A Hierarchical K-NN Classifier for Textual Data

Rehab Duwairi[1] and Rania Al-Zubaidi[2]
[1]Jordan University of Science and Technology, Jordan
[2]Jordan University of Science and Technology, Jordan

Abstract: *This paper presents a classifier that is based on a modified version of the well known K-Nearest Neighbors classifier (K-NN). The original K-NN classifier was adjusted to work with category representatives rather than training documents. Each category was represented by one document that was constructed by consulting all of its training documents and then applying feature selection so that only important terms remain. By this, when classifying a new document, it is required to be compared with category representatives and these are usually substantially fewer than training documents. This modified K-NN was experimented with in a hierarchical setting, i.e. when categories are represented as a hierarchy. Also, a new document similarity measure was proposed. It focuses on co-occurring or matching terms between a document and a category when calculating the similarity. This measure produces classification accuracy compared to the one obtained if the cosine, Jaccard or Dice similarity measures were used; yet it requires a much less time. The TrechTC-100 hierarchical dataset was used to evaluate the proposed classifier.*

Keywords: *Text categorization, hierarchical classifiers, K-NN, similarity measures, category representatives*.

Received October 23, 2008; accepted August 3, 2009

1. Introduction

In the last ten years, text classification as one of the most important problems in machine learning and data mining, has witnessed a thundering interest because of the increased availability of documents in electronic form and the resulting need to organize them [15, 19, 24]. Text classification is the task of assigning text documents to one or more predefined categories [16, 25, 27]. In general, text classification can be formally defined as [15, 24]: Assigning a Boolean value (T or F) to each pair $\{d_j, c_i\}$, where d_j is a document in the domain of the training documents (j =1, 2, …, *n*, where *n* is the number of training documents), and c_i is a category in the predefined set of categories (i = 1, 2, …, m, where m is the number of predefined categories). A value T indicates that the document d_j belongs to the category c_i, while a value F indicates that the document d_j does not belong to the category c_i. Text classification can be governed by different constraints. The case in which each document in the training set has to be classified to exactly one category is called *single-label*. While on the contrary, *multi-label* is the case where the same document may be assigned to any number of categories [24]. In text classification, most of the researchers have focused on flat classification. In flat classification, the predefined set of categories is treated independently of each other, and there is no structure defining the relationships among them [25, 27]. Nevertheless, it will become much more difficult to browse and search the predefined categories, when the number of these categories grows to a significantly large number. One way to solve the shortcomings of flat classification is to organize the categories into a hierarchy (tree-like structure) such that there are parent-child relationships between the predefined categories. Hierarchical classification allows a large classification problem to be addressed by using a divide-and-conquer approach [25, 27]. Directed acyclic category graph is identified as one of the most commonly used structures for text classification. In this structure, documents can be assigned to both internal and leaf categories [15, 25]. Each category –except the root– is labeled. The category in the hierarchy tree may be divided into multiple sub-categories. To determine the label of a new document using a hierarchical classifier, the document is assigned to the root category, and then the document classification can be repeated in each of the sub-categories until the document reaches some leaf categories or cannot be further classified into any sub-category. Hierarchical text classification helps users to find information more quickly and accurately, with large number of categories organized as a tree [16].

This paper introduces a new framework for hierarchical text classification. This framework is comprised of using a modified version of the K-Nearest Neighbor (K-NN) classifier and of introducing a new similarity measure that is based on the expected information value commonly used with the ID3 decision tree classifier. In the classical K-NN, a new document is assigned a label by comparing that document to every training example, then the K nearest neighbors of that test document are identified and their labels are recorded. The label of the new document equals the frequent label among the k nearest neighbors. When dealing with hierarchical classifiers or when dealing with large number of

documents, the classification time of K-NN degrades. This is because that every new document needs to be compared with every training example. The performance of the K-NN can be improved by indexing the document space so that a new document is compared only to a portion of the training documents. However, in this work, we chose to process the training documents that belong to a given category by identifying the most important features to that category and then by combining them into one document (called category representative). This fundamentally reduces the classification time as each category is represented by one document.

