New Media and Mass Communication
ISSN 2224-3267 (Paper) ISSN 2224-3275 (Online)
Vol.88, 2020

Hierarchical Afaan Oromoo News Text Classification

Fanta Teferi Megersa
College of Engineering and Technology, Mettu University

Abstract
The advancement of the present day technology enables the production of huge amount of information. Retrieving useful information out of these huge collections necessitates proper organization and structuring. Automatic text classification is an inevitable solution in this regard. However, the present approach focuses on the flat classification, where each topic is treated as a separate class, which is inadequate in text classification where there are a large number of classes and a huge number of relevant features needed to distinguish between them.This paper explores the use of hierarchical structure for classifying a large, heterogeneous collection of Afaan oromoo News Text. The approach utilizes the hierarchical topic structure to decompose the classification task into a set of simpler problems, one at each node in the classification tree. An experiment had been conducted using a categorical data collected from Ethiopian News Agency (ENA) using SVM to see the performances of the hierarchical classifiers on Afaan Oromoo News Text. The findings of the experiment show the accuracy of flat classification decreases as the number of classes and documents (features) increases. Moreover, the accuracy of the flat classifier decreases at an increasing number of top feature set. The peak accuracy of the flat classifier was 68.84 % when the top 3 features were used. The findings of the experiment done using hierarchical classification show an increasing performance of the classifiers as we move down the hierarchy. The maximum accuracy achieved was 90.37% at level-3(last level) of the category tree. Moreover, the accuracy of the hierarchical classifiers increases at an increasing number of top feature set compared to the flat classifier. The peak accuracy was 89.06% using level three classifier when the top 15 features were used.Furthermore, the performance between flat classifier and hierarchical classifiers are compared using the same test data. Thus, it shows that use of the hierarchical structure during classification has resulted in a significant improvement of 29.42 % in exact match precision when compared
with a flat classifier.

Keywords: Automatic Text Classification
DOI: 10.7176/NMMC/88-01
Publication date: February 29th 2020

1 Introduction

Humans use classification techniques to organize things in various activities of their life. People make their own judgment to classify things in their everyday life – they classify things based on similarities or likeliness of color, size, concept, ideas, subject and so on (Koller& Sahami, 1997).

The need to classify information resources has become an important issue as the production of such resources increase dramatically from time to time. More specifically, for the last 6 decades

(Dumais & Chen, 2000), there is a great increase in the production of information. Manuscripts, newspapers, journals, magazines, thesis and dissertations are available electronically in different formats such as text, audio, video, and graphics. Several studies have shown that these collections are constantly growing from time to time due to the advancement of technology (Chien et.al. 2004). However, seeking information of one's need in these huge collections requires organization. Especially in a system where there is large collection of documents, retrieval of a given document or set of documents is possible if the collection is organized systematically. Many web sites offer a hierarchically organized view of the Web. E-mail clients offer a system for filtering e-mail. Academic communities often have a Web site that allows searching on papers and shows an organization of papers.

Nowadays, news items are produced every day in digital devices and organized in some order (Rennie, 2001). However, most of the time, text classification process is done manually which brings about enormous costs in terms of time and money. In other words, organizing documents by hand or creating rules for filtering is painstaking and labor-intensive

Therefore; automatic classification systems are very desirable since they minimize such problems (Neumann & Schmeier, 1999; Rennie,2001).

Automatic text classification can be done using two approaches (Sebastiani, 2002; Rasmussen, 1992): clustering or classification. Text clustering is the automatic identification of a set of natural categories and the grouping of documents under each. Text classification, on the other hand, is the automatic assignment of documents to a predefined set of categories. Many previous studies focus on *flat classificatio*n, in which the predefined categories are treated individually and equally so that no structures exist to define relationships among them (Yang &Liu, 1999; D'Alessio et al., 2000). A single huge classifier is trained which categorizes each new document as belonging to one of the possible predefined classes.

Limitations to the flat classification approach exists in the fact that, as the Internet grows, the number of possible categories increases and the borderlines between document classes are blurred. As we use a large corpus we may have hundreds of classes and thousands of features. The computational cost of training a classifier for a problem of this size is prohibitive.

Furthermore, the variance of the resulting classifier is typically very large; since such a model will have many thousand parameters which need to be estimated and thus can easily lead to over fitting of the training data.Even though, previous work (Koller & Sahami, 1997) has shown that feature selection can be a useful tool in dealing with these issues by eliminating many of the words that appear in the corpus as being indicative of any topic so as to obtain a significant increase in accuracy, the computational cost and the robustness still pose significant limitations.

1.2 Afaan Oromo Languages

Afaan Oromo is one of the major African languages that is widely spoken and used in most parts of Ethiopia and some parts of other neighbor countries like Kenya and Somalia (Journal of Oromo studies, 1793). It is used by Oromo people, who are the largest ethnic group in Ethiopia, which amounts to 25.5% of the total population. Besides first language speakers, a number of members of other ethnicities who are in contact with the Oromo's speak it as a second language, for example, the Omotic speaking Bambassi and the Nilo-Saharan-speaking Kwama in northwestern Oromia. Currently, Afaan Oromo is an official Language of Oromia regional state (which is the largest Regional State among the current Federal States in Ethiopia). Being the official language, it has been used as medium of instruction for primary and junior secondary schools of the region. Moreover, the language is offered as a subject from grade one throughout the schools of the region. Little literature Works, a number of newspapers, magazines, educational resources, official credentials and religious documents are published and available in the language.

