Text Classification Combining Clustering and Hierarchical Approaches

By

Shankar Ranganathan

B. E. (Computer Science and Engineering)

University of Madras, Chennai, India

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

_____

Dr. Susan Gauch, Chairperson

_____

Dr. Perry Alexander, Committee Member

_____

Dr. David Andrews, Committee Member

Date Accepted _____

Abstract

The Internet presents a vast resource of information that continues to grow exponentially. Most of the present day search engines aid in locating relevant documents based on keyword matches. However, to provide the user with more relevant information, we need a system that also incorporates the conceptual framework of the queries. This is the goal of KeyConcept, a search engine that retrieves documents based on a combination of keyword and conceptual matching. An automatic classifier is used to determine the concepts to which new documents belong. Currently, the classifier is trained by selecting documents randomly from each concept's training set and it also ignores the hierarchical structure of the concept tree. In this thesis, we present a novel approach to select these training documents by using document clustering within the concepts. We also exploit hierarchical structure in which the concepts themselves are arranged. Combining these approaches to text classification, we achieve an improvement of 67% in accuracy over the existing system.

Contents

List of Charts

List of Figures

Chapter 1

Introduction

1.1 Motivation

The Web has experienced exponential growth since its creation. As the number of web pages grows, the task of finding relevant information becomes very complicated. Search engines often provide too many irrelevant results. This is mostly because of the fact that a single word might have multiple meanings [Krovetz 92]. For example, the query 'salsa' returns the same results to a person searching for a recipe as to one searching for details of a dance form. Thus current day search engines that match documents only based on keywords prove inaccurate.

To overcome this problem, the KeyConcept search engine [KeyConcept 03] takes into account both the keyword for which the user is searching and the meaning, or the *concept,* in which the user is interested. The conceptual arrangement of information can be found on the Internet in the form of directory services such as Yahoo! [Yahoo] or the Open Directory Project [ODP]. These arrange Web pages conceptually in a hierarchical browsing structure. While it is possible that a lower level concept may belong to more than one higher-level concept, in our research we consider the hierarchy as a classification tree. In this case, each concept is a child of at most one parent concept.

During indexing, KeyConcept uses an automatic classifier to assign newly arriving documents to one or more of the preexisting classes or concepts. Currently, KeyConcept uses a 'flat classifier' that does not take the hierarchical relationships between concepts into account. However, the most successful paradigm for organizing large amounts of information, making it comprehensible to people, is by categorizing the different documents according to their topic, where topics are organized in a hierarchy of increasing specificity [Koller 97]. By utilizing known hierarchical structure, the classification problem can be decomposed into a smaller set of problems corresponding to hierarchical splits in the tree. One first learns to distinguish among classes at the top level, and then the lower level distinctions are learned only within the appropriate top level of the tree. Each of these sub-problems can be solved much more accurately and efficiently as well [Dumais 00].

For any classifier, the performance will improve if the documents that are used to train the classifier are the best representatives of the categories. Clustering can be used to select the documents that best represents a category.

1.2 Goals

In this thesis, we investigate and evaluate text classification performance when we exploit hierarchical relationships between the categories by using top-down level-based hierarchical text classification. We also explore the use of clustering to select the documents that are best representatives of a category to train the classifier.

We present a novel approach by combining the above two techniques. We show that there is an improvement of 70% when hierarchical structure is taken into account along with clustering techniques to train the classifier.

1.3 Thesis Outline

This thesis consists of six chapters including the current one. Chapter 2 discusses work related to the topics that are involved in this thesis. Chapter 3 presents an overview of KeyConcept's architecture. Chapter 4 details our approach to the problem and the implementation of the system. We conduct rigorous experiments to validate our approach and analyze their results in Chapter 5. Chapter 6 focuses on the conclusions that can be drawn from our work and outlines ideas on possibilities for future work.

Chapter 2

Related Work

2.1 Text Classification

Text classification is the process of matching a document with the best possible concept(s) from a predefined set of concepts. Text classification is a two step process: training and classification. In the first step, training, the system is given a set of pre-classified documents. It uses these to learn the features that represent each of the concepts. In the classification phase, the classifier uses the knowledge that it has already gained in the training phase to assign a new document to one or more of the categories. [Schutze 95] has underlined the role of feature selection in document classification. They have shown improvement by using latent semantic indexing and optimal term selection to reduce the number of features.

Several methods for text classification have been developed differing in the way in which they compare the new document with the reference set. A comparison of these methods is presented in [Yang 99] and [Sebastiani 02]. An outline of each method is as follows:

- **K-Nearest Neighbor:** Documents and concepts are represented as vectors in a vector space whose dimensions represent the various keywords in the vocabulary. The categories for a new document are determined by calculating the angle

between the document vector and the category vector to identify the k-Nearest neighbors. The weights of the dimensions are, for text, calculated using tf*idf.

- **Linear Least Squares Fit:** This approach creates a multivariate regression from the set of training documents (represented in the vector space model) and their categories (represented as binary vector). By solving a linear least squares fit between these pairs of vectors, a matrix of word-category regression coefficients is obtained. This matrix can be multiplied by a new document vector to get its corresponding category vector.
- **Naïve Bayesian:** [Elkan] This approach uses the joint probabilities of words co-occurring in the category training set and the document to be classified to calculate the probability that the document belongs to each category. The document is assigned to the most probable category(ies). The naïve assumption in this method is the independence of all the joint probabilities.
- **Support Vector Machines:** This method represents every document as a vector and tries to find a boundary that achieves the best separation between the groups of vectors. The system is trained using positive and negative examples of each category and the boundaries between the categories are calculated. A new document is categorized by calculating its vector and determining the partition of the space to which the vector belongs.
- **Decision trees:** This kind of classifier builds a decision tree from the documents in the training set. Each branch defines a test on some attribute of the document. New documents are walked down the decision tree until the matching category is found.

- **Neural networks:** [Ruiz 99] In this method, a neural network is trained using the documents from the training set. Once it has been trained, it can be used to categorize new documents.

2.1.1 Hierarchical Text Classification

In 'flat text classification', categories are treated in isolation of each other and there is no structure defining the relationships among them. A single huge classifier is trained which categorizes each new document as belonging to one of the possible basic classes. In 'hierarchical text classification' [Sun 01] we address this large classification problem using a divide-and-conquer approach. [Koller 97] proposed an approach that utilizes the hierarchical topic structure to decompose the classification task into a set of simpler problems, one at each node in the classification tree. At each level in the category hierarchy, a document can be first classified into one or more sub-categories using some flat classification methods. We can use features from both the current level as well as its children to train this classifier.

The following are the motivations for taking hierarchical structure into account [D'Alessio 00]:

1. The flattened classifier loses the intuition that topics that are close to each other in hierarchy have more in common with each other, in general, than topics that are spatially far apart. These classifiers are computationally simple, but they lose accuracy because the

categories are treated independently and relationship among the categories is not exploited.

2. Text categorization in hierarchical setting provides an effective solution for dealing with very large problems. By treating problem hierarchically, the problem can be decomposed into several problems each involving a smaller number of categories. Moreover, decomposing a problem can lead to more accurate specialized classifiers.
Category structures for hierarchical classification can be classified into [Sun 03]:

i) *Virtual category tree*: In this category structure, categories are organized as a tree. Each category can belong to at most one parent category and documents can be assigned to leaf categories only.

ii) *Category tree*: This is an extension of virtual category tree that allows documents to be assigned into both internal categories and leaf categories.

iii) *Virtual directed acyclic category graph*: In this category structure, categories are organized as a Directed Acyclic Graph (DAG). Similar to a virtual category tree, documents can only be assigned to leaf categories.

iv) *Directed acyclic category graph*: This is perhaps the most commonly-used structure in the popular web directory services such as Yahoo! [Yahoo] and Open Directory Project [ODP]. Documents can be assigned to both internal and leaf categories.

There are two basic approaches to hierarchical classification, namely, big-bang approach and the top-down level-based approach [Sun 03]. In the big-bang approach, the classifier assigns a document to a class in one single step whereas in top-down level-based approach, the classification is accomplished with the cooperation of classifiers built at each level of the tree. The test document starts at the root of the tree and is compared to categories at the first level. The document is assigned to the best matching level-1 category and is then compared to all sub-categories of that category. This process continues until the document reaches a leaf or an internal category below which the document cannot be further classified. One of the obvious problems with top-down approach is that a misclassification at a parent class may force a document to be mis-routed before it can be classified into child classes.

[Labrou 99], [Sasaki 98] and [Wang 01] have worked on big bang approach. [D'Alessio 00], [Dumais 00] ,[Koller 97] and [Pulijala 03] have focused on top-down approach based on levels.

2.2 Ontologies

An *ontology* is a specification of a conceptualization. In other words, an ontology is an arrangement of concepts that represents a view of the world [Chaffee 2000] that can be used to structure information. Ontologies can be extracted from hierarchical collection of documents, such as those in Yahoo! [YAHOO] or the Open Directory Project [ODP]. The Open Directory is the most widely distributed database of Web content classified by

humans. It can be used as a freely available hierarchical arrangement of concepts and associated Web pages.

Online ontologies can be used as the targets for automatic classification. [Labrou 99] describes the use of the TellTale [Pearce 97] classifier to map new documents into the Yahoo! directory. This is done by first training the classifier with documents for each of the concepts and then finding the concept that has the best match with the new document. Furthermore, an ontology can be used to allow users to navigate and search the Web using their own hierarchy of concepts. The OBIWAN project [Chaffee 2000] [Zhu 99] accomplishes this by letting users define their own hierarchy of concepts, then mapping this personal ontology to a reference ontology.

[Liu 02] and [Pitkow 02] have used personalization to match relevant concepts as related to the user. Liu [Liu 02] approaches the problem of personalizing information retrieval according to user preferences by analyzing previous queries made by the user and developing a user profile. The user profile is then used to select the most appropriate categories for a user when a user makes a new query. A general profile for categories is also prepared using the ODP [ODP] structure. In this system, users have to select the relevant documents for a query and the relevant categories for each document too. This lays too much emphasis on the user's decision on whether the document/category is relevant or not. [Pitkow 02] uses *contextualization* to obtain relevant search results from an Internet search engine. The ODP structure is used here to identify the user's search

context. The user's past browsing history is also stored. This information is used to augment a query by adding relevant terms from the user's profile and current context.

2.3 Document clustering

Document clustering is a complementary research area to classification. Because it requires an existing set of classes with associated training data, text classification that is an example of supervised learning. Broadly speaking, in clustering we automatically group related documents into clusters to create classes. Since, automatic clustering does not require training data, it is an example of unsupervised learning.

There are many different clustering algorithms, but they fall into a few basic types [Manning 99]. One way to group the algorithms is by: hierarchical clusterings or the flat non-hierarchical clusterings. Software Packages such as Cluto [CLUTO] provide multiple classes of clustering algorithms to make comparison between approaches possible.