Cosine is the most commonly used similarity measure with K-NN when classifying documents [10, 17, 18]. Jaccard and Dice are also used with K-NN [12]. Calculating the similarity using these measures relies on vector manipulation where each document is represented as a vector of features that occur in the training documents together with their frequencies. In the proposed similarity measure (called new expected information value and denoted by I_{new}), only the shared terms between a test document and a category representative are considered when calculating the similarity. This results in smaller vectors. The proposed framework was tested against a TechTC-100 dataset [9]; and was evaluated in terms of classification times and classification accuracy (using precision and recall). The experiments reveal that the proposed classifier has a better classification time when using I_{new} compared to cosine, Jaccard and Dice. Also, precision and recall obtained using I_{new} were very close to the values obtained using the other three similarity measures.

The rest of this paper is organized as follows: section 2 presents related work. section 3, by comparison, describes the classification framework. Section 4 summarizes the properties of the dataset and describes the experiments and the results that were obtained. Finally, section 5 introduces the conclusions of this work and highlights future work.

2. Related Work

Pulijala and Gauch [21] explored the use of the hierarchical structure for classifying a large and heterogeneous collection of web content. They aimed to construct a hierarchy of classifiers that increases accuracy rather than to build a single massive classifier. Dumais and Chen [11] explore the classification for a large and heterogeneous collection of web content using hierarchical structure. They used Support Vector Machine (SVM) classifiers, which have not been previously explored in the context of hierarchical classification problems. SVM classifiers are efficient and effective learning scheme for flat text classification.

Peng and Choi [20] proposed an automatic web page classification algorithm, which uses the hierarchical structure to improve the classification accuracy. The algorithm assigns a web page to a category by searching through only one path of the tree structure. This single-path technique reduces the computational expense and increases the accuracy by 6% when compared to the search algorithms that are used by most existing classification algorithms.

Ceci and Malerba [5] addressed many aspects in hierarchical categorization. They studied feature extraction, the construction of the classifiers and the classification process. A novel technique for the selection of relevant features from training pages was proposed. Also, two classifiers have been considered and a threshold algorithm has been proposed in the case of a reject class. For the classification, they presented a graph search technique that explores all possible paths. Adami, Avesani, and Sona [1] proposed a semi-automatic process whose aim is to minimize the work required to the administrators when creating, modifying, and maintaining directories. Within this process, bootstrapping taxonomy with examples represents a critical factor for the effective exploitation of any supervised learning model.

Benkhalifa, Mouradi, and Bouyakhf [3] suggested, for text categorization, the integration of external WordNet lexical information to supplement training data for a semi-supervised clustering algorithm which uses a finite design set of labeled data to help agglomerative Algorithms Hierarchical Clustering (AHC) partition a finite set of unlabeled data and then terminates without the capacity to classify other objects.

Chakrabarti, Dom, Agrawal, and Raghavan [6] explored how to organize large text databases hierarchically by topic to aid better searching, browsing and filtering. They described an automatic system that starts with a small sample of the corpus in which topics have been assigned by hand, and then updates the database with new documents as the corpus grows, assigning topics to these new documents with high speed and accuracy. To do this, they used techniques from statistical pattern recognition to efficiently separate the feature words, or discriminants, from the noise words at each node of the taxonomy.

Chuang, Tiyyagura, Yang, and Giuffrida [7] presented an efficient algorithm for text classification using hierarchical classifiers based on a concept hierarchy. The simple TFIDF classifier is chosen to train sample data and to classify other new data. Despite its simplicity, results of experiments on Web pages and TV closed captions demonstrate high classification accuracy. D'Alessio, Murray, Schiaffino, and Kershenbaum [8] considered the problem of assigning documents to one or more categories from the point of view of a hierarchy with more or less depth. It could be chosen to make use of none, part or

all of the hierarchical structure to improve the categorization effectiveness and efficiency. Sun, Lim, and Ng [26] have found that the existing hierarchical classification experiments used a variety of measures to evaluate performance. These performance measures often assume independence between categories and do not consider documents misclassified into categories that are similar or not far from the correct categories in the category tree. They therefore, proposed new performance measures for hierarchical classification. The proposed performance measures consist of category similarity measures and distance based measures that consider the contributions of misclassified documents.

3. Classification Framework

The proposed classification framework is divided into three major modules: the preprocessor, the category representative extractor, and the K-NN classifier which are described next.

Preprocessing: preprocessing includes tokenizing where a document is divided into a set of tokens. It also includes stopwords removal where words such as *"the, a, not, nor, that"* are eliminated because they add nothing to the meaning of a document. To reduce the number of terms that appear in a given document, terms are replaced by their stems. Finally, preprocessing, in this paper, also includes selecting the most important terms by applying the TFIDF feature extraction method. For more details on the preprocessing that was used in this paper, please refer to section 4.1.