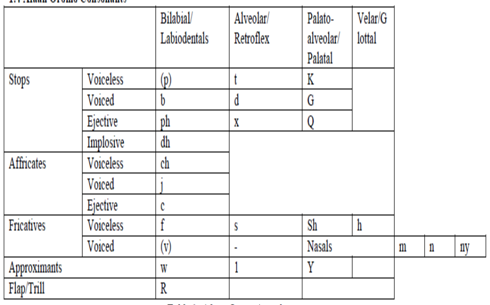
Afaan Oromo is Cushitic language which is family of Afro Asiatic languages. It is spoken by more 22 Million peoples and most of native speakers are people living in Ethiopia, Kenya, Somalia and Egypt. It is third largest language in Africa following Kiswahili and Hausa; 4th largest language, if Arabic is counted as Africa language (Duwairi R, April 1807.`) The exact time when the Latin alphabet started being used for Afaan Oromo writing was not well known, but on November 3, 1791 it adopted as official alphabet of Afaan Oromo on. Now it is language of public media, education, social issues, religion, political affairs, and technology.

In general, Afaan Oromo is widely used as written and spoken language in Ethiopia and neighboring courtiers like Kenya and Somalia. With regard to the writing system, "**Qubee**"(a Latin-based alphabet) has been adopted and become the official script of Afaan Oromo.

1.3 Afaan Oromo Alphabets and Writing System

Afaan Oromo is a phonetic language, which means that it is spoken in the way it is written. The writing system of the language is straightforward which is designed based on the Latin script. Unlike English or other Latin based languages there are no skipped or unpronounced sounds/alphabets in the language. Every alphabet is to be pronounced in a clear short/quick or long /stretched sounds. In a word where consonant is doubled the sounds are more emphasized. Besides, in a word where the vowels are doubled the sounds are stretched or elongated.

Like in English, Afaan Oromo has vowels and consonants. Afaan Oromo vowels are represented by the five basic letters such as a, e, i, o, u. Besides, it has the typical Eastern Cushitic set of five short and five long vowels by doubling the five vowel letters: „aa", „ee", „ii", „oo", „uu". Consonants, on the other hand , do not differ greatly from English, but there are few special combinations such as "**ch**" and "**sh**" (same sound as English),"**dh**" in Afaan Oromo is like an English "*d*" produced with the tongue curled back slightly and with the air drawn in so that a glottal stop is heard before the following vowel begins. Another Afaan Oromo consonant is "**ph**" made when with a smack of the lips toward the outside "**ny**" closely resembles the English sound of "gn". We commonly use these few special combination letters to form words. For instance, **ch** used in **barbaachisaa** *„important'*, **sh** used in **shamarree '***girl'*, **dh** use in **dhadhaa** *„butter'* , **ph** used in **buuphaa '***egg',* and **ny** used in **nyaata** *„food'* . In general, Afaan Oromo has 36 letters (26 consonants and 10 vowels) called "**Qubee".** All letters in English language are also in Afaan Oromo except the way it is written. Table 2 shows Afaan Oromo alphabet.



1.4 Afaan Oromo Consonants

		Bilabial/ Labiodentals	Alveolar/ Retroflex	Palato-alveolar/ Palatal	Velar/G lottal			
Stops	Voiceless	(p)	t	K				
	Voiced	b	d	G				
	Ejective	ph	x	Q				
	Implosive	dh						
Affricates	Voiceless	ch						
	Voiced	j						
	Ejective	c						
Fricatives	Voiceless	f	s	Sh	h			
	Voiced	(v)	-	Nasals		m	n	ny
Approximants		w	1	Y				
Flap/Trill		R						

Table 1: Afaan Oromo vowels

	Front	Central	Back
High	i , ii	u , uu	
Mid	e , ee	o , oo	
Low	a	aa	

Table 2: Afaan Oromo Alphabet

Afaan Oromo punctuation mark is placed in text to make meaning clear and reading easier. Analysis of Afaan Oromo texts reveals that different punctuation marks follow the same punctuation pattern used in English and other languages that follow Latin Writing System. Similar to English, the following are some of the most commonly used punctuation marks in Afaan Oromo:

1.2. Materials & Methods

Text categorization is the task of automatically assigning input text to a set of categories. Text categorization can be divided in to text classification and text clustering based on the category it uses. Text clustering is the automatic identification of a set of categories and assigns set documents under the automatically identified categories.

1.2.1.Development Tools and Techniques

There are many Machine learning algorithms such as Support Vector Machine(SVM),Decision Tree, Naïve Bayes and Artificial Neural Network used for hierarchical text classification(Ranganatan, 2001) SVM is selected for this study due to its capability of providing the following benefits as
compared to other algorithms (Joachims, 1998). SVMs: support high dimensional input space so that it can deal with large data sets tend to be less prone to over fitting since the learned classifier is characterized by the number of support vectors rather than the dimensionality of the data are capable of providing more accurate results LIBSVM$^{\textbf{multiclass}}$ is selected as a tool for this experiment as it provides the following main
features: Different SVM formulations

- Efficient multi-class classification
- Cross validation for model selection
- Automatic model selection which can generate contour of cross validation accuracy.
- Automatic parameter tuning to select the best parameter for training the classifier.
- Python 3.0 is used as a programming tool for data preprocessing as it is a powerful too in text preprocessing, easy to use and familiarity with the researcher.