Flat clusterings simply create a certain number of clusters and the relationships between clusters are often undetermined. Most algorithms that produce flat clusterings are iterative. They start with a set of initial clusters and improve them by iterating a reallocation operation that reassigns objects. Non-hierarchical algorithms often start out with a partition based on randomly selected seeds (one seed per cluster), and then refine this initial partition. [Manning 99].

A hierarchical clustering algorithm produces a hierarchy of clusters with the usual interpretation that each node stands for a subclass of its mother's node.

[Dubes 88] [Kaufman 90] discuss about clustering algorithms in detail. There are two basic approaches to generating hierarchical clustering:

a) Agglomerative: Start with the points as individual clusters, and as merge similar clusters as you proceed.
b) Divisive: Start with a single, all inclusive cluster and divide this cluster until individual singleton clusters remain.

[Guha 98], [Guha 99] and [Karypis 99] have used agglomerative algorithms whereas [Zhao 02] uses a divisive approach.

Chapter 3

Overview of KeyConcept Architecture

KeyConcept retrieves documents based on a combination of keyword matches and concept matches. This is done by extending the traditional inverted index to incorporate mappings between concepts and documents. Our augmented inverted index contains both conceptual and keyword information, thus making retrieval based on a combination of concept and keywords possible.

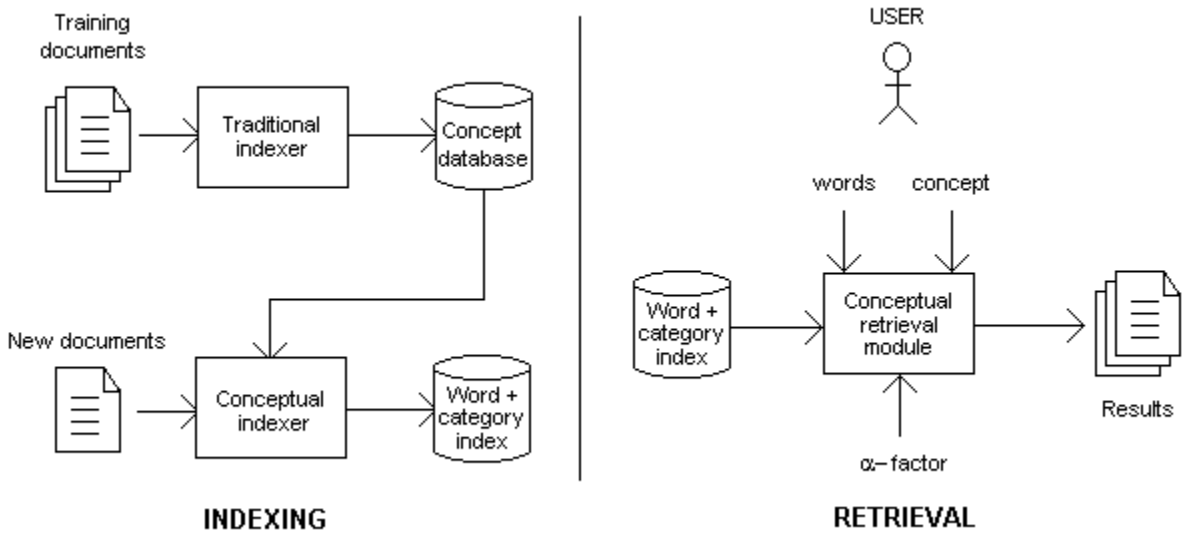

Figure 1. KeyConcept Architecture

3.1 Indexing

The indexing process is comprised of two phases [Gauch 04]: Classifier training and collection indexing. During classifier training, a fixed number of sample documents for each concept are collected and merged, and the resulting super-documents are preprocessed and indexed using the *tf * idf method*. This essentially represents each concept by the centroid of the training set for that concept. During collection indexing, new documents are indexed using a vector space method to create a traditional word-based index. Then, the document is classified by comparing the document vector to the centroid for each concept. The similarity values thus calculated are stored in the concept-based index.

3.2 Retrieval

During retrieval, the user provides a query containing words, concept identifiers, or both [Gauch 04]. After receiving user data, the search engine performs the search and stores the results for word and concept matches in separate accumulators. The final document scores are computed using the formula:

$$\text{Document score} = (\alpha \times \text{concept score}) + ((1-\alpha) \times \text{word score})$$

In our current online demo, we use a value of 0.3 for α, based upon the results of empirical experiments.

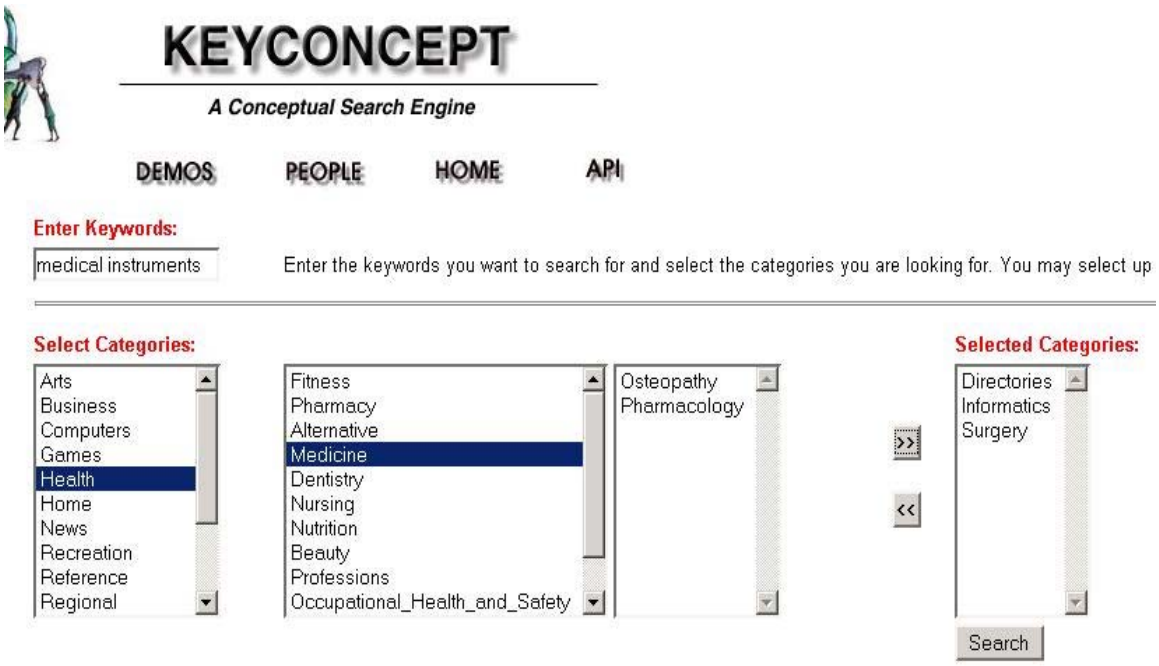

Figure 2. Keyword entry and concept selection in KeyConcept

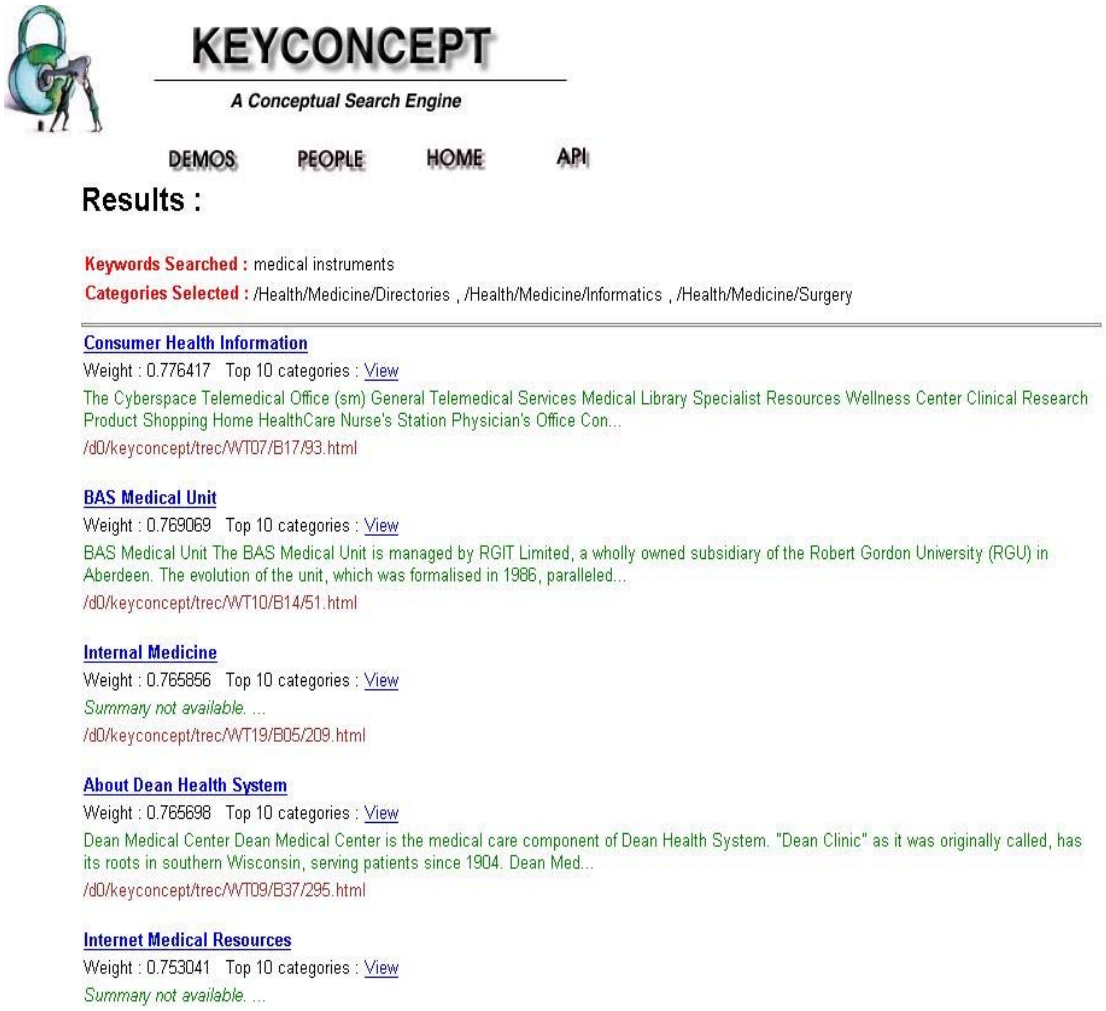

Figure 3. Search results after conceptual search

Since KeyConcept retrieves documents based on their categories, the more accurate the classification of the document, the more accurate are the search results. Our work is an attempt to improve the classification for KeyConcept.