Category Representative Construction: each category in the hierarchy has been represented with one representative document, instead of dealing with many training documents. The features (words) for this single document are combinations of the most important features of its training documents. TFIDF has been used as a feature selection method. The representative document for a leaf category was created as described below:

- All positive documents of the leaf category were read and distinct terms were recorded with their frequencies in a file.
- To decrease the number of terms in the above file, feature selection using TFIDF was applied.
- The terms were sorted in descending order based on their TFIDF values.
- All terms that their TFIDF values are greater than a threshold were selected. The average value of TFIDF values was taken as a threshold which was 0.0002.

For a non leaf category, the category representative is created by merging the representative documents of its children and then using the terms that have TFIDF values greater than a threshold. The category representative saves time and effort; instead of dealing with j training documents in the i^{th} level (j >> number of categories in i^{th} level), we deal with q representative documents (q = number of categories in i^{th} level). With the increasing size of datasets used in text classification, the number and quality of features provided to describe the data has become a relevant and challenging problem. There is a need for effective feature reduction strategies [13, 28]. Most popular feature selection methods for text applications include Information Gain (IG), X^2–test (Chi-Square), Mutual Information (MI), Document Frequency (DF), Term Frequency Inverse Document Frequency (TFIDF) and Term Frequency Variance (TFV) [28].

TFIDF (which was used for feature selection in this research) is a weight often used in text mining and information retrieval and its variations are used by search engines to score and rank the relevance of a document when given a user query [23]. The number of times that a given term t_i occurs in such document is known as term frequency (n_{ti}). To prevent a bias towards longer documents, n_{ti} value is usually normalized by dividing the occurrences of the term on the length of that document (nd_j). So the normalized term frequency for term t_i in document d_j is: $TF_{ij} = n_{ti} / nd_j$. The importance of a term increases proportionally to the number of times that it appears in the document, but if this term appears frequently in many documents in the dataset, then its general importance decreases. The Inverse Document Frequency (IDF) is a measure that shows the general importance of the term in the whole collection. So the inverse document frequency for term t_i in all documents of the collection (N) is: $IDF_i = \log_2(N / D_{ti})$, where, D_{ti} is the number of documents such that t_i happens at least once. Then $TFIDF = TF \times IDF$

K-NN Classifier: one of the most popular classification techniques is K-NN. It was first introduced by the researchers E. Fix and J. Hodges in their paper, Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties, in 1951 [14]. It is a generalization of one nearest neighbor (1-NN) rule. The 1-NN is based on assigning to the non-labeled document, the label of the nearest (closest) labeled document, while K-NN considers the most frequent label in the K closest neighbors.

The labeled documents in the dataset are called training documents, and the new document that needs a label is called test document. Training documents are mapped into a multidimensional feature space, which is portioned according to the labels of the training documents. To label a test document, all neighbors equally contribute to determine the label for the test document [2, 4]. K-NN is a lazy algorithm, such that a storing of the feature vectors and the training documents' labels is only done in the training phase of

the algorithm. In the actual classification phase, the feature vector for the test document (whose label is not known) is computed using the same features space of the training documents. Also, in this phase, the label of the test document is predicted according to the frequent label within the set of K closest training documents.

The K parameter is very important and the optimal choice for its value depends upon many factors, and the most influencing one is the nature of the data [14]. Although larger values make boundaries between labels less distinct, they reduce the effect of noise on the classification process. The most popular technique for selecting good K is cross validation [4].

There are many measures used to assess the accuracy of the K-NN classifier. Precision and recall are the most popular measures. The K-NN algorithm is appropriate when dealing with a large set of training labeled documents and a small set of test non-labeled documents which have to be classified with the most suitable label. It is also, quite simple and easy to implement, but it suffers from large computing power requirements; since classifying any non-labeled document involves computing the distance between that document and every training document in the set. Cosine, Jaccard and Dice are similarity functions that are commonly used with the K-NN classifier [12]. Let a test document be represented as $t=(r_1, r_2, \ldots, r_n)$ and a category vector be represented as $c=(v_1, v_2, \ldots, v_n)$, n is the number of distinct terms in the dataset, then the cosine, Jaccard and Dice functions are calculated as:

$$cosine(t,c) = \frac{\sum_{i=1}^{n} r_i \times v_i}{\sqrt{\sum_{i=1}^{n} (r_i \times r_i)} \times \sqrt{\sum_{i=1}^{n} (v_i \times v_i)}}. \quad (1)$$

$$jaccard(t,c) = \frac{\sum_{i=1}^{n} r_i \times v_i}{\sum_{i=1}^{n} (r_i \times r_i) + \sum_{i=1}^{n} (v_i \times v_i) - \sum_{i=1}^{n} (r_i \times v_i)}.. \quad (2)$$

$$dice(t,c) = \frac{2\sum_{i=1}^{n} r_i \times v_i}{\sum_{i=1}^{n} (r_i \times r_i) + \sum_{i=1}^{n} (v_i \times v_i)}. \quad (3)$$