1.3. Text Categorization Techniques

Text categorization techniques can be supervised, unsupervised and semi supervised learning.

Supervised learning is the search for algorithms that reasons from externally supplied class to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features.

The unsupervised learning does not require externally supplied knowledge for categorizing instances. The main aim of this learning algorithm is to generate a group of features that have similar properties. So the grouping is done without any supervision of human beings. Text clustering is an example of unsupervised learning. In semi supervised learning, parts of the documents are required external knowledge and other does not require external knowledge in the categorization process.

1.3.1. Text classification

Text classification is an example of supervised learning where a given document is assigned to predefined categories based on the similarities of the labeled documents in the training set. Text classification can be done manually or automatically. Traditionally, text classification has been performed manually. The manual text classification uses expert (human being) to categorize the document in to predefined categories. However, as the number of documents explosively increases, the task becomes no longer amenable to the manual categorization, and it requires a vast amount of time and cost. This has led to numerous researches for automatic document classification. The automatic text categorization is generally divided in to two main categories. These are flat text categorization and hierarchical text categorization.

1.3.2. Text Clustering

It is easy to collect unlabeled documents for text categorization purposes. As a result, a text categorization mechanism is required for categorizing the unlabeled documents. This mechanism is called text clustering. Text clustering is an unsupervised learning which does not require pre-defined categories and labeled documents. The main aim of text clustering is to determine the intrinsic grouping in a set of unlabeled data. The intrinsic groups have high intragroup similarities and low intergroup similarities. Text clustering algorithms are categorized into two main groups such as hierarchical algorithms create a cluster by splitting the data into k partition where each partition represents a cluster.

1.4. Text Categorization Phases

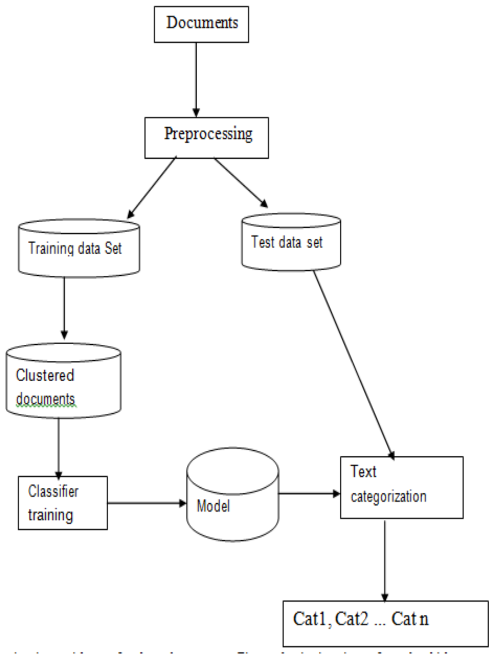
Text documents represented in a natural language are not easily used by the classifier for building algorithms. In order to solve this problem mapping a text document into a schematic representation of its content is required. As a result, the categorization algorithm transforms each document into a vector of weights corresponding to an automatically chosen set of keywords. This transformation has two main steps (Duwairi R., April 1807).

First, suitable representation of the document has to be chosen. This representation is used for all documents to be indexed and it has all necessary words that can characterize the documents. The information retrieval and machine learning researchers have different views in representing strings .Second; it assigns weights to each representation term which shows the frequency of occurrence of the term in the indexed document. Even though the document indexing is performed, the results obtained has high dimension and take a large amount of storage space.

1.5. Proposed System Architecture

In this study, the proposed text categorization system is developed in three stages as shown in figure1. These are preprocessing, clustering, classifier training, and testing the text categorization system. The preprocessing makes the raw data ready for the experiment. In this stage the irrelevant terms are removed from the documents and words of the same context with different forms are converted into the same word. The final goal of this stage is to convert the collection of text in to matrix of index terms with their tf×idf weight values.

In the experimentation, cluster of documents are created for training data set. The resulting clustering solution is used to train a text classification model. The developed model is tested using test data set. Finally, the system comes up with categories of Afaan Oromo news text documents.



Text Preprocessing

Text preprocessing is crucial step for the subsequent text clustering and classification tasks. During text preprocessing, there are a sequence of steps applied to generate content-bearing terms and assigns weights that shows their importance for representing the document they identified from. First, tokenization is performed which attempts to identify words in the document corpus. The common method of representing the document text is using the bag of words approaches where the word from the document corresponds to a feature and the documents are considered as a feature vector. This indicates that the words can only discriminate the given documents in the categorization process. So, the punctuation marks and digits are irrelevant component of the text documents. In the present study, the punctuation marks and digits are removed and replaced with space. Tokenization is the process of chopping character streams in to tokens, while linguistic preprocessing then deals with building equivalence classes of tokens which are the set of terms that are indexed. Tokenization in this work also used for splitting document in to tokens and detaching certain characters such as punctuation marks.

```
For file in corpus
    Define word delimiter to space|
    Read files
        For file in read
        If there is word delimiter
            Put each terms as separate token
```

Algorithm 1: Tokenization

Normalization involves process of handling different writing system. Primarily every term in the document should be converted in to similar case format in this study lower case. For instance "INFORMATION‟, „Information‟, „information‟ is all normalized to understandable as lower case „information‟ the system.