Chapter 4

Implementation

4.1 Approach

4.1.1 Incorporating clustering

Feature selection for text classification plays a primary role towards improving the classification accuracy and computational efficiency. The Web has grown over such a wide range of topics that concept hierarchies built to organize the content are very large. With any large set of classes, the boundaries between the categories are fuzzy. The documents that are near the boundary line will add noise if used for training and confuse the classifier. Thus, we want to eliminate documents, and the words they contain, from the representative vector for the category. It is important for us to carefully choose the documents from each category on which the feature selection algorithms operate during training.

Hence, in order to train the classifier, we need to:

- Identify within-category clusters, and
- Extract the cluster(s)' representative pages

[Perkowitz 00] was the first to coin the phrase *cluster mining*. As opposed to the conventional use of clustering techniques to compute a partition for a complete set of data (documents (web) in our case), the aim of cluster mining is to identify only some representative clusters of Web pages within a Web structure. What we would like to do to

fulfill the two goals mentioned above is to use clustering techniques to get some kind of information about the arrangement of documents within each category space and select the best possible representative documents from those clusters. In essence, we are doing document mining within the framework of cluster mining.

4.1.2 Incorporating hierarchical classification

We train our classifier to recognize concepts in the Open Directory Project [ODP]'s category tree.

In a flat classification, the predefined categories are treated in isolation and there is no structure defining the relationships among them. [Sun 01]. Hierarchical classification was first proposed by [Koller 97]. The basic insight supporting hierarchical approach is that topics that are close to each other in the hierarchy typically have a lot more in common to each other than topics that are far apart [Koller 97]. Therefore, even when it is difficult to find the precise topic of a document, e.g., color printers, it may be easy to decide whether it is about 'agriculture' or about 'computers'. Building on this intuition, hierarchical classification approaches the problem with a divide and conquer strategy. In the above example, we have one classifier that classifies documents whether they belong to agriculture or computers. The task for further classifying within each of these wider categories is done by separate classifiers within agriculture or computers respectively. Each of these subtasks is simpler than the original task as the classifier at a node in the hierarchy need only distinguish between a small number of categories.

There are basically two approaches adopted by existing hierarchical classification methods, namely, the big-bang approach and the top-down level based approach. [Sun 01]. In the big bang approach, only a single classifier is used in the classification process. In the top-down level based approach, one or more classifiers are constructed at each category level and each classifier works as a flat classifier.

In our approach, we adopt a top-down level based approach that utilizes the hierarchical topic structure to decompose the classification task into a set of simpler problems. The classifiers we use are based on the vector space model.

We perform empirical evaluation using hierarchical structures to both select features and combine concept information from multiple levels of hierarchy. We first build a top-level classifier to discriminate among the level one categories. We conduct a series of experiments to get the best observed results by changing various parameters into account such as: the depth to which we need to traverse the tree, whether we need to take into account the category-wise break up of children concepts, etc. At the second level, we build classifiers within each cluster of categories. Each level two classifier thus concentrates on a smaller set of categories and is trained with more specific words that can distinguish between more specific documents. We thus proceed deeper into the tree.

4.2 System Architecture

We study and evaluate the performance of the text classifier when it is trained by using documents selected with the help of clustering from each category and by using a top-down, level-based, hierarchical approach of text classification. To do this, we need the following components:

- A system to perform document clustering within each category. We then choose the documents based on the result of clustering so that the documents that are best representatives of the category are selected.
- Automatic classifier(s) that will be trained for evaluation purposes. We will have one comprehensive classifier for flat classification and one classifier for each non-leaf node in the tree in case of hierarchical classification.
- A mechanism to test the classifier with documents that it has not seen before to evaluate its classification accuracy. We use the accuracy of classification as an evaluation measure.

4.2.1 A system to perform document clustering

The CLUTO Clustering Toolkit [CLUTO] - Release 2.1 is used to perform clustering within each category and conduct experiments to select documents that are best representatives of the category.

4.2.2 Classifier Training

To train the classifier [Madrid 00], content documents for each concept are concatenated to create a collection of super-documents, D. The super-documents are then pre-processed to remove high-frequency function words (stop words) and HTML tags. Finally, the Porter stemmer [Frakes 92] is used to reduce each word to its root to decrease the effect of word variations on the classification.

In a vector-space classifier, each concept *j* is represented by a vector, *cj*, containing one entry per term in the vocabulary. The weight for a term *i* in concept *j*, tc_{ij}, is a factor of the frequency of the term *i* in the super-document for the concept *j*, tf_{ij}, and the rarity of term *i* in other concepts, idf_i.

In more detail, the tc_{ij}, weight of term *i* concept *j* is given by:

$$tc_{ij} = tf_{ij} * idf_i \qquad (1)$$

where

D = *the collection of super-documents.*

t_i = i^{th} *term in vocabulary.*

d_j= *the* j^{th} *super-document.*

tf_{ij} = *number of occurrences of* t_i *in* d_j. (2)

idf_i =*Log (Number of documents in D /*

Number of documents in D that contain t_i) (3)

The dimensionality of the concept vectors is very large, one dimension for every word used in any document in the collection. This dimensionality is somewhat reduced by

removing stop words and further reduced by stemming. However, since most super-documents contain only a small fraction of the possible words, and absent terms receive a weight of 0, these concept vectors are very sparse. Because not all documents are the same length, the concepts vary somewhat in the amount of training data. To compensate for this, the term weights in each concept vector are normalized by the vector magnitude, creating unit length vectors.

Thus, ntc_{ij}, the normalized weight of term *i* concept *j* is given by:

$$ntc_{ij} = (tc_{ij} \ / \ vector\text{ - }length_j) \qquad (4)$$

where

$$vector\text{ - }length_j = \sum_i tc_{ij} \qquad (5)$$

4.2.3 Classification Phase (Testing)

Similar to the processing of the training documents, a term vector is generated for the document to be classified. This vector is compared with all the vectors in the training inverted index and the category vectors most similar to the document vector are the categories to which the document is assigned. The similarity between the vectors is determined by the cosine similarity measure, i.e., the inner product of the vectors. This gives a measure of the degree of similarity of the document with a particular category. The results are then sorted to identify the top matches. A detailed discussion on tuning the various parameters, such as number of tokens per document, number of categories per

document to be considered, etc., to improve the performance of the categorizer can be found in [Gauch 04].

Chapter 5

Experimental Observations and Results

5.1 Experimental Set up

Source of Training Data:

Because Open Directory Project hierarchy [ODP 02] is readily available for download from their web site in a compact format, it was chosen as the source for classification tree. It is becoming a widely used, informal standard. As of April 2002, the Open Directory had more than 378,000 concepts. With such a fine granularity, subtle differences between certain classes may be apparent to a human but indistinguishable to a classification algorithm. In our research with hierarchical text classification, the top few levels of the tree are sufficient. We decided to classify documents into classes from the top three levels only, although training data from the top four levels was used in some experiments.

Test Data:

We tested the accuracy of the classifier by classifying randomly-selected level 3 documents. These documents were excluded from the training process and were selected from different level 3 classes. Since we know the class from which the document was selected, we compare the accuracy of our classifier against "truth" by evaluating how often the classifier assigns the test documents to the classes from which they originally came.

5.2 Experiment 0 – Determining the baseline

Currently, KeyConcept uses a flat classifier. Documents are randomly selected from the categories. To evaluate our experiments, we must first establish a baseline level of performance with the existing classifier.

Experimental Set-up: The classifier was built from all level 3 or above concepts that had at least 32 documents. From each concept, we used randomly selected 2 documents for testing and also randomly selected 30 documents for training the classifier. The training set thus consisted of 1484 concepts and approximately 44500 training documents. Based on [Gauch 04], we trained with 30 documents per concept.

We tested with around 1484 * 2 = 2978 test documents. Based on [Gauch 04], we extracted the top 20 words from each document to act as the document surrogate during classification.

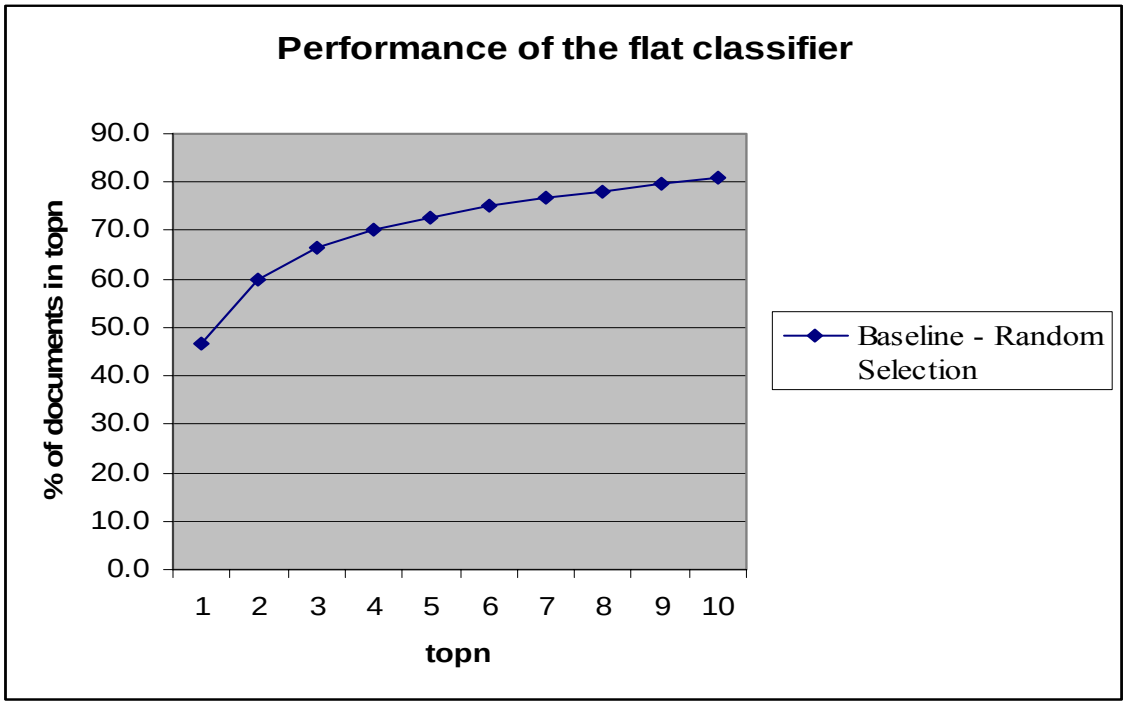

Chart 1: Baseline. Performance of the Flat Classifier when it is trained using 30 documents randomly selected from each concept.