Cosine, Jaccard and Dice measures were used in this paper with the K-NN classifier. Each category representative in each level in the hierarchy is represented as a vector. Also, the test document is represented as a vector. To make the vectors product easy, each feature in the dataset is considered as a dimension for both vectors of the test document and each category representative. The values stored in these vectors are normalized term frequencies. A new similarity measure is introduced. It is called I_{new}. The idea of this new measure is built based on the Expected Information Value (denoted by I). This value is used in ID3, which is a version of decision tree induction classifier. Initially, the computation of expected information values was defined for structured datasets. i.e. a database of tuples; and each tuple contains values for attributes that are shared between all tuples. Thus, the original I value is calculated as: let s be the set of tuples in the database. Let m be the set of distinct classes C_i (for i=1,…,m). Let s_i be the number of tuples in class C_i.

$$I(s_1, s_2, \ldots, s_m) = -\Sigma p_i \log_2 (p_i),\ i=1,\ldots,m \quad (4)$$

where p_i is the probability that an arbitrary tuple belongs to class C_i and is estimated by $\frac{s_i}{s}$. Suppose that the tuples are replaced by documents, then I is calculated as: let s be the set of training documents. Let m be the set of categories C_i (for i=1,…,m). Let s_i be the number of training documents in category C_i.

$$I(s_1, s_2, \ldots, s_m) = -\Sigma p_i \log_2 (p_i),\ i=1,\ldots,m \quad (5)$$

where p_i is the probability of arbitrary training document i belongs to category C_i. And is estimated by $\frac{s_i}{s}$. Now, suppose that we want to define I at the term level; i.e. we want to compute the expected information value between two documents by comparing their terms' frequencies: let m be number of shared terms between the test document and the category representative. Let s be the total occurrences for the terms in that category. Let s_i be the frequency of a shared term i in the test document, then I_{new} is defined as:

$$I_{new}(s_1, s_2, \ldots, s_m) = -\sum \frac{s_i}{s} \times \log_2 (\frac{s_i}{s}),\ i=1,\ldots,m \quad (6)$$

I_{new} is used as a similarity measure by assigning the test document to the suitable category according to the following procedure: The test document is represented as a vector of the terms that also appear in the category currently under consideration (called shared terms); the frequency of these terms in the test document are recorded as frequencies of the shared terms; these values are normalized by that category's length. Finally I_{new} is calculated. This procedure is repeated for every category at a given level in the hierarchy. The test document is assigned to the category that has the highest I_{new} value.

I_{new} is suitable as a similarity measure because as the number of shared terms between a test document and a given category increases the value of I_{new} also increases. The values of I_{new} are nonnegative; expect for the case where the frequency of shared terms becomes greater than the sum of terms in a category representative. This is unusual as category representatives are constructed by counting the

frequencies of terms in all training documents that belong to that category. To compute the similarity between the test document and each category's representative using I_{new}, there is a need to build only one vector. This vector consists of the shared terms of the test document and a given category. Thus the size of this vector is much smaller than the size of the vectors that are built in the case of cosine, Jaccard and Dice. Recall that the latter vectors' dimensions equal the number of terms in the dataset. This explains why the time necessary for classifying documents using I_{new} is smaller than the time required when using cosine, Jaccard or Dice.

4. Experimentation and Result Analysis

4.1. Dataset Description

The TechTC-100 test collection [9] which is one of Technion repository of text categorization datasets and depends on the Open Directory Project (ODP) (http://dmoz.org) was used. The TechTC-100 test collection contains 100 labeled. Each dataset contains a pair of ODP categories with an average of 150 to 200 documents. The datasets are single-labeled, which means, every document belongs to exactly one category. Each category contains links to actual Internet sites, and each link cataloged in the ODP is used to obtain a small representative sample of the target Web site. To this end, the target site was crawled starting from the URL listed in the directory. A predefined number of Web pages were downloaded and concatenated into constructional document and then HTML markup was removed from that document.

