MAPPING SEARCH RESULTS INTO SELF-CUSTOMIZED CATEGORY HIERARCHY

Saravadee Sae Tan[1], Gan Keng Hoon[1], Chan Huah Yong[2], Tang Enya Kong[1] and Cheong Sook Lin[1]
[1]*Computer Aided Translation Unit (UTMK)*
School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia

[2]*Grid Computing Lab*
School of Computer Sciences, Universiti Sains Malaysia, 11800 Penang, Malaysia
{saratan | khgan | hychan | enyakong | slcheong}@cs.usm.my

Abstract: With the rapid growth of online information, a simple search query may return thousands or even millions of results. There is a need to help user to access and identify relevant information in a flexible way. This paper describes a methodology that automatically map web search results into user defined categories. This allows the user to focus on categories of their interest, thus helping them to find for relevant information in less time. Text classification algorithm is used to map search results into categories. This paper focuses on feature selection method and term weighting measure in order to train an optimum and simple category model from a relatively small number of training texts. Experimental evaluations on real world data collected from the web shows that our classification algorithm gives promising results and can potentially be used to classify search results returned by search engines.

Key words: information retrieval, results classification, feature selection, self-customized categories

1. INTRODUCTION

As the number of online information is growing rapidly, it becomes more and more difficult in finding relevant information. Two common tools that assist user in retrieving information from the web are *web search engines*

and *web directories*. Search engines return a ranked list of results in response to a user search query. Nowadays, a simple query may return thousands (or even millions) of results. Results of different topics are mix together and the user has to browse through the long ranked list to find for relevant information. On the other hand, web directories such as Yahoo! and LookSmart classify web pages into hierarchical structured categories. However, the manual classification of web pages into directory makes web directories impossible to keep up with the rapid growth of the web.

Chen and Dumais [3] present a user interface that organizes web pages returned by a search engine into hierarchical structured categories. This approach combines the advantage of broad coverage in search engines and structured categories in web directories. Their study [4] shows that the category interface increase the speed in finding information compared to a long ranked list. Paper [3] more focuses on the category interface instead of the classification algorithm. In this paper, we present a classification algorithm that automatically classifies search results into user defined category hierarchy.

In our methodology, a user is allowed to define a set of categories of his interest by providing a set of training text for each category. Our classification algorithm automatically learns the characteristics of the categories from the training texts. Later, it would be able to map the results returned by a search engine into categories based on the knowledge learned. Intuitively speaking, the user has the flexibility to define the categories as well as the concepts in which each category represent. For this reason, he may foresee the information or content being classified in each category thus can directly focus on relevant topics. This gives more flexibility to the user compare to web directories, in which he is only allowed to browse the topic hierarchy defined by editors.

In our classification algorithm, characteristic of the categories are represented using a Category Model (CM). The CM consists of the categories along with a set of keywords for each category. The keywords are automatically generated from the training texts. The quality of CM will strongly affect the performance of the classification algorithm. Therefore, in this paper, we pay more attention on the feature selection method as well as the term weighting measure in order to increase the ability of our classification algorithm in selecting a set of optimum keywords from a relatively small number of training texts.

We have previously proposed a feature selection method in annotating a topic hierarchy [9]. In this paper, we show that the idea from our previous work is also applicable to text classification task.

2. CLASSIFICATION ALGORITHM

The main aim of the classification task is to map search results into user defined categories. In order to apply real time classification, a classification algorithm with high accuracy and reasonable processing time is required. Figure 1 shows the processing flow of our results classification algorithm. It consists of two phase: category model generation phase and results classification phase.