Chart 1 provides us with the baseline with which we can compare our future work. It shows the percentage of documents within the top n (n =1,2,..10) concepts plotted against the rank n. 46.6% of the documents are correctly classified as belonging to the their true category and the correct answer appears within the top 10 selections over 80% of the time.

5.3 Experiment 1 – Effect of Clustering on Flat Classification

Hypothesis: Rather than randomly selecting documents from each concept to train the classifier, the performance of the classifier will improve if we select the documents that are best representative of the concepts. We can use clustering technique to identify the documents that could best represent the concepts.

Experimental Set-up: In each of the experiments, 2 documents per concept were randomly withheld for testing. The clustering algorithm was applied to rest of the concept's associated documents to select 30 documents for training using the different approaches below. After training, the categorizer was tested with test documents which were withheld for the purpose of testing.

As mentioned in Chapter 4, we used Cluto clustering toolkit [CLUTO] to perform the task of clustering and selection of documents. The following are the important clustering parameters that were used:

- **Clustering Method**: Partitional Clustering - using bisections.
- **Similarity Function:** Cosine Function
- **Particular clustering criterion function used in finding cluster**: I_2

where, I_2 is given by :

$$\text{maximize} \sum_{i=1}^{k} \sqrt{\sum_{v,u \in S_i} \text{sim}(v,u)}$$

In the above equation, *k* is the total number of clusters, *S* is the total objects to be clustered, *Si* is the set of objects assigned to the *i* th cluster, *ni* is the number of objects in the *i* th cluster, *v* and *u* represent two objects, and sim*(v, u)* is the similarity between two objects.

- ***vcluster*** [CLUTO] function was used for clustering and ***z-scores*** [CLUTO] was turned on to observe distances from centroid.

The primary goal of this experiment is to find the best observed way of selecting the most representative documents from each concept. These documents, when used to train the classifier, should enhance the effectiveness of the classifier when classifying unknown documents.

We try the following approaches for training document selection:

- **Experiment 1.1:** We choose 30 documents that are closest to the centroid from each concept to train the classifier.

- **Experiment 1.2:** We choose 30 documents that are farthest from the centroid in each concept to train the classifier.

- **Experiment 1.3:** We choose 30 documents that are farthest to each other in each concept to train the classifier.

- **Experiment 1.4:** Two clusters are identified within each concept and the 30 documents that are closest to their respective centroids are selected from these two clusters, normalized by the size of each cluster.

- **Experiment 1.5:** Three clusters are identified within each concept and the 30 documents that are closest to their respective centroids are selected from these three clusters, normalized by the size of each cluster.

- **Experiment 1.6:** We first discard a proportion of documents that are farthest from the centroid. Then, we identify two clusters in the remaining documents. We pick 30 documents from these two clusters normalized by their size. The basic intuition for this experiment is as follows: As the experiments were conducted incrementally from 1.1 above, we observed that the documents closest to the centroid yield better results, where as those farther away lower the accuracy. Hence, we wanted to study the effect selection from more than one clusters, after discarding farthest documents from the centroid.

  This was accomplished as follows:

  No. of docts to discarded, n = (Total no. of docts in the concept – 30) / 2

After discarding the farthest *n* documents from the centroid as above, two clusters are identified from the remaining documents and documents that are closest to the centroid are chosen from within these clusters normalized by the cluster size.

Results:

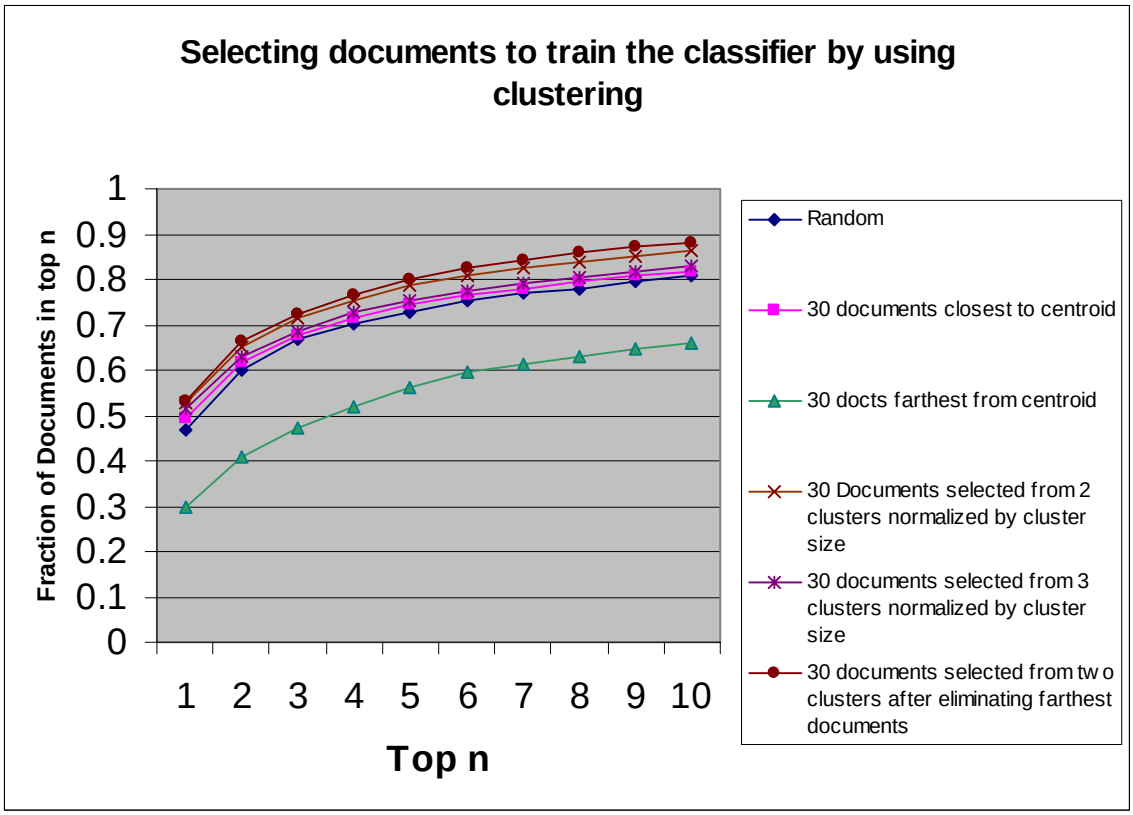

Chart 2: Using within-category clustering to select the documents to train the flat classifier.

Discussion:

Chart 2 shows the comparison of the results obtained from the six experiments mentioned above.

It is clear from the graph above that experiment 1.2, which is selecting documents that are farthest from the centroid, yields the poorest of the results. The percentage of exact matches in this is just 29.6%. This denotes a fall of 36% as compared to our random baseline of 46.6%. Experiment 1.1, which involves selecting documents closest to the centroid from each concept, gives us 49.5% of exact matches. This translates to an improvement of 3% in exact terms over random training. In experiment 1.3 we train the classifier with 30 documents that are farthest from each other. The results of this experiment show that the percentage of exact matches is 48.6%, an improvement of 2% over our baseline.

The accuracy of the classifier is 51.6%, 52.2% and 52.9% for experiments 1.4, 1.5 and 1.6 respectively. The best observed performance among these 6 trials is for experiment 1.6, selecting from two clusters after discarding outliers, which shows an improvement of 6.3% over baseline.

Conclusion:

There is a 3% improvement when we use documents that are closest to the centroid to train the classifier as against randomly selecting documents. This is the simplest of all the approaches. Although there is an improvement of 6.3%, the algorithm used in experiment

1.6 requires us to perform clustering thrice within the category and for huge sets of data proves to be inefficient. From the end user's perspective, an improvement of 3% might not be observable. Thus, we decide to stick to the closest to the centroid approach for the rest of our work. We can also conclude that, use of clustering only results in slight improvement in case of flat classification.

5.4 Experiment 2: Effect of clustering on training set selection for hierarchical classification

From experiment 1, we found that the use of clustering to select the training documents results in a small improvement in the accuracy of a flat classifier. Experiments by [Pulijala 03] on the same data show that exploiting hierarchical structure of concepts improves the classification accuracy for randomly selected training documents. In the next set of experiments, we wish to investigate whether or not using clustering to choose the training set for a hierarchical classifier results in further improvement.

We try to answer the following questions for our hierarchical classifier:

1. Does selecting documents closest to the centroid improve the classifier when compared with selecting documents randomly?
2. a. Does hierarchical classification show an improvement over a flat classifier when used in conjunction with clustering approach?

   b. If yes, how far down the hierarchy should we go to select training documents to get the best observed results?
3. What is the appropriate number of documents to train the classifier at each level so that we get the best observed result?

Experimental Set-up:

For the hierarchical classifier, we constructed a set of classifiers, one at each level of the classification tree, using the training method described above. Thus, there was one classifier for level 1, 15 classifiers for level 2, and around 358 classifiers for level 3. We test our classifier with 1000 documents that are randomly selected from level 3 concepts. We first classify each test document using the level 1 classifier and then, based on the top result, reclassify the document using the appropriate level 2 classifier to find the appropriate level 2 match. After a final classification for the best match at level 3, the final level 3 class is assigned to the test document was recorded.

We conducted various experiments by selecting training documents from the each concepts associated documents as well as documents from children and grandchildren concepts. If the documents were selected from children and/or grandchildren concepts, the tree structure was ignored and all documents from children or grandchildren concepts were put in the same pool from which training documents were selected. In section 5.5, we describe another set of experiments in which the concept structure is retained during training set selection. We compare the effect of clustering on training set selection as opposed to random selection in each of these experiments.

[CLUTO] was used for clustering. We used the program *vcluster* that takes as input the actual multi-dimensional representation of the documents that need to be clustered (*i.e.*, "v" comes from *vector*), and the number of clusters that we need to create. In order to observe the distances from centroid within each concept, the number of clusters as

specified as one, and the *z-score* parameter was turned on. The cosine function was used to compute similarity measure.

Experiment 2.1: Study of Level 1 Decision

In the following experiments, we first concentrate on the crucial level 1 decision and try to maximize the number of documents that make this decision correctly.

The goal of these experiments is to:

i. Compare the performance of the classifier when it is trained by documents that are selected randomly as against documents that are selected by clustering, that is, the documents closest to the centroid.

ii. Determine the number of documents required to train the classifier to get the best results for level 1 decision, and,

iii. Determine how far down the hierarchy we need to go when selecting training documents in order to obtain the highest accuracy for the level 1 decision.

Hypothesis 1: For the level 1 decision, it is enough if we train the classifier using just level 1 documents and selecting the documents that are closest to the centroid improves the performance of the classifier.

The underlying assumption is that by going further down into the hierarchy, we broaden the feature set of concepts at the top level. This makes it difficult for the classifier to distinguish between these concepts. Since we are interested only in the level 1 decision, documents at level 1 should be able to provide us with the necessary information.