These individual pages were referred to as sub-documents and their concatenation yields one document for the categorization task. The TechTC-100 test collection has concatenated up to 5 first pages crawled from each site. Finally, HTML documents are converted into plain text and organized as a dataset, which was presented in a simple XML-like format.

The TechTC-100 test collection is available as plain text format. In this format, each labeled dataset contains a pair of categories, which uniformly were called "positive" and "negative". Each category has an ASCII text file which contains the documents labeled with one category. Positive category has the file "all_pos.txt" and negative category has the file "all_neg.txt".

This dataset format which is discussed above is inefficient for this research work, thus some processing mechanisms were applied on the TechTC-100 dataset. As mentioned above, each category id is hyperlinked with the corresponding categories path in the ODP hierarchy. The researchers follow 200 hyperlinks to build the TechTC-100 test collection hierarchy. For example, 1622 refers to Top: Arts: Music: Bands and Artists: U. "all_pos.txt" file of this 1622 category belongs to U leaf category. 10341 refers to Top: Regional: North America: United States: Arkansas. "all_pos.txt" file of this 10341 category belongs to Arkansas leaf category. After completing the construction of the TechTC-100 collection hierarchy, the contents of categories which are not leaves and have documents were removed. Each file in each leaf category, whether it was "all_pos.txt" file, or "all_neg.txt" file was processed in the following manner:

- Decompose the one file to its constitutional units (documents).
- Remove tags and special symbols.
- Convert document letters to lowercase.
- Tokenize the document to words; the following symbols were recognized as word separators: new line, >, white space, tab, comma, semicolon, and colon.
- Remove stopwords.
- Stem the remaining keywords using the English Porter2 stemmer [22].
- Select the most important terms of each document using the TFIDF measure.

The size of the resulted dataset is 57.3 MB, distributed on 143 categories with 13083 documents. According to our research, the dataset has been designed to follow the *strong subsumption constraint*. This concept means that all documents which belong to a child also belong to its parent [26].

4.2. Experiments

To evaluate the proposed work, the researchers have run four experiments by constructing four test datasets. These test datasets were created by using the *holdout* method. This method holds over a certain amount of instances from the original dataset to play the role of testing and the remainder is used for training. Both training and testing data have to have the standard preprocessing (lexical analysis (tokenizing), elimination of stopwords, and word stemming), so the test data is taken out from the original dataset, after the preprocessing steps are done. Most classification research holds out one-third of the dataset to be the test data. Each test dataset for each experiment contains two folders. The first folder consists of the positive documents which their label is known, while the second folder represents the negative documents which their label is not similar to positive documents' label.

In our hierarchical text classification, there is a need to associate a classifier with each tree category; and test documents are classified using top-down level-based strategy, in which the classification process starts with assigning the document to the root level, and then determines the subtree that the document belongs to. This procedure is repeated until the document reaches a leaf category. It is obvious that the

label for positive and negative documents is a hierarchical one covers a path from the top until we reach to such leaf. The process of selecting test data labels was done randomly but many issues were taken into consideration. Our test datasets comes under leaves which belong to different levels in the hierarchy, and we took into account to follow different paths, the component's categories for every path have different numbers of siblings on different levels.

The first experiment (*Exp1*) dealt with 129 documents (89 positives and 40 negatives) which have <*Top: Recreation: Autos: Enthusiasts*> as a positive label, the second experiment (*Exp2*) dealt with 111 documents (77 positives and 34 negatives) which have <*Top: Sports: Football: RugbyLeague*> as a positive label, the third experiment (*Exp3*) dealt with 66 documents (34 positives and 32 negatives) which have <*Top: Regional: Asia: India: Gujarat*> as a positive label , while the fourth experiment (*Exp4*) dealt with 80 documents (40 positives and 40 negatives) which have <*Top: Society: Religion_and_Spirituality: Christianity: Broadcasting*> as a positive label. The negative label for the negative documents in each experiment was a mix of multiple labels except the positive label for that experiment. Preprocessing is orthogonal to the similarity functions used in this work. Each category in the hierarchy was represented as one document by choosing the most important features.