Number	Stop word	Meaning
1	kana	This
2	Sun	That
3	Inni	He
4	Ani	Me?
5	Isaan	They
6	Ishee	She
7	Akka	Like
8	Ana	Me
9	Fi	And

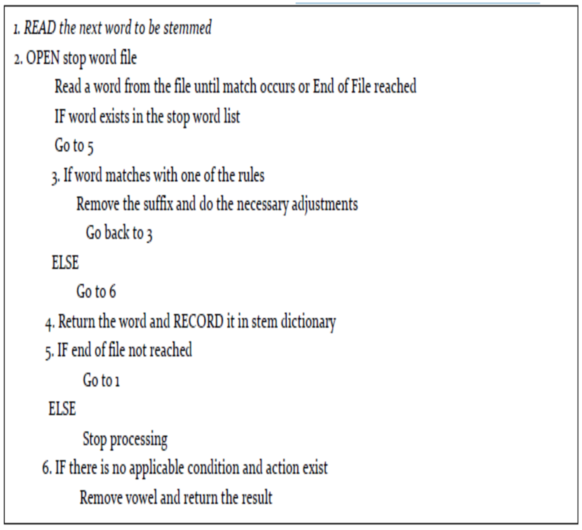
Table 1.1: Sample Stop Word lists of Afaan Oromo documents

In the present study, stop words are identified manually by consulting books, dictionaries and different research articles of the Afaan Oromo languages. The consulted books and dictionaries help to identify preposition, conjunction, articles, pronoun and auxiliary verbs of the Afaan Oromo language. After identifying the stop words in the Afaan Oromo documents, algorithm 1.1.2 remove the stop words from the document corpus.

```
: Stop word removal
Read stop word list file
Open the file for processing
Do
  Read the content of the file line by line
   Assign the content to string
    For word in string split by space
      If word in stop word list
       Remove word from the index term
      Else
        Continue
     End if
  End for
While end file
```

Algorithm 1.1.1: Stop word removals

Stemming: stemming is process used in most search engines and information retrieval systems. It is core natural language processing technique for efficient and effective IR system. Generally stemming transforms inflated words in to their most basic form. There are different stemming algorithms but the most common one is that of Porter, called „Porter Stemmer‟. Even if stemming is very similar to lemmatization in most of indexing process stemming is used. Stemming is language dependent process in similar way to other natural language processing techniques. It is often removing inflectional and derivational morphology. E.g. automate, automatic, automation = *automat*.

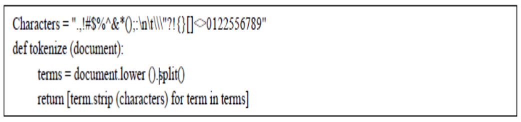


```
1. READ the next word to be stemmed
2. OPEN stop word file
      Read a word from the file until match occurs or End of File reached
      IF word exists in the stop word list
      Go to 5
      3. If word matches with one of the rules
         Remove the suffix and do the necessary adjustments
          Go back to 3
      ELSE
         Go to 6
    4. Return the word and RECORD it in stem dictionary
    5. IF end of file not reached
          Go to 1
      ELSE
          Stop processing
    6. IF there is no applicable condition and action exist
          Remove vowel and return the result
```

Algorithm 1.1.2: Stemming Algorithm

Index Construction

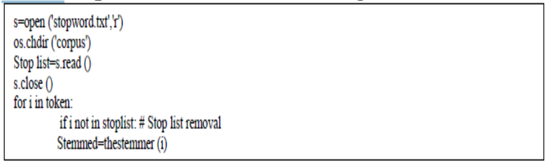
Indexing involves tokenizing, normalization, stop word removal and stemming. The code 1.2.1 is fragmenting code that Tokenizes and normalizes the terms in the document.



```
Characters = ".,!#$%^&*();:\n\t\\\"?!{}[]<>0122556789"
def tokenize (document):
      terms = document.lower ().split()
      return [term.strip (characters) for term in terms]
```

Code 1.2: Tokenization and Normalization

Here this fragment code splits document based on space between each words, then convert every words in document in to lower case if it is not in lower case primarily. The documents in lower case are checked as if it is not punctuation mark or number. Finally normalized and tokenized document will be returned for next process. The index terms selected in this study are content bearing terms which are not part of stop list. Primarily there are identified list of stop words which are not content bearing, just used for grammatical purpose only. The following code is used for the stop word removal from the documents.



```
s=open ('stopword.txt','r')
os.chdir ('corpus')
Stop list=s.read ()
s.close ()
for i in token:
        if i not in stoplist: # Stop list removal
        Stemmed=thestemmer (i)
```

Code 1.2.1: Stop word removal

Support Vector Machine

The core idea behind the SVM classifier is to fit the linear model to the mapped training data by maximizing the margin of the classifier [G., Steinbach, M. and Kumar, V. Karypis,]. This shows that the value of the given parameter of hyper plane to them nearest training patterns from given classes is maximized as many training patterns as possible. In this study, the researcher classifies the Afaan Oromo documents using sequential minimal optimization supports Support Vector machine (SVM) classifiers. It has advantage over the other support vector machine classifiers. First, the amount of memory required for SVM is linear in SVM breaks the large quadratic programming problem into a series of smallest possible quadratic programming problems. The training set size which allows SVM to handle very large training sets. Second, it avoids matrix computation and this makes SVM to scale somewhere between linear and quadratic in the training set size for various test problems.