Experiment 2.1.1: The classifier is trained using documents from the concepts at level 1 only.

Results:

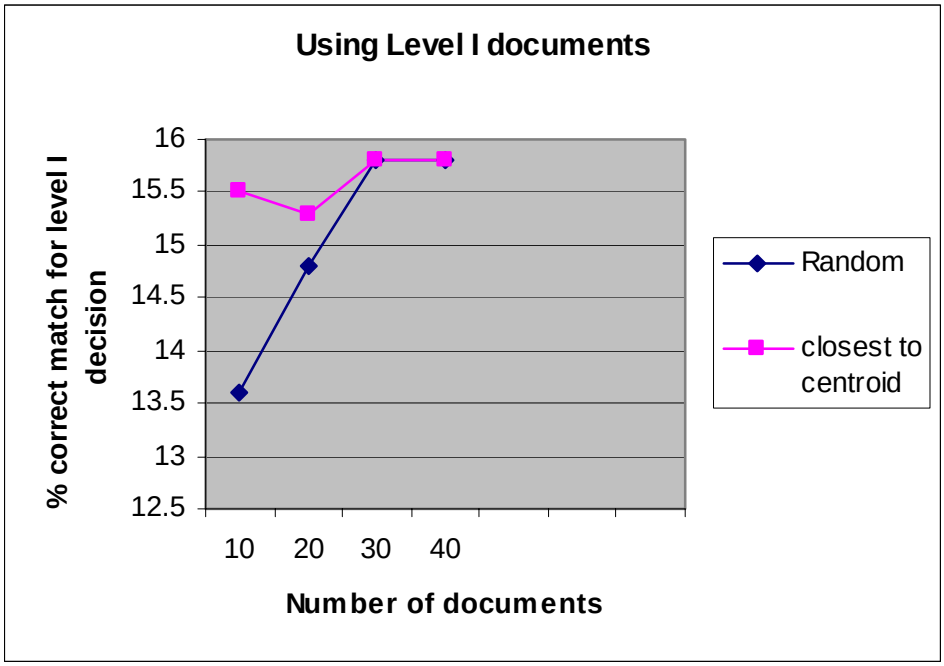

Chart 3: Level 1 decision using only documents from level 1

Discussion: From Chart 3 we see that we get the highest exact match accuracy of 15.8% for 30 documents. This poor performance is because there is very little training data at Level 1. A majority of the concepts do not have any documents at all. There are no concepts that contain a large number of documents (>30). However, we find that even in this case, choosing the documents that are closest to the centroid yields a marginally better result, 15.5% versus 13.6% for 10 documents.

15.8% accuracy is a poor result for level 1 decision. As we go down the tree for further decisions at the lower levels, the accuracy will monotonically decrease yielding a very poor final result. Therefore, we need to investigate augmenting the training set with documents from further down the hierarchy and use documents from lower levels to train our classifier for the level 1 decision.

Hypothesis 2: The accuracy of the classifier at level one can be improved by training the classifier using the documents at the current level (level 1) and its children (level 2). The performance of the classifier will improve if clustering is used to determine the documents that are closest to the centroid for training the classifier.

The above hypothesis is formed based on the previous observations of insufficient data at level 1 and better performance of the classifier if the quality of the training documents is improved by choosing the documents closest to the centroid.

Experiments 2.1.2:

In both the experiments below, the documents from the parent and its children concepts are put in the same bucket to form the training set.

a) Classifier is trained by selecting documents randomly from the training set.
b) The training set is clustered and documents closest to the centroid are selected to train the classifier.

The number of documents that are selected to train the classifier for each concept is varied from 10 to 90 in each of the cases above to find the best observed number of documents required to train the classifier.

Results:

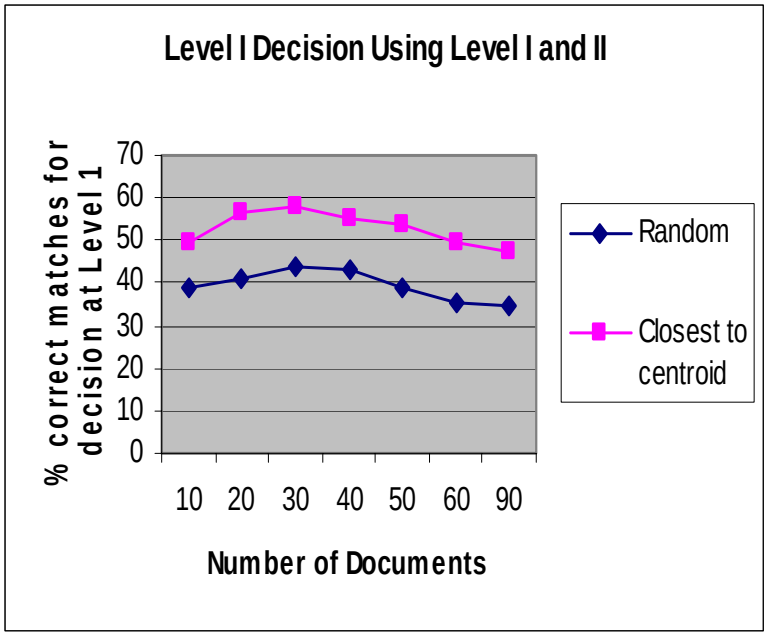

Chart 4: Training Level 1 classifier using documents from both level 1 and level 2 (ignoring tree structure at level 2).

Discussion: It is observed from chart 4 that, compared to using level 1 training data only, there is an increase in the accuracy from 15.8% to 43.5% in case of random selection. Once again, 30 documents seem to be best for training when clustering is used.

The jump in the accuracy of the classifier from experiment 1 to experiment 2 validates the use of documents from level 2. There is a further improvement of the classifier's accuracy when using of clustering to identify documents closest to the centroid to train the classifier.

Hypothesis 2: The accuracy of the classifier at level one can be improved by training the classifier using the documents at the current level (level 1) and its grandchildren (level 3) and its children(level 2). The performance of the classifier will improve if clustering is used to determine the documents that are closest to the centroid for training the classifier.

The motivation for the above hypothesis is from the results of experiments 2.1.2. We observe in experiment 2.1.2 that there is a significant improvement both in case of using the children concepts and also, when we use clustering to select the documents that are closest to the centroid from the training set.

Experiment 2.1.3:

We pooled training documents for each concept from that concept, its children at level 2 and grandchildren at level 3 to create the training set.

a) Classifier is trained by selecting documents randomly each concept's training set.
b) Documents closest to the centroid are selected to train the classifier after clustering.

The number of documents that are selected to train for each concept is varied from 10 to 90 in each of the cases above to find the optimum number of documents required to train the classifier.

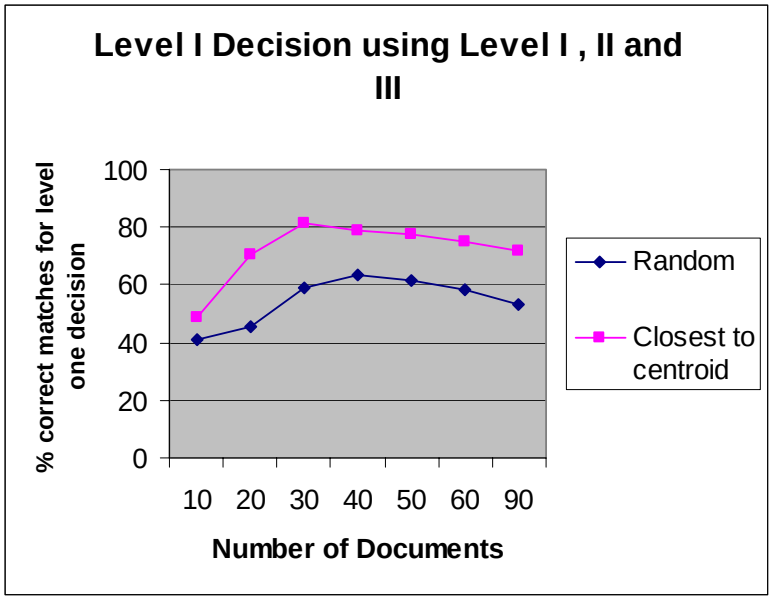

Chart 5: Training Level 1 classifier using documents from levels 1, 2 and 3 (ignoring tree structure at levels 2 and 3).

Discussion: In chart 5, The highest accuracy for randomly selected documents is 63.6%, an improvement of 20% over random selection using training documents just levels 1 and 2. There is a further improvement (81.6%) when we use clustering to select the documents that are closest to the centroid.

The result validates our hypothesis that adding training documents from level 3 improves accuracy and that the use of clustering to select documents that are closest to the centroid enhances classifier's performance.

From the above experiments, it is clear that we get the best results for level 1 decision when we use documents from all the levels 1, 2, and 3 and we select the 30 documents closest to the centroid from each category to train the classifier.

Experiment 2.2 Improving the level 2 decision

Having improved our level 1 decision from 15.8% accuracy to 81.6% accuracy, we now concentrate on the level 2 decision.

Hypothesis 1: It is enough if we train the classifier using only documents at the current level. Use of clustering to selecting documents closest to the centroid yields better results than random selection of documents to train the classifier.

Experiment 2.2.1: We train the classifier using documents from level 2 only. We vary the number of documents from each category to determine the number of documents that results in the highest accuracy.

