorization. *ACM Computing Surveys , 34* (1), 1-47.
31. Sebastiani, F. (2005) Text categorization. In A. Zanasi (Ed.), *Text Mining and its Applications to Intelligence, CRM and Knowledge management.*
32. Sinha, A. P., and Zhao, H. (2008) Incorporating domain knowledge into data mining classifiers: An application in indirect lending. *Decision Support Systems , 46*, 287-299.
33. Stenger, B., Thayananthan, A., Torr, P. and Cipolla, R. (2007) Estimating 3D hand pose using hierarchical multi-label classification. *Image and Vision Computing , 5* (12), 1885-1894.
34. Sun, A. and Lim, E.-P. (2001) Hierarchical Text Classification and Evaluation. *The first IEEE International Conference on Data Mining (ICDM'01)*, Los Alamitos, CA: IEEE Computer Society Press, 521-528.
35. Toutanova, K., Chen, F., Popat, K. and Hofmann, T. (2001) Text Classification in a Hierarchical Mixture Model for Small Training Sets. *The Tenth International Conference on Information and Knowledge Management (CIKM'01)*, Atlanta, Georgia: ACM Press, 105-112.
36. Vens, C., Struyf, J., Schietgat, L., Dzeroski, S. and Blockeel, H. (2008) Decision trees for hierarchical multi-label classification. (J. Furnkranz, Ed.) *Machine Learning , 73*, 185-214.
37. Weiss, S. M., Indurkhya, N., Zhang, T. and Damerau, F. J. (2005) *Text Mining.* Springer Verlag.
38. Wibowo, W. and Williams, H. E. (2002a) Simple and accurate feature selection for hierarchical categorisation. *The 2002 ACM Symposium on Document Engineering (DocEng*), New York, NY: ACM Press, 111-118.
39. Wibowo, W. and Williams, H. E. (2002b) Strategies for minimising errors in hierarchical Web categorisation. *The Eleventh International Conference on Information and Knowledge Management (CIKM)*, New York, NY: ACM Press, 525–531.
40. Witten, I. H. and Frank, E. (2005) *Data Mining: Practical machine learning tools and techniques* (2nd ed.). San Francisco: Morgan Kaufmann Publishers.