Finally, it is the fastest classifier for linear SVM and sparse data sets. Generally, the SVM algorithm involves three important components. These are an analytic solution to the optimization problem for the two chosen multipliers; a heuristic for choosing the two multipliers to optimize at a given step; and a step to compute the offset.

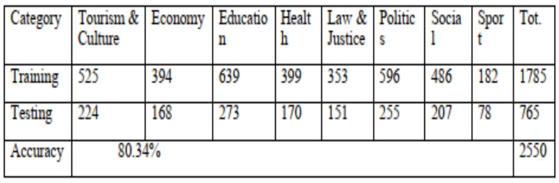
2. Experiments and Results 2.1.1 Experimental Setup Training data The number of news documents used in this experiment was 5100. Since hierarchical classification emphasizes the relationship among classes, rather than building single huge classifier, a classification is accomplished with the cooperation of classifiers built at each level of the tree. The training data is organized into 3 levels: from level-0(root level) to level-2. Each level represents classes or subclasses in a classification tree. Thus, there were 8 classes at level-0, 20 classes at level-1, and 69 classes at level-2 with at least 14 documents in them. The classifiers at each level were trained using the associated documents of all subclasses of that class. Thus, the level-0 classifier was trained using documents of all subclasses of that class from level-1 through 2. In contrast, each level-1 classifier was trained with documents from the appropriate level-1 subclasses up level-2 .**Testing data** The accuracy of the classifier was tested using the test data selected from each level-3 documents. These documents were excluded from the training process and were selected from different level-3 classes. Since the class from which the test documents were selected is known, the accuracy of the classifier is evaluated how often the classifier assign the test documents to the classes from which they originally came.

Effects of the numbers of classes of documents on flat classification

We created a single classification system by training a flat classifier for all classes in the top 3 levels of the classification tree, ignoring structure. In other words, each of the 97 classes was trained using 70% of the documents from each class. We had broken the classification process into pieces of classes taken separately at a time to see the performance of the classifier while increasing number of classes and documents (features). Thus, classes in level-0, level-1, and level-2 were separately considered for the first, second, and third experiment respectively. Since each document is assigned in the leaf node of the classification tree; level-0 classes will have the same number of documents as that of level-1 and level-2 when used separately for the next corresponding experiment. Hence, I selected documents using 50% of the collection to experiment on level-1 classes, and 70% of collection for the second experiment and 90% of the document collection for the third experiment. Moreover, the effect of top features on the classifier performance of a flat classification system and the reasons behind is addressed in this section of the experiment.

Experiments labeled as I, II and III shows the result and discussion of the experimentation when an increasing number of classes and documents were considered whereas Experiment IV shows the effect of top features on the classifier performance of a flat classification system.

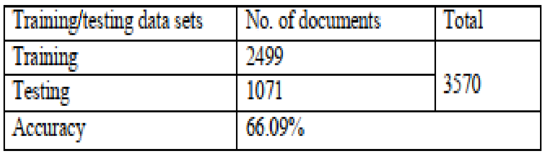
Classification with 8-Classes In this level, 8- Classes and 50% of the total number of documents in the collection were considered. Training and testing data shares the 70% and 30% of these data. Hence, 1785 documents were used for training and 765 documents were used for testing. Thus, 80.34% accuracy was found as result of this experiment.



Category	Tourism & Culture	Economy	Educatio n	Healt h	Law & Justice	Politic s	Socia l	Spor t	Tot.
Training	525	394	639	399	353	596	486	182	1785
Testing	224	168	273	170	151	255	207	78	765
Accuracy	80.34%								2550

Table 2. Experiment on 8-(level-0) classes

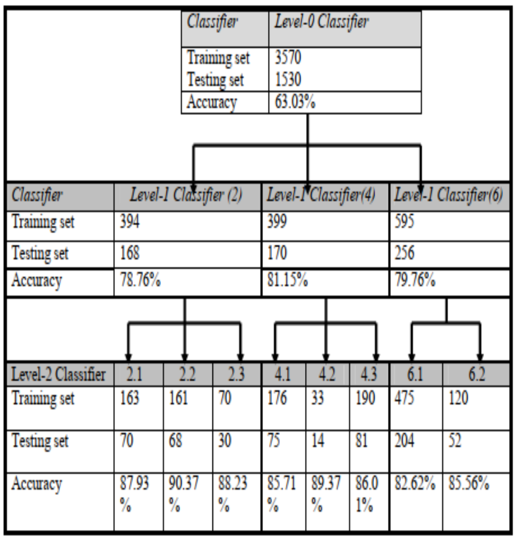
Classification with 20-Classes The number of classes which are considered in this experiment was 20; and then 70 % of the total documents in the collection (from which 70% of it for training and the remaining for testing) were used. Thus, 66.09% accuracy was found as the result of the experimentation. Table 2.2 shows the result of the experimentation when more number of classes and documents were used