Results:

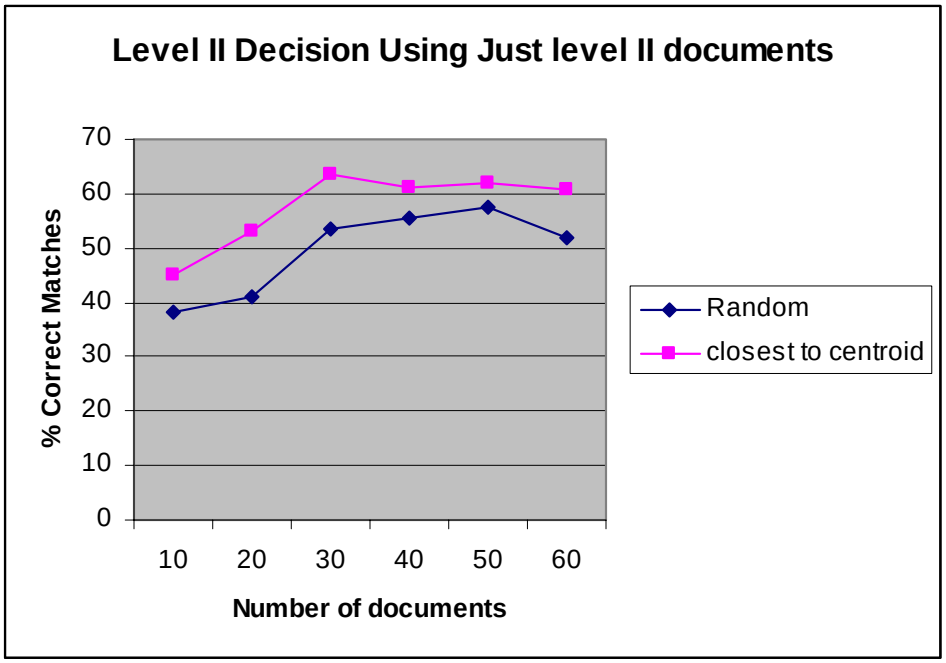

Chart 6: Training Level 2 classifiers using documents from only level 2

Discussion: It is observed from chart 6 that in case of random selection of documents, the best results (57.4%) are observed when we use 50 documents to train the classifier for each concept. However, we are able to get higher accuracy (61.2%) when we use 30 documents from each category that are closest to the centroid. Lower results associated with 10, 20 documents can be attributed to the fact that we are not able to get enough words to accurately classify test documents. However, as we increase the number of documents, it first peaks and then the graph plateaus, indicating that 30 good documents contain sufficient content to adequately train the classifier. There is an improvement when we use clustering to select the training documents that are closest to the centroid as against randomly selected documents. However, the number of documents that are correctly classified at level 2 by just using level 2 data falls sharply when compared to the level 1 results. The best result of 61.2 % shows a 20% drop in performance in absolute terms and 25% in relative terms over that provided by the level 1 classifier. This allows us to speculate that we may get better results if we train the level 2 classifiers with documents selected from lower level concepts.

Hypothesis 2: The accuracy of the classifier at level 2 will improve if we use documents from the parent (level2) as well as its children (level 3).

Experiment 2.2.2: We train the classifier using training data from both level 2 and level 3. The data from the parent as well as its children concepts are put in the same bucket. Then the training documents are chosen either randomly, or those closest to the centroid.

The number of documents that are chosen is varied from 10 to 60 to determine the necessary number of training documents for accurate classification.

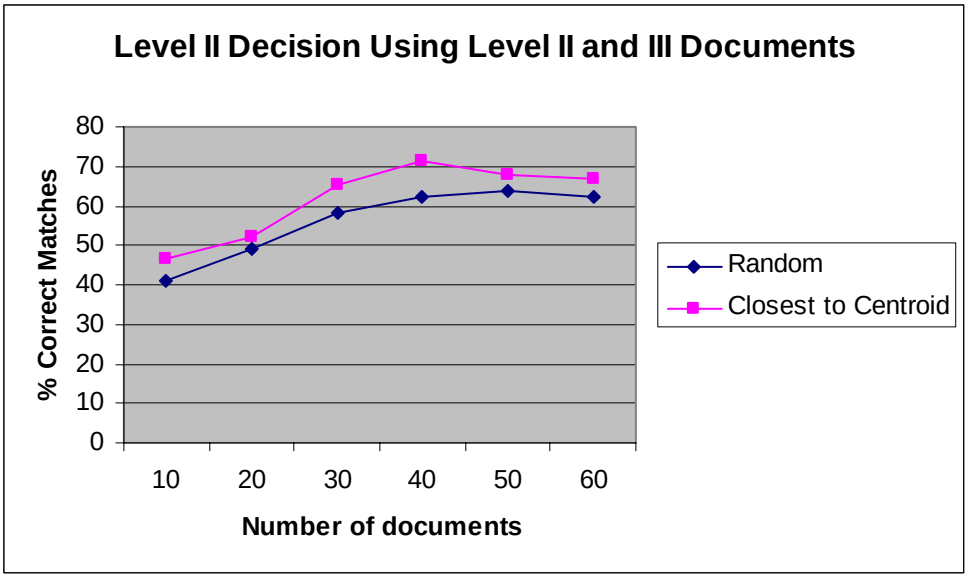

Chart 7: Training the level 2 classifiers using documents from both levels 2 and 3

Discussion: It is observed from Chart 7 that the maximum accuracy of the classifier is found to be 64% for 50 documents when the selection is random. When we choose the documents closest to the centroid to train the classifier, the best performance of 71.3% occurs with 40 documents. There is a drop of 10% absolute and 12% relative over level 1 accuracy. This is an improvement over the results base don level 2 data alone. We also find that there is an improvement of roughly 7% when clustering is used to identify the documents. This result validates the use of clustering to select documents for the level 2 decision.

Thus, we can conclude that we get the best results for level 2 decision when we use documents from both level 2 as well as level 3 to train the classifier. This selection is done by putting parent and child concepts in the same bucket and choosing 40 documents closest to the centroid.

Experiment 2.3: Study of Level 3 decision

At the end of the experiments for level 2 decision, out of 1000 initial test documents 713 have correctly identified their level 2 concept. Since all the test documents are drawn from level 3, the last step is to measure how many documents make it correctly to their true concepts. As in the previous cases, we try to determine the best possible way to train the classifier to maximize the number of documents reaching their true destinations.

Hypothesis: Using clustering to select the documents that are closest to the centroid will help us improve the performance of the classifier when we use level 3 data for the level 3 decision.

Experiment: We use documents from level 3 categories to train the classifier. We repeat the experiment by varying the number of documents to find the necessary number of documents to train the classifier for highest accuracy. The same experiment is tried by randomly selecting documents and also selecting documents closest to the centroid.

Results:

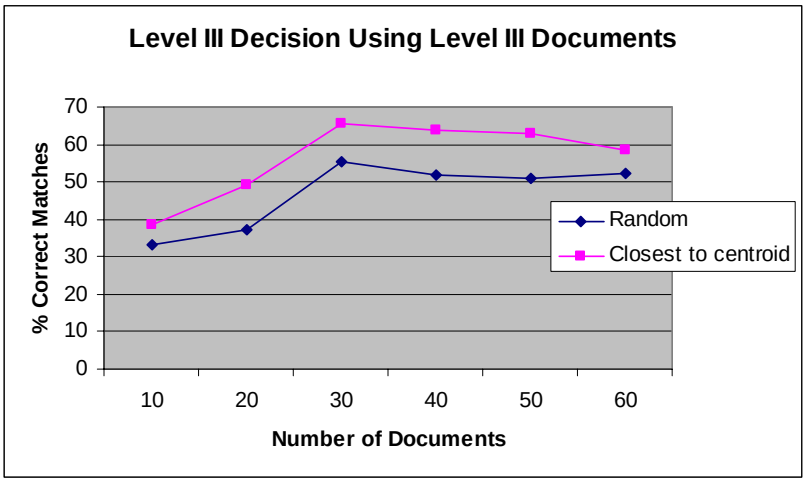

Chart 8: Performance of level 3 classifiers

Discussion: Out of the 713 documents that made it to the level 2 correctly in the best case scenario, we get the best results of 552 make it to the correct concept when the classifier is trained with randomly selected documents versus 654 when the classifier is trained with documents closest to the centroid. This is improvement of 10.2% (absolute) and 18.4% (relative) when the documents that are chosen to train are closest to the centroid. Both peaks occur when we use 30 documents to train each concept. The documents that are closest to the centroid happen to be the best representatives of the knowledge of that concept. As the number of documents is increased we find that there is no noticeable increase in the performance, this can be attributed to the fact that with the increase in the number of documents, more noise is included to train the classifier and the classifier looses its focus. When the number of documents is 10 or 20, the accuracy is lower because of insufficient information.

We can conclude that with our current approach 65.4% of the documents reach their true concepts versus 46.6% with the flat classifier. This represents a 18.8% (absolute), or 40.3% (relative) improvement in the accuracy of the classifier.

Summarizing the above results; we achieve the highest accuracy of 65.4% when:

1. For the level 1 decision, the classifier is trained using documents from levels 1, 2 and 3. The documents are selected by putting the parent, child, and grandchild training documents in the same bucket and selecting 30 documents closest to the centroid for each individual concept.

2. For the level 2 decision, the classifiers are trained using documents from levels 2 and 3. The documents are selected by putting the parent and child training documents in the same bucket and selecting 40 documents closest to the centroid for each concept.

3. For level the 3 decision, the classifiers are trained using documents from levels 3. The 30 documents closest to the centroid are selected for each concept.

5.5 Experiment 3: Effect of clustering on training set selection for hierarchical classification, distributing the training set across sub-concepts

Hypothesis: The accuracy of the classifier can be improved by training the classifier using documents from each concept. A minimum number of training documents should be selected from each sub-concept, and these documents should be closest to the centroid.

The reasoning for this assumption is that the reduction training documents for each concept should be representative of the breadth of the concept. So, each concept should be trained on documents selected from each sub-concept. The hypothesis is inspired from the earlier work done by [Pulijala 03] who achieved accuracy of 79% at level 1 by selecting a single random document from each sub-concept down to level 3. We extend that idea further using clustering to identify the centroid of each category. This document should represent the category better than a randomly selected document. Further, we investigate whether selecting more than one documents closest to the centroid improves the accuracy of the classifier.

Experiment 3.1 : For the crucial level 1 decision, we conduct 4 different experiments using documents from just level 1, levels 1 and 2, levels 1, 2 and 3, and levels 1, 2, 3 and 4 to train the classifier. In each of the above experiments, we identify the number of documents required to train the classifier from each category such that we get the highest accuracy.

Results:

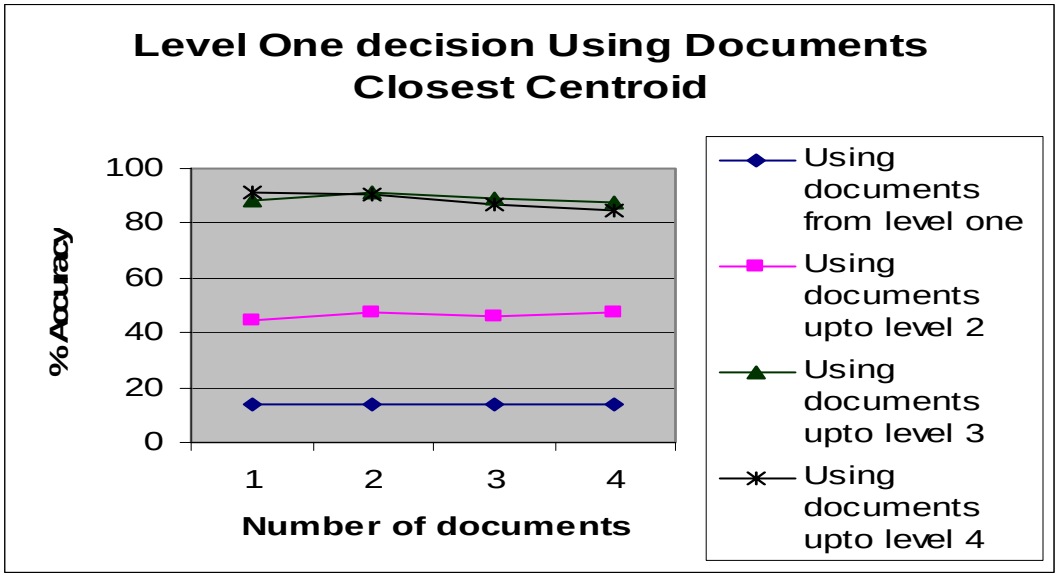

Chart 9: Level one decision using documents closest to the centroid from each concept

Discussion: From Chart 9, we find that 14.1 % of the documents are correctly classified at level one if we use just documents from level 1. This is because we have too little data available at this level. The result improves to 47.4 % when we also include documents from level 2. We find a sharp improvement in accuracy when the documents from level 3 are included. The highest accuracy of the classifier for level 1 decision, 91.2%, is obtained when we use 2 documents closest to the centroid while using documents up to level 3 of hierarchy.