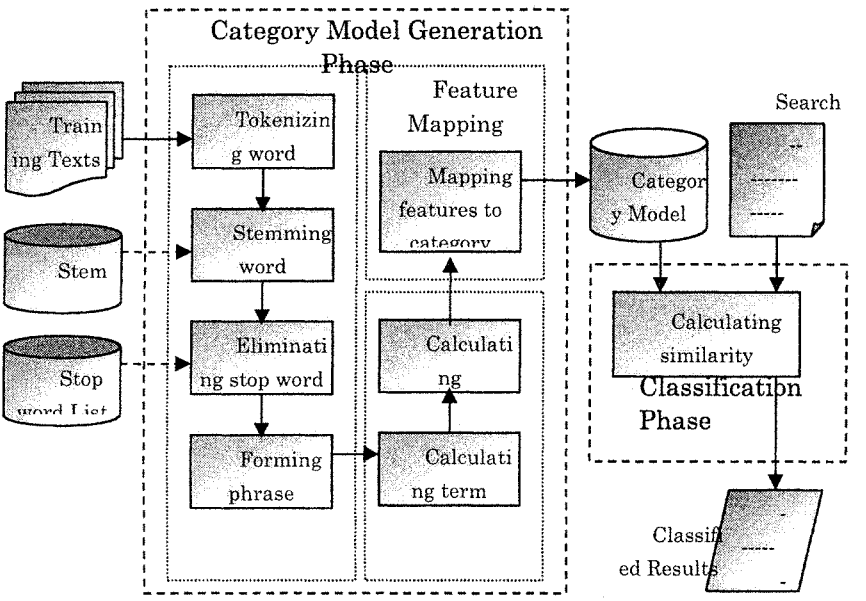

Figure 1. Processing flow of the classification algorithm

3. CATEGORY MODEL (CM) GENERATION

Size and quality of a category model are major concerns in the performance of our classification algorithm. Since the CM is prepared beforehand, processing time is not a major issue and it will not affect the real time classification. Thus, we propose methods on feature generating and selecting in order to compose an optimum CM for higher accuracy and faster processing.

3.1 Document Representation

A simple and frequently used document representation is the feature vector representation. It is a 'bag-of-words' representation, in which each document is represented as a vector of features. Mladenic and Globelnik [6] extend the 'bag-of-words' representation to 'bag-of-phrases' representation by adding new features generated from word sequences, also know as N-grams.

In this paper, we define each distinct *term* or *phrase* corresponds to a feature. By using phrases as features, we attempt to preserve the information left out by single word representation. We use up to 3-grams feature vector representation. Experiments show that word sequences of length up to 3 is sufficient in most classification systems [6].

3.2 Feature Pre-Processing

In order to transform a text document into a feature vector, pre-processing is needed. Four type of pre-processing are considered in this paper.

Word tokenizing: Single words are extracted from documents. All capitalized letters are converted into lower case.

Word stemming: All single words extracted are replaced by their stem. Words variants usually only convey one piece of information. Thus, it would be more reasonable to reduce them into one representative for both efficiency and effectiveness concerns.

Stop word eliminating: The high dimensionality of a feature space is initially reduced by eliminating all stop words. Stop words are words that do not take semantic meaning themselves. They are not helpful to capture documents' semantic, hence they are irrelevant to classification. Two type of stop word are considered in this paper, 'general stop words' (e.g. *a*, *the*, *of*...) and 'web page stop words' (e.g. *html*, *home*...).

Phrase forming: Phrasal feature consist of up to 3 stemmed token occurring as a sequence in a document are formed.

3.3 Feature Selection

Feature selection is used as a basic step in the process of building a classifier. It is a process that chooses a subset of features from the original set of features so that the feature space is optimally reduced according to a certain criterion. This tends to produce classification models that are simpler, clearer, and computationally less expensive [7]. In our methodology, feature

selection method is used to select the set of feature that compose the Category Model.

Various approaches of feature selection have been developed for dimensionality reduction in a classification task. Basically, these methods can be broadly divided into 2 main approaches, (i) Feature Selection in Machine Learning, and (ii) Feature Selection in Text Learning. Feature Selection in Machine Learning traverse a search space and evaluate every candidate subset in order to find the best subset. These methods are less practical when the number of features is large. On the other hand, Feature Selection in Text Learning evaluates every feature independently, in which a scoring criterion is used to measure the goodness of a feature. All features are sorted in a list and a predefined number of best features are selected. However, the number of features to be selected is a main experimental issue in these methods.

Feature selection method adopted in this paper is from the author's previous work. It combines the idea from both methods of feature selection in machine learning and text learning [9]. All features are sorted in a list according to their significance in term of discriminating power between categories. An optimum set of features is selected by finding a cut-off point in the list using consistency measure.

3.3.1 Term Weighting

Intuitively, frequency may be a main indicator of the importance of a term. This comes from two major concerns. A term is considered as representative if it appears many times *within* a document. On the other hand, a term is regarded as not informative if it appears too many times *among* documents [1]. These two aspects are commonly used in term weighting functions.