4.3 Classification Time

To compute the time necessary for giving the test documents labels, the four test datasets were passed to the cosine, Jaccard, Dice and I_{new} K-NN classifiers. In the case of cosine, Jaccard and Dice a frequency matrix for each test document has to be built. This matrix contains the TFIDF of words in the test document. Notice that the number of words in this matrix equals the number of words in the training space. The time necessary for constructing this matrix is referred to as "building frequency matrix".

Classification time, on the other hand, means the time necessary to calculate the similarity of a test document with categories' representatives and finding the most similar category. Thus, the time necessary to complete a classification task for a test document is the sum of the previous two time values. In the case of I_{new}, there is no need to build such a matrix. Tables 1, 2, and 3 show the time required to classify the test documents that were prepared earlier and denoted by *Exp1*, *Exp2*, *Exp3* and *Exp4* using the four similarity measures. As it can be seen from the tables, the proposed I_{new} measure outperforms the cosine, Jaccad and Dice even when the time of building the frequency matrix to test documents is not considered.

Table 1. The classification time for I_{new} vs. cosine.

Experiment	I_{new} Classification Time in Seconds	Cosine Time in Seconds		
		Building Frequency Matrix	Classification	Total
Exp1	610	12600	915	13515
Exp2	549	12600	844	13444
Exp3	427	12600	613	13213
Exp4	366	12600	557	13157

Table 2. The classification time for I_{new} vs. Jaccard.

Experiment	I_{new} Classification Time in Seconds	Jaccard Time in Seconds		
		Building Frequency Matrix	Classify-cation	Total
Exp1	610	12600	945	13545
Exp2	549	12600	865	13465
Exp3	427	12600	635	13235
Exp4	366	12600	575	13175

Table 3. The classification time for I_{new} vs. Dice.

Experiment	I_{new} Classification Time in Seconds	Dice Time in Seconds		
		Building Frequency Matrix	Classification	Total
Exp1	610	12600	890	13490
Exp2	549	12600	822	13422
Exp3	427	12600	593	13193
Exp4	366	12600	540	13140

4.4. Classification Accuracy

To evaluate the proposed I_{new} similarity measure, we used precision and recall. *Precision* is defined as the fraction of the retrieved documents which is relevant, while *recall* is defined as the fraction of the relevant documents which has been retrieved. Table 4 summarizes the values of precision and recall for the four test datasets when calculated using the four similarity measures under consideration. The precision obtained using I_{new} was the highest in *Exp1* and *Exp2*. However, the precision was highest using the cosine similarity function in *Exp3* and in *Exp4*. Recall values were the highest when calculated using I_{new} in *Exp1* and *Exp2*. The highest recall values for *Exp3* were achieved when using cosine and Dice functions. For *Exp4*, the highest recall value was obtained using the cosine function.

Table 4. Precision and Recall values for I_{new} vs. cosine, Jaccard and Dice.

		I_{new}	Cosine	Jaccard	Dice
Exp1	**Precision**	0.8437	0.7656	0.8125	0.8281
	Recall	0.6067	0.5505	0.5842	0.5955
Exp2	**Precision**	0.8181	0.7547	0.7777	0.7678
	Recall	0.5844	0.5194	0.5454	0.5584
Exp3	**Precision**	0.6538	0.72	0.6521	0.6666
	Recall	0.5	0.5294	0.4411	0.5294
Exp4	**Precision**	0.5833	0.6216	0.5714	0.5675
	Recall	0.525	0.575	0.5	0.525

There is a variance between the evaluation results of the I_{new} function and the cosine, Jaccard and Dice functions. For some cases, we find I_{new} is better than cosine, Jaccard, and Dice. For other experiments, we find the cosine Jaccard and Dice measures are better than I_{new}. The explanation of this variance can be found in literature studies [13]. Most of the research papers adopt the idea of dependency; the classification process accuracy is data dependent. According to our experiments, we can say that the accuracy of I_{new} classifier is better than the cosine, Jaccard, and Dice classifiers on average. The average value of precision for I_{new} is 0.725, while it is 0.715 for cosine, 0.703 for Jaccard, and 0.708 for Dice. More over, the average value of recall for I_{new} is 0.554, while it is 0.544 for cosine, 0.518 for Jaccard, and 0.552 for Dice.