The results are almost similar when we go deeper down the hierarchy and include documents from level 4. The best accuracy in this case is obtained when we use one

document per concept which is 91.6%. This shows a slight improvement of 0.4% when compared to our earlier best result. But taking into account the computational time required to train up to level 4, we decide to use the previous best case of 91.2% matches for 2 documents per concept by using concepts down to level 3 for our further analysis.

Experiment 3.2: In experiment 2, we use the 912 of the 1000 documents that have been classified correctly at level 1 using 2 documents per concept closest to the centroid and all concepts till level 3. We train our classifier in different ways to find out the way to get peak accuracy for level 2 decisions for the documents that were able to make the first step correctly. For the level 2 decision, we conduct 3 different experiments using documents just from level 2, levels 2 and 3, and levels 2, 3 and 4 to train the classifier. In each of the above experiments, we try to find the number of documents required to train the classifier from each concept for maximum accuracy.

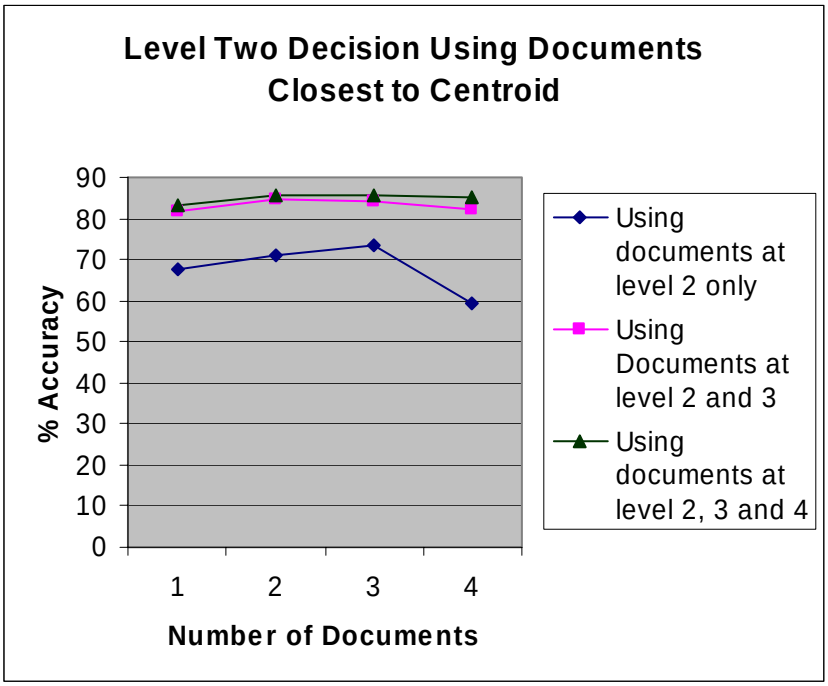

Chart 10: Level 2 decision using documents closest to the centroid

Discussion: It is clear from the above graph that the performance of the classifier is poor when it is trained with only level two documents. The accuracy of the classifier is nearly identical when we use documents from levels 2 and 3 versus documents from levels 2, 3, and 4. When we use data just from levels 2 and 3, the highest accuracy of 84.8% is obtained using 2 documents per concept. When we also include documents from level 4, there is a slight improvement in performance of the classifier and we get a peak performance of 85.7% accuracy when 2 documents closest to the centroid are used to train the classifier. We observe that as the number of documents per concept is increased, there is a drop in the performance of the classifier. This is explained because of the fact that as we bring in more documents we are increasing the noise.

Though we get the best observed accuracy of the classifier for level 2 decision when we use 2 per documents per concept that are closest to the centroid from each concept from levels 2,3, and 4, this shows only a improvement of 0.9% when compared to our earlier result with levels 2 and 3. But taking into account the computational time required to collect documents and train up to level 4, we use the documents only till level 3 for further analysis.

Experiment 3.3: We try to find the best observed way to train the classifier for level 3 decisions. As all the test documents belong to level 3 concepts, we are testing against their true concepts and determining how many of the documents manage to reach their actual concepts.

We first train the level 3 classifiers using only data from level 3 then we train the classifier using data from level 3 as well as level 4. In both the above cases, we select the documents that are closest to the centroid and vary the number of documents to find the combination that will give us the best observed classifier accuracy.

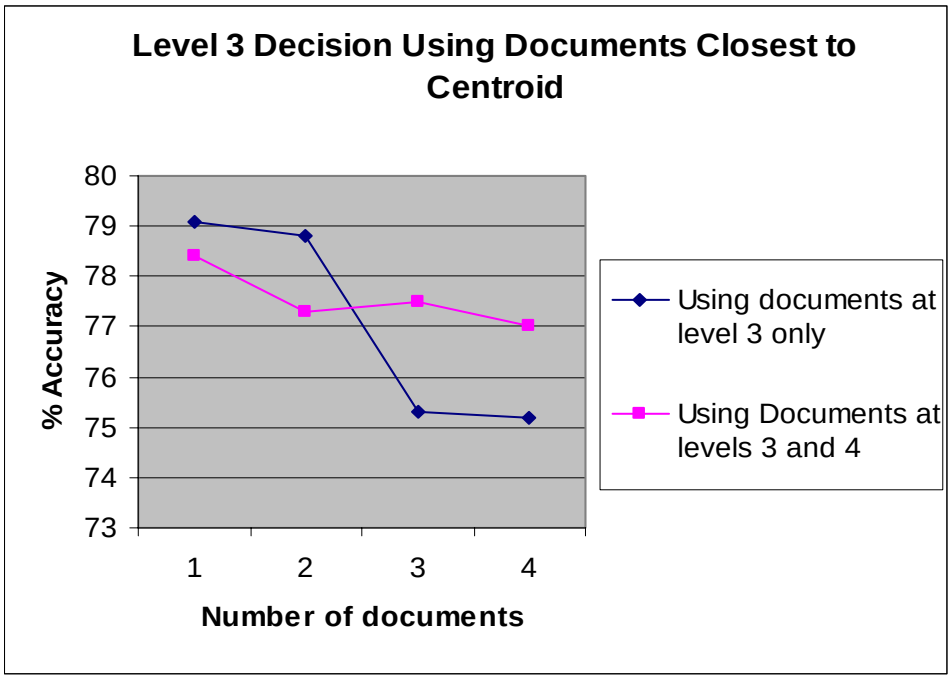

Chart 11: Level 3 Decision using documents closest to centroid from each concept

Discussion: We get 79.1% and 78.4% of exact matches using one document per concept for data from level 3 and level 3 and 4 respectively. The percentage of exact matches progressively decreases when we use more documents from each concept. This is because as we use more documents the focus of the training broadens and it diminishes the difference between the concepts. The fact that we are using document closest to the

centroid from each concept ensures that the selected training document is the best representative for that concept.

Thus, we can conclude that we get the best results for level three decisions when we use one document closest to the centroid from each concept at level 3. We get an exact match in 79.1% of the cases. Compared to the performance level of 46.6% of the flat classifier, we get an overall improvement of 70%.

Training Strategy:

It can be concluded from the above experiments we can devise a simple training strategy. If we train the classifier for each concept by selecting two training documents from each concept and sub-concept down to level 3, we get a high accuracy. It is sufficient to use training data only down to level 3 in the concept hierarchy. Although going to level 4 results in a minor improvement, this improvement is not significant and requires extra effort collecting and processing the additional training documents.

Experiment 3.4: To explore the improvement due to choosing the documents closest to the centroid, we train the classifier with two documents from each concept from all concepts down to level 3. This time we randomly select these two documents and compare that to using the two documents closest to the centroid.

Results:

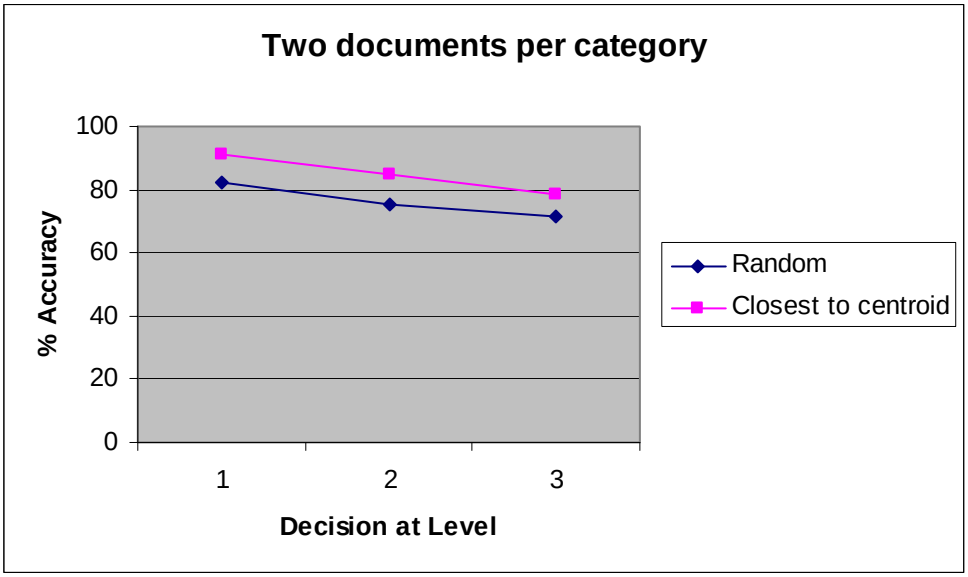

Chart 12: Comparison between training the classifier by choosing two random documents as against two documents that are closest to the centroid.

Discussion: For every level of decision there an improvement when we use clustering to select two documents that are closest to the centroid to train the classifier as compared to random selection. At level 1, 91.2% of the test documents make their decision correctly as versus just 82% for random selection. We observe the same trend continuing both for level 2 and level 3 decisions. At level 2, 84.8% of the test documents make the correct decision when we train the classifier using 2 documents closest to the centroid whereas only 75.2% of the test documents make it to this step correctly for random selection. Since all test documents belong to the level 3, the level 3 decision gives us the accuracy of the classifier, that is, the number of documents that have been corrected classified to

their true concepts. In our clustering approach, we find that 77.9% of the documents make it to their true concepts, whereas when we randomly select documents from concepts for training, only 71.8% of the documents are correctly classified to their true concept.