TFIDF is a commonly used term weighting function in a classification task [2][8]. TFIDF measures the importance of a feature in term of discriminating power between documents using the two aspects mentioned previously. The former addresses how frequent a term appears *within* a document, which is known as term frequency (TF). The latter measures how many documents that the term appears among all documents, which is known as inverse document frequency (IDF).

Term weighting function presented in this paper, named TFDF is an extension of TFIDF. TFDF measures the importance of a feature in term of discriminating power among categories. Frequency distribution of a feature across categories is taken as a basis in weighting a term. Two main aspects considered are:

Term Frequency **(TF)**: TF of a feature f_i denotes the frequency occurrence of the term in a category C_k. An additional factor, total frequency of f_i in all other categories, is added in order to boost terms that occur more frequent in category C_k compare to other categories.

Although raw TF does emphasize those features with high frequency, normalization is needed because training texts are not of uniform length. Therefore, relative frequency is preferred. *Maximum frequency normalization* is used in this paper, in which the term frequency of a feature is divided by the highest frequency over all the features in category C_k.

$$W_{TF}(f_i, C_k) = \frac{TF(f_i, C_k)}{TF_{\max}(C_k) \times \left(1 + \sum_{l \neq k} TF(f_i, C_l)\right)}$$

The rational behind term frequency is that the ability of a feature in discriminating categories depends on how frequently it occurs in a category as against the other categories.

Document Frequency **(DF)**: DF of a feature f_i denotes the number of documents in category C_k in which the term occurs at least once. Again, we boost terms that occur in more documents in category C_k against the rest of categories. This is done by dividing DF with the total number of documents that contain f_i in other categories.

Since the number of training text in each category is different, we consider the relative frequency by normalizing DF with total number of documents in the category.

$$W_{DF}(f_i, C_k) = \frac{DF(f_i, C_k)}{DF_{total}(C_k) \times \left(1 + \sum_{l \neq k} DF(f_i, C_l)\right)}$$

The main idea of document frequency is that features that occur in more documents in a category against other categories are more discriminative than features that occur in many documents in many categories.

By combining both TF and DF weights together, we can obtain a final TFDF weight of a feature f_i in a particular category C_k, as follow.

$$TFDF(f_i, C_k) = W_{TF}(f_i, C_k) + W_{DF}(f_i, C_k)$$

3.3.2 Consistency Measure

The size of Category Model is a main concern in the processing speed of our classification algorithm. Thus, it is important to concern the set of features selected in order to compose an optimum CM. We expect that the

selected features are informative enough to represent the concepts of the categories, neither too few to miss semantics or too many to burden the processing speed.

In this paper, a set of selected feature is evaluated by *class separability* measure. A feature subset is considered 'optimal' when it maximizes the class separability within a domain. Consistency measure is a conservative way of achieving class separability. It does not attempt to maximize the class separability but tries to retain the power of class separability defined by the original set of features. The idea is to find a smallest set of features that can distinguish classes as well as the full set of features. [5].

In our feature selection method, consistency measure is applied in the task of finding cut-off point in the list of features sorted by their weight. The measure is an inconsistency rate over the training text for a given feature set. This criterion heavily relies on the set of training text and it uses Min-Features bias in selecting a feature subset.

3.4 Mapping Feature to Category

From the definition of term weighting function in Section 3.3.1, the weight indicates the importance of a feature in discriminating a particular category from the rest of the categories. Thus, a feature is mapped to the category that has the highest weight. Intuitively speaking, a feature is treated as a descriptive keyword for the category, which can express the concept of the category more clearly.

4. RESULTS CLASSIFICATION

Every result is represented by a Boolean vector in our Category Model vector space in which the Boolean value denotes the occurrence of a particular feature in the result.

4.1 Similarity Calculating

Result classification is typically performed by comparing representation of the result with representation of categories and computing a measure of their similarity. The measure is a *cosine* value between two vectors. The smaller angle the two vectors has, the more similar the results and the contents of the category. The cosine value of a result r with every category C_k is calculated and the result is assigned to the most similar category.

$$\cos ine\theta(r, C_k) = \frac{r \cdot C_k}{|r||C_k|}$$

5. EXPERIMENTS

5.1 Data Collection

In our experiments, we used a human edited directory of the web, Open Directory Project (ODP) as reference in defining our own categories. We selected a total of 9 categories as our category hierarchy. For each category, we collect a set of web pages as training text. These web pages are selected manually in which they can comprise the concepts represented by each category.