5. Conclusions and Future Work

This paper addresses hierarchical classification because it increases the specificity and helps users to find information more quickly and accurately, with large number of categories organized as a tree [16, 27]. This paper also presents a modified K-NN classifier to decrease the costly computations of the classical K-NN classifier and to increase the accuracy of the classification task. Each category in the hierarchy was represented by one representative document where its features are a combination of the most important features in that category's training documents. Moreover, a new technique to compute the similarity was introduced; it is called (I_{new}). The frequency of each distinct term in a category representative was computed as it appeared in the test document. These frequencies were used to compute the proposed new measure which is based on the expected information measure that is used in decision tree induction classifiers. The test document was assigned to the category in i^{th} level where its I_{new} value was the highest. The process of classifying a new document using I_{new} K-NN classifier needs less time than using cosine, Jaccard, and Dice K-NN classifiers in all experiments. The computation of the classification using I_{new} K-NN classifier deals with only the frequency of shared features between the test document and the category representative with simple computational operation, while the computation of the classification using the cosine, Jaccard, and Dice K-NN classifiers deal with the frequency of all data space features and many multiplication operations during the vectors product (category representative vector and test data vector). In terms of precision and recall, the experiments showed a variance between the evaluation results of I_{new}, cosine, Jaccard and Dice. For some experiments, I_{new} performed better than cosine, Jaccard, and Dice. For other experiments, cosine, Jaccard, or Dice measures outperformed I_{new}. This is because classification accuracy is data dependent. On average, I_{new} is slightly better than the cosine, Jaccard, and Dice classifier. There is no large difference between the values of I_{new} and the other measures, but the values of I_{new} can be considered good; because they are very close to other well-known measures.

References

[1] Adami G., Avesani P., and Sona D., "Bootstrapping for Hierarchical Document Classification," *in Proceedings of the 12th ACM International Conference on Information and Knowledge Management (CIKM)*, New Orleans, USA, pp. 295-302, 2003.

[2] Bang S., Yang J., and Yang H., "Hierarchical Document Categorization with k-NN and Concept-Based Thesauri," *Information Processing and Management*, vol. 4, pp. 387-406, 2006.

[3] Benkhalifa M., Mouradi A., and Bouyakhf H., "Integrating WordNet Knowledge to Supplement Training Data in Semi-Supervised Agglomerative Hierarchical Clustering for Text Categorization," *International Journal of Intelligent Systems*, vol. 16, pp. 929-947, 2001.

[4] Brants T. and Stolle R., "Finding Similar Documents in Document Collections," *Proceedings of 3rd International Conference on Language Resources and Evaluation (LREC), Workshop on Using Semantics for Information Retrieval*, Las Palmas, Spain, 2002.

[5] Ceci M. and Malerba D., "Hierarchical Classification of HTML Documents with WebclassII," *in Proceedings of the 25th European Conference on Information Retrieval (ECIR'03)*, Pisa, Italy, pp. 57-72, 2003.

[6] Chakrabarti S., Dom B., Agrawal R., and Raghavan P., "Scalable Feature Selection, Classification and Signature Generation for Organizing Large Text Databases into Hierarchical Topic Taxonomies," *Journal of Very Large Databases*, vol. 7, pp. 163-178, 1998.

[7] Chuang W., Tiyyagura A., Yang J., and Giuffrida G., "A Fast Algorithm for Hierarchical Text Classification," *in Proceedings of 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, London, UK, pp. 409-418, 2000.

[8] D'Alessio S., Murray K., Schiaffino R., and Kershenbaum A., "The Effect of Using Hierarchical Classifiers in Text Categorization," *in Proceeding of the 6th RIAO International Conference*, April, Paris, France, 2000.

[9] Davidov D., Gabrilovich E., and Markovitch S., "Parameterized Generation of Labeled Datasets for Text Categorization Based on A Hierarchical Directory," *in Proceedings of the 27th Annual ACM SIGIR Conference*, pp. 250-257, Sheffield,

UK. Available online from http://techtc.cs.technion.ac.il/, 2004, (*Last Accessed June 25th, 2007*).

[10] Dumais S., Platt J., Heckerman D., and Sahami M., "Inductive Learning Algorithms and Representations for Text Categorization," *in Proceedings of the 7th International Conference on Information and Knowledge Management (CIK'98)*, Bethesda, USA, pp. 148-155, 1998.

[11] Dumais S. and Chen H., "Hierarchical classification of web content," *in Proceedings of 23rd ACM International Conference on Research and Development in Information Retrieval*, August, Athens, Greece, pp. 256-263, 2000.

[12] Dunham M., *Data Mining Introductory and Advanced Topics*, Prentice Hall, New Jersey, USA, 2003.