Broadly speaking, we can conclude that the best way to train the classifier is to use 2 documents per category that are closest to the centroid from all categories up to level 3. In this configuration the classifier gives us an accuracy of 77.9%. When compared against random selection of 2 documents from the same category it gives us an improvement of 8.4%. Compared to the original KeyConcept flat classifier, this denotes an improvement of 67% (relative).

Validation Testing:

To validate our testing strategy, we try to classify 400 new test documents. That is, we conduct the previous experiment with 400 test documents that the classifier has not seen before. We compare the results of using documents closest to the centroid to train the classifier versus selecting random documents to train the classifier.

Results:

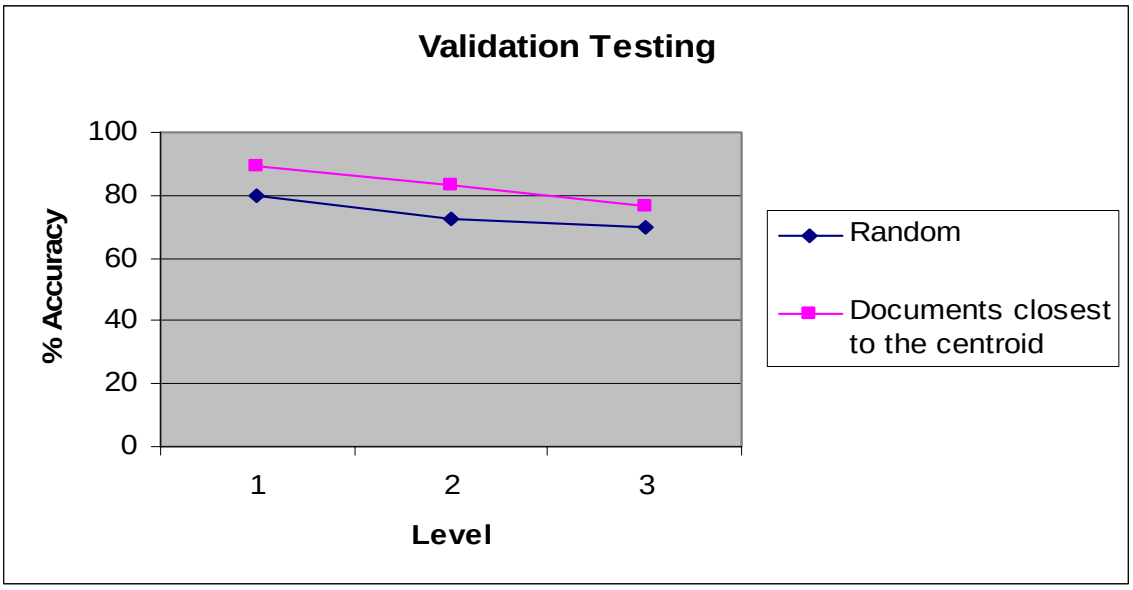

Chart 13: Validation Testing

We observe that using clustering to select the documents closest to the centroid enhances the level 1 classifier's accuracy from 79.7% to 89% and those of level 2 from 72.6% to 83.4%. The final accuracy is 76.2% on selecting the documents closest to the centroid when compared to 69.8% for random selection. This denotes a statistically significant improvement (*t-test value = 3.23E-05)* in case of hierarchical classification. Thus, there is a relative improvement of 63.5% over our baseline and 9.1% over randomly-selected hierarchical classification.

Chapter 6

Conclusions and Future Work

In this thesis, we presented a novel approach to text classification by combining within-concept clustering with hierarchical approach. We conducted experiments to determine how deeply we need to traverse the concept tree to collect training documents. We also did experiments to find out the number of training documents that when used to train the classifier will yield the best observed results. We achieved an improvement of 67% over flat classification using our approach. It can be concluded from our experiments that:

- The use of clustering only results in slight improvement in case of flat classification.
- For hierarchical text classification, by using clustering to select documents that are closest to the centroid in order to train the classifier, we get an accuracy of 65.4%. This represents 18.8% (absolute), or 40.3% (relative) improvement in the accuracy over the flat classifier. In this experiment, the documents from the parent and its children (and grandchildren) concepts are put in the same bucket to form the training set.
- For hierarchical text classification, if we train the classifiers is using 2 documents per category that are closest to the centroid from all categories up to level 3, we achieve an accuracy of 77.9%. This is an improvement of 67% (relative) or 31.3% (absolute) over the flat classifier.

There are many interesting avenues for future work. We used a vector space k-nn classifier in our research. The same techniques can be evaluated for other classifiers such as Support Vector Machines (SVM). We used Open Directory Project Web pages to evaluate our approach. Trials can be conducted on other datasets. Given the dynamic nature of the Web, dynamically changing concept hierarchy is needed to accommodate newly formed concepts.

In our hierarchical classification approach, an error made at the parent category is not recoverable further down the tree. Some mechanism to take care of this error cascading can be developed so that child classifiers are able to recover from the errors made at their parents.

References

[Chaffee 00] J. Chaffee, S. Gauch. Personal Ontologies For Web Navigation. In *Proceedings of the 9th International Conference On Information Knowledge Management (CIKM),* 2000, pages. 227-234.

[CLUTO] http://www-users.cs.umn.edu/~karypis/cluto/index.html

[Dubes 88] R. C. Dubes and A. K. Jain, Algorithms for Clustering Data, *Prentice Hall*, 1988.

[Dumais 00] S. Dumais and H. Chen. Hierarchical classification of Web content. In *Proc. Of the 23rd ACM International Conference on Research and Development in Information Retrieval*, Athens, GR, 2000, pages 256-263.

[D'Alessio 00] S. D'Alessio, K. Murray, R. Schiaffino, and A. Kershenbaum. The effect of using hierarchical classifiers in text categorization. In *Proceedings Of the 6th International Conference "Recherched'Information Assistee par Ordinateur"*, Paris, FR, 2000, pages 302-313.

[Elkan] C. Elkan, Naïve Bayesian Learning

[Frakes 92] W.B Frakes and R. Baeza-Yates. Information Retrieval: Data Structures and Algorithms.

[Gauch 04] S. Gauch, J. M. Madrid, S. Induri, D. Ravindran, and S. Chadalavada. KeyConcept : A Conceptual Search Engine. *Information and Telecommunication Technology Center, Technical Report : ITTC-FY2004-TR-8646-37, University of Kansas*. 2004

[Guha 98] S. Guha, R. Rastogi, and K. Shim. CURE : An efficient clustering algorithm for large databases. In *Proceedings of 1998 ACM –SIGMOD International Conference on Management of Data,* 1998.

[Gujha 99] S. Guha, R. Rastogi, and K. Shim. ROCK : a robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering .,* 1999.

[Karypis 99] G. Karypis, E.Han, and V.Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer,* 32(8): 68-75, 1999.

[Kaufman 90] L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis. *John Wiley and Sons*. 1990.

[KeyConcept 03] KeyConcept. http://ittc.ku.edu/keyconcept/

[Koller 97] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning,* 1997.

[Krovetz 92] Robert Krovetz and Bruce W. Croft. Lexical Ambiguity and Information Retrieval. *ACM Transactions on Information Systems*, 10(2), April 1992, pages. 115-141.

[Labrou 99] Y. Labrou, T. Finin. Yahoo! As An Ontology – Using Yahoo! Categories To Describe Documents. In *Proceedings of the 8th International Conference On Information Knowledge Management (CIKM)*, 1999, pages. 180-187.

[Liu 02] F. Liu, C. Yu and W. Meng. Personalized web search by mapping user queries to categories. In *Proceedings of the 11th International Conference on Information and Knowledge management, McLean, Virginia, USA*, 2002, pages. 558 - 565

[Madrid 00] J. Madrid and S. Gauch, Incorporating Conceptual Matching in Search. Submitted to the 11th Conference on Information and Knowledge Management. 2002

[Manning 99] C. D. Manning and H. Schutze. Foundations of Statistical Natural Language Processing. *The MIT Press.* 1999.

[ODP] Open Directory Project. http://dmoz.org

[Pearce 97] C. Pearce, E. Miller. The TellTale dynamic hypertext environment: Approaches to scalability. In *Advances in Intelligent Hypertext, Lecture Notes in Computer Science. Springer-Verlag*, 1997.

[Perkowitz 00] M. Perkowitz and O. Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Artificial Intelligence, 118:245-275,* 2000.

[Pitkow 02] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E.-Adar, T. Breuel. The consumer side of search: Personalized search. In *the Communications of the ACM*, September 2002. pages. 50-55

[Pulijala 03] A. Pulijala, Hierarchical Text Classification. Master's Project submitted to the EECS Dept., at The University of Kansas.

[Ruiz 99] M. Ruiz, P. Srinivasan. Hierarchical Neural Networks For Text Categorization. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, August 1999, pages. 281-282.

[Sasaki 98] M. Sasaki and K.Kita. Rule-based text categorization using hierarchical categories. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics,* La Jolla, US, 1998. Pages 2827-2830

[Schutze 95] H. Schutze, D. Hull, and J. Pedersen. A comparison of document representations and classifiers for routing problem. In *Proceedings of SIGIR 1995,* Pages 229-237.

[Sebastiani 02] F. Sebastiani. Machine Learning in Automated Text Categorization – *In ACM Computing Surveys* 34(1), 2002, pages. 1-47

[Sun 01] A. Sun and E. Lim. Hierarchical Text Classification and Evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM 2001),* California, USA, November 2001, pages 521-528.

[Sun 03] A. Sun, E. Lim, and W. Ng. Performance Measurement Framework for Hierarchical Text Classification. *Journal of the American Society for Information Science and Technology, 54(11), 2003*. Pages 1014-1028.

[Wang 01] K. Wang, S. Zhou, and Y. He. Hierarchical classification of real life documents. In *Proceedings of the 1st SIAM International Conference on Data Mining,* Chicago 2001.

[YAHOO] Yahoo! http://www.yahoo.com

[Zhao 02] Y. Zhao and G. Karypis, Evaluation of Hierarchical Clustering Algorithms for Document Datasets. In *Proceedings of Conference On Information Knowledge Management (CIKM),* 2002.

[Zhu 99] X. Zhu, S. Gauch, L. Gerhard, N. Kral, A. Pretschner. Ontology-Based Web Site Mapping For Information Exploration. In *Proceedings of the 8th International Conference On Information Knowledge Management (CIKM),* 1999, pp. 188-194.