As testing text, we collect 892 distinct site listing from the ODP sub-categories pages. For every site listing, we generate two testing texts, the first testing text only contain a title and a short description provided by the editor in ODP. The second testing text is the full text of the web site.

The first set of testing text contains less content. However, the content is more 'clean' and 'informative'. This is because the title and description are provided by human editor and may fully describe a particular site. We include this set of testing text in order to provide an overview on how our classification algorithm can be applied on text data with limited content. This is because search result returned by search engines usually only contains a title and a short description about the site.

The full text of a web site may contain more 'noise' due to the heterogeneous of web pages. The contents vary from page to page, where some pages may contain useful textual information while others may contain only graphics, animation, multimedia and etc. Our classification algorithm considers only textual information in the classification task. In this paper, we use the text contain in the first page of the web site as our testing text.

Table 1 shows the number of training and testing text collected for every category.

Table 1: The data set used in the evaluation of our classification algorithm

Categories	Sub-Categories	No of Training Text	No of Testing Text	
			Title & Desc	Full Text
Artificial Intelligence	Artificial Life	43	100	100

	Fuzzy Logic	31	29	29
	Genetic Programming	33	48	48
	Neural Networks	39	87	87
Natural Language Processing	Information Management	60	135	135
	Language Analysis	41	92	92
	Machine Translation	34	69	69
	Speech Processing	38	184	184
High Performance Computing	-	59	148	148
Total		378	1784	

5.2 Performance Measure

Every testing text has one correct category (its ODP category). The performance is measured by calculating the accuracy of our classification algorithm.

$$\text{Accurac} \quad \frac{\text{Number of testing text that are assigned to the correct category}}{\text{Total number of testing text}}$$

5.3 Experimental Results and Discussion

We carried out experiments to test on the performance of our classification algorithm. The set of training text is used to train our Category Model (CM). Later, the CM is used to classify the testing text. Two type of testing text are prepared, (i) title and description of web sites and (ii) full text of the web sites.

We first compare the impact of different feature types ('single terms' and 'phrases') using the TFDF term weighting function presented in Section 3.3.1. 'Single term' means only individual word component are considered as features, whereas 'phrases' means phrases of length up to 3 are generated as features. Figure 1 and Figure 2 show the results of our classification algorithm using different top n% of features from the feature list to generate the category model.

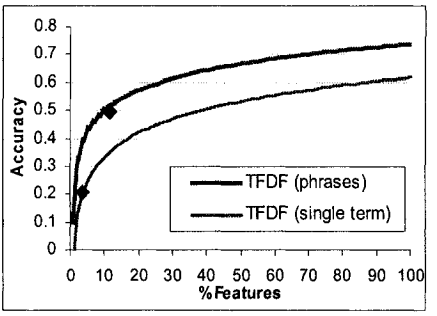

Figure 2. Accuracy of classification for 'Title & Desc' testing texts

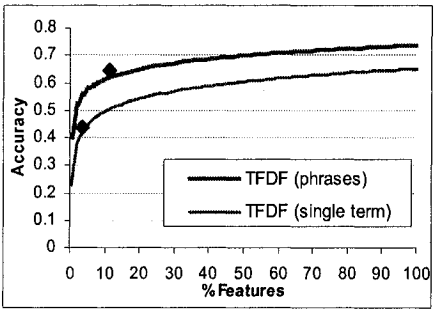

Figure 3. Accuracy of classification for 'Full Text' testing texts.

Figure 2 and Figure 3 show that the overall performance for 'phrases' feature type is better than 'single term'. The results prove that considering phrases as features can preserve the information left out by single words and thus increase the accuracy of our classification algorithm.

Figure 2 and Figure 3 also show a trend for the performance of our classification algorithm. The classification accuracy increases with the number of features that contain in our CM and become more stable after a certain point. More features may give us a better performance; however, the high dimensionality of feature space in CM may create a more complex classifier and increase the processing time. In order to find a compromise point between the accuracy and the size of CM, we apply the consistency measure to determine the size of our CM. We also show the cut-off point in our feature list using consistency measure discussed in Section 3.3.2.in the figures above. The consistency measure applied in our feature selection method can be useful in helping us to predict a compromise point between the accuracy and the size of our CM.