[13] Eyheramendy S. and Madigan D., "A Novel Feature Selection Score for Text Categorization," *in Proceedings of the Workshop on Feature Selection for Data Mining, in conjunction with the 2005 SIAM International Conference on Data Mining*, Newport Beach, CA, pp. 1-8, 2005.

[14] Fix E. and Hodges J., "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties," *International Statistical Review*, vol. 57, no. 3, pp. 238-247, doi:10.2307/1403797, 1989.

[15] Freitas A. and Carvalho A., "A Tutorial on Hierarchical Classification with Applications in Bioinformatics," *In: chapter VII, Research and Trends in Data Mining Technologies and Applications*, Idea Group Publisher, pp. 175-208, 2007.

[16] Han J. and Kamber M., *Data Mining Concepts and Techniques*, Morgan Kaufmann, San Francisco, CA 94104, USA, 2001.

[17] Iwayama M. and Tokunaga T., "Cluster-Based Text Categorization: A Comparison of Category Search Strategies," *in Proceedings of the 8th International Conference on Research and Development in Information Retrieval (SIGIR'95)*, Seattle, Washington, USA, pp. 273-280, 1995.

[18] Joachims T., "Text Categorization with Support Vector Machine: Learning with Many Relevant Features," *in Proceedings of European Conference on Machine Learning (ECML'98)*, pp. 137-142, Chemnitz, Denmark, 1998.

[19] Kwon O. and Lee J., "Text Categorization Based on K-Nearest-Neighbor Approach for Web Site Classification," *Information Processing and Management*, vol. 39, pp. 25-44, 2003.

[20] Peng X. and Choi B.**,** "Automatic Web Page Classification in A Dynamic and Hierarchical Way," *in Proceedings of the IEEE International Conference on Data Mining*, Maebashi City, Japan, pp. 386-393, 2002.

[21] Pulijala A. and Gauch S., "Hierarchical Text Classification," *in Proceedings of the International Conference on Cybernetics and Information Technologies (CITSA)*, Orlando, FL, USA, 2004.

[22] Porter M., "An Algorithm for Suffix Stripping," *Program*, vol. 14, no. 3, pp. 130-137, 1980.

[23] Salton G., Fox E., and Wu H., "Extended Boolean Information Retrieval," *Communications of the ACM*, vol. 26, no. 11, pp. 1022-1036, 1983.

[24] Sebastiani F., "Machine Learning in Automated Text Categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1-47, 2002.

[25] Sun A. and Lim E., "Hierarchical Text Classification and Evaluation," *in Proceedings of the IEEE International Conference on Data Mining*, Washington, DC, USA, pp. 521-528, 2001.

[26] Sun A., Lim E., and Ng W., "Performance Measurement Framework for Hierarchical Text Classification," *Journal of the American Society for Information Science and Technology*, vol. 54, pp. 1014-1028, 2003.

[27] Sun A., Lim E., and Ng W., "Hierarchical Text Classification Methods and their Specification," *Cooperative Internet Computing*, Kluwer Academic Publishers, pp. 236-256, 2003.

[28] Tang B., Shepherd M., Milios E., and Heywood M., "Comparing and Combining Dimension Reduction Techniques for Efficient Text Clustering," *in Proceedings of the Workshop on Feature Selection for Data Mining, in conjunction with the 2005 SIAM International Conference on Data Mining*, Newport Beach, CA, pp. 17-26, 2005.

Rehab Duwairi, received her BSc degree (1989) in computer science from Yarmouk University, Jordan; MSc and PhD degrees in computer science from the University of Wales, Cardiff, UK, in the years 1994 and 1997, respectively. In August 1997, she joined Jordan University of Science and Technology, where she is currently working as an associate professor of computer science. Dr. Duwairi acted as a department head for the Department of Computer Science (1998-2000) and as a vice dean of the College of Computer and Information Technology (2004-2006). She joined Qatar University team on September 2007 as an associate professor of computer science. Her research interests include object oriented databases, data mining, semantic integration of structured and unstructured data, and Arabic text categorization. She is a member of ACM.

Rania Al-Zubaidi, is a computer science teacher in a high school. She earned her MSc in 2007 and BSc in 2004 in computer science from Jordan University of Science and Technology (JUST). She worked as a teaching assistant for JUST for one semester in 2005. She also worked as a technical assistant at Middlebury College, USA in 2006. Her research interests focus on data mining and information retrieval areas.