Training/testing data sets	No. of documents	Total
Training	2499	3570
Testing	1071	
Accuracy	66.09%	

Table 2.2.Experiment on 8-(level-0) classes

Experiment Using Hierarchical Classification For the hierarchical classifier, we constructed a set of classifiers, one at each level of the classification tree, using training method described above. Thus, there was one classifier at level-0 (trained on the 8 level-1 classes), 8 classifiers for level-1 (one for each level- 1 class), and 20 classifiers for level-2 (one for each level-2 class). Since each classifier has to deal with a more easily separable problem, and can use an independently optimized feature set, it leads to slight improvements in accuracy apart from the gain in training (learning) and testing speed. Table 2.3 shows the performance of hierarchical classifier as it improves down the hierarchy for randomly selected classes (Education(code 2), Health (code 4) & Politics (code 6)).



Classifier	Level-0 Classifier
Training set	3570
Testing set	1530
Accuracy	63.03%

Classifier	Level-1 Classifier (2)	Level-1 Classifier(4)	Level-1 Classifier(6)
Training set	394	399	595
Testing set	168	170	256
Accuracy	78.76%	81.15%	79.76%

Level-2 Classifier	2.1	2.2	2.3	4.1	4.2	4.3	6.1	6.2
Training set	163	161	70	176	33	190	475	120
Testing set	70	68	30	75	14	81	204	52
Accuracy	87.93 %	90.37 %	88.23 %	85.71 %	89.37 %	86.0 1%	82.62%	85.56%

The Increasing performance of hierarchical classifiers Table 2.3 shows the improved performance of the hierarchical classifiers at each levels of the classification tree. This is because each classifier deals with the documents associated to only that class or subclasses of that class and it concentrates on a smaller set of documents, those relevant to the task at hand. As it is shown above, exploiting the relationship among classes and utilizing the hierarchical topic structure results in a considerable increase in the classifier accuracy. The testing data in the table 2.3 shows those instances which participate in testing the level-0 classifier, extracted from the corresponding class in level1 through 2. This is because the LibSVM only evaluates whether the classifier assigns the test documents to the classes from which they originally came; from which the accuracy is calculated using the equation described in equation in the above equation. Thus, in this study level-1 and level-2 classifiers are tested with documents selected from that classes and subclasses of that class; others left out as it always degrade the accuracy value of the classifier.

Effects of Number of Top Features in Hierarchical Classifiers

The documents were initially classified at level-0 using a varying number of features per document where the features were selected based on their *tfidf* weights. The first run used only the highest weighted feature for classifying the documents and number of features was increased in each subsequent run until a maximum of 20 features. The level-0 classifier had a peak accuracy of 81.50% when the top 5 features were used. Figure 2.3 shows the result of the experiment.

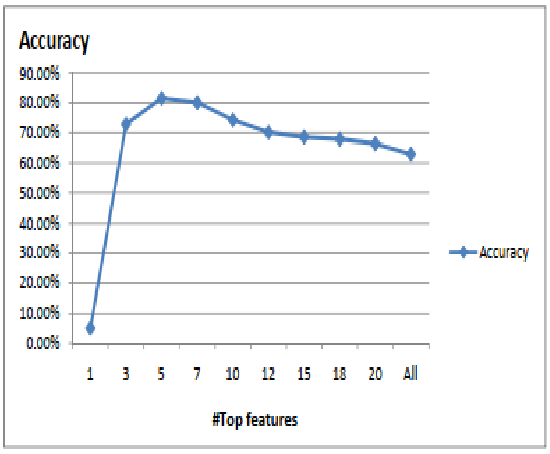


Figure2.3: The effect of the number of top features selected from test documents on level-0 classification accuracy

The test documents were then classified at level-1 while again varying the number of top features from 1 to 20. At level-1, the classification process is same as above, but it is constrained to consider only the subclasses of the best matching class at level-0. As shown in figure2.4 below, the level-1 classifier had a peak accuracy of 85.07% when the top 10 features were used.

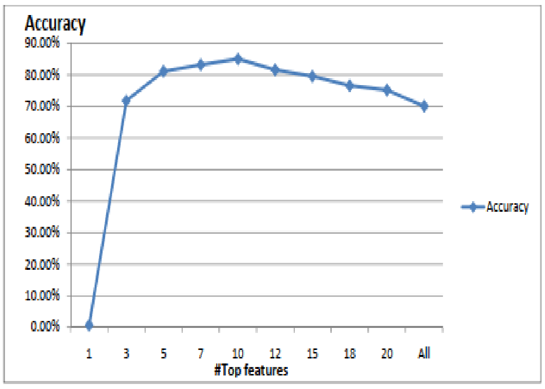


Figure2.4: The effect of the number of top features selected from test documents on level-1 classification accuracy

Finally, the test documents were classified at level-2 with the classification process now constrained to consider only the subclasses of the best matching class at level-1. Since all the test documents originally came from level-2 classes, the accuracy of the classifier overall is best judged by the accuracy at level-2. The level-2 classifier had an exact match precision of 89.06% when the top 15 features were used. This means that, from a set of 97 classes, the hierarchical classifier correctly classified 89.06% of documents to their original class.