It is interesting to see how the performance of our classification algorithm can be improved by considering additional aspects in term weighting function as presented in Section 3.3.1. We test the performance of our classification algorithm using *raw term frequency* as term weighting function. The results comparison between raw TF and TFDF are shown in Figure 4 and Figure 5.

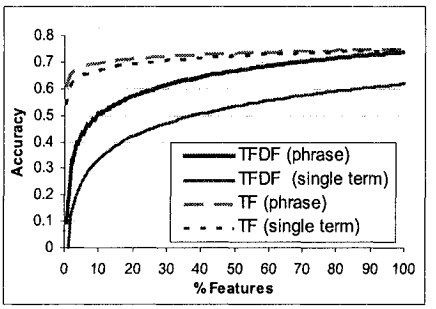

Figure 4. Results comparison of TF and TFDF for 'Title & Desc'

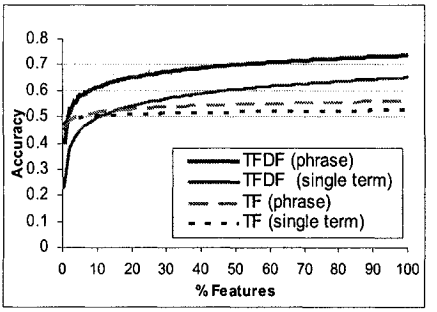

Figure 5. Results comparison of TF and TFDF for 'Full Text' testing texts.

For 'clean' and 'informative' testing texts, TF alone can give us encouraging results. However, in the situation where the testing texts contain a lot of 'noise' ('full text' testing texts), TFDF outperform TF. In this case, considering the distribution of a feature in documents among categories is useful in helping us to identify discriminative keywords.

6. CONCLUSION AND FUTURE WORK

The quality of the classification task is mainly depends on two factors:

The inherent separation between categories selected. Clearly, even the best training and classification algorithm cannot achieve high accuracy if the categories are fuzzy and ambiguous. The categories selected should be mutually exclusive and 'well-separated' so that their intersection and overlapping is minimized.

The ability of the training process to capture the difference between categories from the training data. In order to generate a category model with reasonable size and high accuracy, the quality of training data is important. The training data should reasonably reflect the concepts of the category they belong.

This paper presents a classification algorithm that can be used to classify search results as well as web pages. We focus on feature selection method and term weighting function in order to generate a category model that consists of a set of optimum keywords which can fully describe the concepts of the categories. The results of our experiments are encouraging and shows that the classification algorithm proposed can potentially be used to classify search results returned by search engines.

In this paper, we only focus on linear classification. In linear classification, all categories are considered at the same level and the hierarchical structure of category hierarchy is not taken into account. As future research directions, we are interesting in developing a hierarchical classification algorithm that considers the hierarchical structure of categories. Intuitively, many potentially good keywords are not useful discriminators in non-hierarchical representation. We believe that considering the hierarchical structure of the categories can increase the accuracy of a classification task.

REFERENCES

1. C, Liu. (2004). A Survey: Automatic Text Categorization. *CS412 Report*, University of Illinois at Urbana-Champaign.
2. T, Joachims. (1997). A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. *In Proceedings of the 14th International Conference on Machine Learning (ICML97)*, pp143-151.
3. H, Chen. and S.T. Dumais. (2000). Bringing Order to the Web: Automatically Categorizing Search Results. *In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2000)*, pp145-152.
4. H, Chen., S.T. Dumais and E, Cytrell. (2001). Optimizing Search by Showing Results in Context, *In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2001)*. pp277-284.
5. H, Liu., H, Dash., and H, Motoda. (2000). Consistency Based Feature Selection, *In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2000)*

6. D, Mladenic. and M, Grobelnik. (1998). Word sequences as features in text-learning. *In Proceedings of ERK-98, the seventh Electro-technical and Computer Science Conference*, pp145-148.
7. M, Sahami. and D, Koller. (1996). Toward Optimal Feature Selection. In *Proceedings of the 13th International Conference on Machine Learning (ICML96)*, San Franscisco CA, Morgan Kaufmann, pp 284-292.
8. G, Salton. and C, Buckley. (1988). *Term Weighting Approaches in Automatic Text Retrieval.* In Technical Report, COR-87-881, Department of Computer Science, Cornell University.
9. S, S, Tan. (2002). *Topic Hierarchy Annotation using Feature Selection Technique.* MSc Thesis, School of Computer Science, Universiti Sains Malaysia.