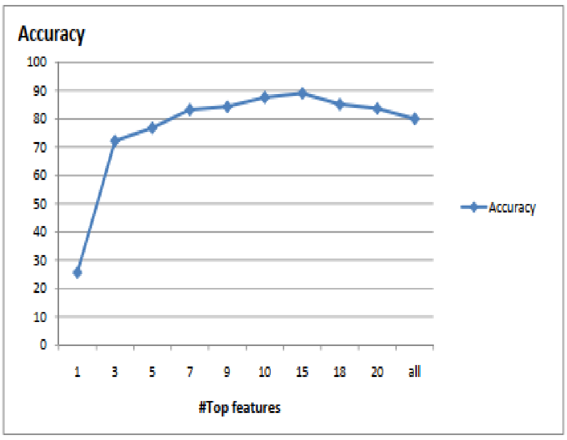


Figure2.5: The effect of the number of top features selected from test documents on level-2 classification accuracy it is interesting to note that, as we move down the hierarchy, the classifiers perform better with more features extracted from test documents. This is because they need more information in order to make finer-grained distinctions between the classes.

Comparison between Flat Classifier and Hierarchical Classifiers Analyzing the result of the above experiments, as the number of class and documents to be considered increases, the performance of flat classifier decreases but the performance of hierarchical classifiers increases as we move down the hierarchy. This shows that flat classification depends on the number of classes and documents to be considered compared to hierarchical classifiers. To compare the relative performance of the flat classifier and the hierarchical classifiers, the same set of test documents was used. As shown in figure2.2 and figure2.5, the flat classifier produced an exact match accuracy of only 68.84 % when the top 3 features were used, whereas with the hierarchical classifiers 89.09 % of the documents classified had exact match accuracy when the top 15 features were taken. This implies that least number of features are needed to discriminate among a large number of classes where there is no relationship among classes; whereas more number of features are needed to discriminate among classes where there are more closer similarity between classes or documents of a class (in hierarchical classification). Thus, we can conclude that the use of hierarchy for text classification results in a significant improvement of 29.42 % in exact match accuracy over the flat classifier.

Conclusion Along with the advancement of Information Technology, we are flooded by huge amount of information. This problem has caused obstacles to human beings to get useful information out of these huge collections. Hence, this necessitates a proper way of data organization and management in such a manner that it can be easily accessible to those who seek it. One way is the use of manual document classification. Manual classification requires individuals to assign each document to one or more categories. However, as the amount of data and information increases, this approach became tedious and consumes times to organize documents using human hand. An alternative way is to organize data and information using automated systems, which are often referred to as automatic document classification. It uses automatic solutions to classify an electronic document into one or more categories based on the characteristics of its contents. Several researches have been done on automatic document classification with the help of different machine learning approaches; and good results were found. However, most of them focus on flat classification system, i.e. each topic (category) is considered independent of others where there is no any relationship among them. Even though, flat classification has become a well-established research area for the last 30 to 40 decades and many good classifiers have been developed, the approach hadn't yet a feasible solution where most real world application need structures that define relationships among them are necessary. As the technology such as internet grows, the number of possible categories increases and the borderlines between document classes are blurred. As we use a large corpus we may have hundreds of classes and thousands of features. The computational cost of training a classifier for a problem of this size is prohibitive to solve these problems, approach that utilizes the hierarchical topic structure to decompose the classification task into a set of simpler problems is proposed. It is often referred to as hierarchical classification approach. Hierarchical text classification uses a divide-and-conquer approach (also known-as top-down approach) to deal the large classification problem into a set of simpler sub problems, one at each node in the classification tree. In such a hierarchical structure document types become more specific as we go down in the hierarchy. Thus, hierarchical classification of documents is very much important to access a specific document or group of documents from the hierarchical organized document collections as compared to flat classification. This paper also introduces support vector machines (SVM) for hierarchical text categorization. It provides both theoretical and empirical evidence that SVMs are very well suited for text categorization in general and hierarchal classification in particular. The theoretical analysis concludes that SVMs acknowledge the particular properties of text: (a) high dimensional feature spaces, (b) few irrelevant features (dense concept vector), and (c) sparse instance vectors. The experimental results show that SVMs consistently achieve good performance on hierarchical text categorization tasks, outperforming existing methods substantially and significantly. With their ability to generalize well in high dimensional feature spaces, SVMs eliminate the need for feature selection, making the application of text categorization considerably easier. Another advantage of SVMs over the conventional methods is their robustness. Furthermore, SVMs do not require any parameter tuning, since they can find good parameter settings automatically. All this makes SVMs a very promising and easy-to-use method for learning text classifiers from examples. LibSVM was used as an experimentation tool due to its ability for efficient multiclass classification, automatic model selection and contains different SVM formulations. The data collected for this study a one-level data. However, it is preprocessed in a manner that hierarchical data (categorically leveled) data is obtained to fit with the purpose of the study. Document-similarity matrix and experts' judgment were used to generate a categorical level data. Therefore, three level hierarchical data is obtained with 8 level-0 classes, 20 level-1 classes and 69 level-2 classes. The experiment was done following three approaches. 1. Assure whether the traditional classification method (flat classification system) is dependent on the number of classes and features 2. Constructing hierarchical classifiers at each levels of the classification tree and see whether the performance of the classifiers were improved as we move down the hierarchy 3.Evaluating the classification performance between existing traditional system (flat classifier) and the hierarchical classifiers with the same test data. Accordingly, the following result was obtained based on the experiments done using the above three approaches. Based on the first approach, the 97 classes were divided into 8, 20 and 69 separate classes with an increasing number of documents in them. Thus, it was found that the accuracy was decreasing from 80.34% to 66.09% and then to 52.32% as the number of classes increased from 8 to 20 and then to 69; and the number of documents increased from 2550 to 3570 and then to 4590 respectively. This shows that as the number of classes and documents increase, the performance of a flat classifier decreases. This is because, as the number of support vectors increases with the increasing number of documents. Since the classification is done in a multidimensional plane where we can draw a number of hyper planes, the increasing number of support vectors causes to narrow the margin between these hyper planes. The smaller the marginal hyper plane then causes maximum misclassification error on the later unseen test instances apart from a difficulty to get the maximum marginal hyper plane (MMH).According to the second approach, an experiment was done on randomly selected class levels as shown in table4.9. Thus, it shows an improved performance as we move down the classification tree. For example, the maximum accuracy achieved is in level-2(the last level in the category level), which is 90.37% in the economy class as designated using code 2.2 in table2.3. The improved performance of the hierarchical classifiers at each levels of the classification tree in the experiment is because each classifier deals with the documents associated to only that class or subclasses of that class, it concentrates on a smaller set of documents, those relevant to the task at hand. Hence, the maximum marginal hyper plane can be easily generated in one hand linear SVM classifier can be easily applied in the other hand. Moreover, we can deduce that a considerable increase in the classifier accuracy is as a result of exploiting the relationship among classes and utilizing the hierarchical topic structure in it.

Again based on the third point, an experiment was done using the top number of features selected from the same test document using one flat classifier and classifiers at each levels of the category tree. Accordingly, a flat classifier shows a maximum exact match accuracy when 3 top features were used whereas the hierarchical classifiers shows an improved exact match accuracy at increasing number of top features at each level of the category tree. In such a way, the flat classifier produced an exact match accuracy of only 68.84% when the top 3 word were used, whereas with the hierarchical classifiers 89.09% of the documents classified had exact match accuracy when the top 15 features were taken. This means that the use of hierarchy for text classification results in a significant improvement of 29.42 % in exact match accuracy over the flat classifier. From this experiment, we can understand that more words are needed to discriminate classes (topics) that are close to each other in hierarchy as they have more in common with each other than classes (topics) that are spatially far apart. Apart from the increased classification performance, classification speed can be taken as an advantage in hierarchical classification approach using Support Vector Machine. The only limitation to SVM is the longer learning/training time. It might take more than a day. This learning time could increase with an increase number of training data. Finally, the findings of this research could be much significant to content-based information retrieval in addition to the different applications explained in the paper.

References

1. Koller & Sahami. Hierarchically classifying documents using very few words. The 14thnational conference on machine learning. Computer Science department, Stanford University , 1997
2. Dumais, S., & Chen, H. Hierarchical classification of web document. Graz University of Technology, Austria, Masters Thesis, 2000.
3. Chien, L.-F, Huang, C.-C., & Chuang, S.-L. Creating hierarchical text classifiers through web corpora. WWW '04: Proc. of the 13th Int. Conf. on World Wide Web (pages 184- 192). New York, NY, USA: ACM Press, 2004.
4. Neumann, Günter & Sven Schmeier. Combining Shallow Text Processing and Machine Learning in Real World Applications. 1999. Available at http://www.dfki.de/~neumann/publications/newps/ijcai99-ws.pdf.
5. Sebastiani. Machine learning in Automated Text Categorization-in ACM Computing surveys 34(1), 2002, pages 1-47.
6. Y. Yang and X. Liu, A re-examination of text categorization methods, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval* (Berkeley, CA, 1999) pages 42–49
7. Tilahun G, "“ Qubee Afaan Oromo : Reasons for choosing the Latin script for developing an Afaan Oromo Alphabet”.,"*Journal of Oromo studies*, 1793.
8. Duwairi R., "“Arabic Text Categorization” ," *The International Arab Journal of Information Technology , ,Jordan University of Science and Technology, Jordan*, vol.Vo.4,No.2, April 1807.`
9. Shankar Ranganatan. Text classification combining clustering and hierarchical approaches. Computer Science and Engineering, University of Madras, Chennai, India, Masters thesis, 2001
10. G., Steinbach, M. and Kumar, V. Karypis, "A Comparison of Document Clustering Techniques. New York, USA:," *ACM Press/Addison-Wesley Publishing Co.*, 1804.