AUTOMATIC DOCUMENT CLASSIFICATION IN SMALL ENVIRONMENTS

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Jonathan McElroy

January 2012

COMMITTEE MEMBERSHIP

TITLE:	Automatic Document Classification in Small Environments
AUTHOR:	Jonathan McElroy
DATE SUBMITTED:	January 2012
COMMITTEE CHAIR:	Franz Kurfess, Ph.D.
COMMITTEE MEMBER:	Alexander Dekhtyar, Ph.D.
COMMITTEE MEMBER:	Gene Fisher, Ph.D.

Abstract

Automatic Document Classification in Small Environments

Jonathan McElroy

Document classification is used to sort and label documents. This gives users quicker access to relevant data. Users that work with large inflow of documents spend time filing and categorizing them to allow for easier procurement. The Automatic Classification and Document Filing (ACDF) system proposed here is designed to allow users working with files or documents to rely on the system to classify and store them with little manual attention. By using a system built on Hidden Markov Models, the documents in a smaller desktop environment are categorized with better results than the traditional Naive Bayes implementation of classification.

Contents

List of Figures

1 Introduction

An important goal in Computer Science is increasing the ease of Human-Computer Interaction. Creating systems that assist users by allowing them to perform tasks more intuitively allows users to limit their busy work. With the rate of computer complexity, speed and disk space increasing continuously in recent years, combined with the increased usage of digital documents, there is a greater desire for the automatic categorization and filing of these documents [32]. The system proposed here aids working with digital documents by allowing users to rely on the system to store them with little manual attention. Filing these documents is accomplished by the classification of the documents through a system that adapts as new categories or documents are added.

Text Classification (TC) is defined as determining and assigning topical labels to documents that define their characteristics [16] [4] [35]. TC is used in several contexts such as document indexing, document filtering, web filtering, search engines and anywhere where documents need sorting [32]. While there are several different approaches to TC, this project focuses on using Hidden Markov Models to classify the documents in a smaller environment. The purpose driving this is the desire to work toward a smarter desktop or semantic desktop, which aides computer users in several ways, one of which is the automatic filing of their documents. By looking at how a user has already filed their documents, or a previously filed learning set, the system then can gain an understanding of how to place the files. The importance here is that files classified as the same category, are not always placed in the same file areas. An example of this is a user filing all his documents about taxes into a folder called Taxes. Within that folder the user subdivides the documents by year. Any tax file with a specific year, needs to

be placed in that year's file, but a generic document about good tax filing ideas should be placed at the top level of the "Taxes" folder.

This thesis focuses on properly classifying documents in a smaller environment (a computer system that is self contained) while maintaining a degree of adaptability. Section 2 discusses related works and the background of Text Classification, Information Retrieval and similar smart systems. In Section 3, development of the HMM classifiers as well as other classification techniques are covered and explained. Implementation of the Automatic Classification and Document Filing (ACDF) system that is used to test the thesis is detailed in Section 4. Section 5 covers typical usage of the system from a user's perspective. Sections 6 and 7 address the results of testing the ACDF system and the analysis of those results. Finally, Section 8 covers future work ideas and Section 9 is the conclusion.

2 Related Works and Background

The project of Automatic Classification and Document Filing (ACDF) uses a combination of Text Classification (TC) techniques and Machine Learning (ML). There are many works that relate to these concepts and use different methods to achieve the desired results. The related works using Bayesian, vector based classification, techniques and agent-based approaches will be covered and other similar projects discussed.

2.1 Text Classification

Text categorization is defined as *the activity of labelling natural language texts with thematic categories from a predefined set* [32]. A formal definition from [33] is:

Definition Let C = { c1, c2, ... cm} be a set of categories (classes) and D = { d1, d2, ... dn} a set of documents.

The task of the text classification consists in assigning to each pair (ci, dj) of C x D (with $1 \leq i \leq m$ and $1 \leq j \leq n$) a value of 0 or 1, i.e. the value 0, if the document dj doesn't belong to ci.

This procedure affiliates the text with groups or topics based on quantitative information gathered from the characteristics of the items. The training set of previously classified items provides the basis of finding the important characteristics. The current main approach of building these classifiers is by machine learning or automatic building from the training set. Previous approaches included knowledge engineering which involved domain experts manually creating

classification labels. [32] is a survey of automated text classification, but most topics will be discussed in the following.

2.2 Naive Bayesian Text Classification

The Naive Bayesian (NB) Classifier is a well known probabilistic classification method. The method uses Bayes' theorem which predicts the classification of a previously unseen item, using information from a training set. This highly useful method has been used for many different types of classification and filtering such as: email classification [35], document classification [32] [23] [7] and even hybrid approaches with Hidden Markov Models [13] [11]. One of the strengths of the NB classifier is its simplicity and in many instances it performs much better than more complex classification methods such as Support Vector Machines [35] [19].

Understanding the usage of the Hidden Markov Models requires using the principles of Bayes classifier. The NB classifier is a probabilistic method that uses Bayes Theorem [35] [11] to classify an unknown document based on information from training data. To calculate the probability for a class A_j given a document B, find the maximum value of

$$P[A_j|B] = \frac{P[B|A_j]P[A_j]}{P[B]}$$

with j ranging over all the classes [35]. The probability that given a category A, document B is in that category, can be evaluated by looking at the number of distinct words in document B and summing up the probabilities that each word would be in category A_j. Bayes' assumption is that if terms are conditionally independent given category A, then the classification depends only on the values of $P[B|A_j]$ and P[A]. These are the probabilities that B comes from A and the

probability of A being chosen from the set of j classes. P[B] is ignored since all documents are considered equally possible. Although this is a simple method for classification, research shows that with the correct features it often performs better than more complex methods such as decision trees, rule-based learning and instance-based learning [35]. NB classifiers appear in a large amount of research papers as well as real world applications making them a great comparison for testing purposes.

2.3 Hidden Markov Models

Hidden Markov Models [14] are used as a tool for probabilistic sequence modeling and learning patterns. HMMs build up a path of sequence possibilities with probabilities in each path. Then they run simulations on these paths multiple times to determine mostly situations. A more formal explanation occurs in Section 3. HMMs are used in many types of text based areas of research including natural language modelling [5] and information retrieval and extraction [15]. These models use Markov chains, a type of memoryless mathematical model, which look for the likelihood of a given future event based on only the present state.

Information Retrieval (IR) [32] is the act of searching for information based on keywords or terms. The large amount of overlap between IR and TC solutions exists since they are almost inverse problems [38]. The former uses a small set of words to retrieve similar documents, while the latter uses a document and seeks similar documents to determine a classification. Using probabilistic methods for Information Retrieval is almost 5 decades old, and [26] was one of the first to use HMMs in IR. Yi [38] explains that with a query Q and a document D, the retrieval

system searches for a document d using Q. The Bayesian Formula is applied to $\max_{d \in D} P(d/Q)$ to produce, $P(d/Q) = \frac{P(Q/d)P(d)}{P(Q)}$. By using the HMMs to model $P(Q/d)$, the query Q was generated and used to interpret the IR model [38].

[14] also used HHMs for text classification by viewing documents as sequence of pages. The outputs of the HHMs are an associated bag-of-words. A function maps state realizations into page categories. A state of category is not based on just the contents of a page, but also by the other pages of the document. This can become helpful by looking at the structure of documents. For example, most research papers start with a title, abstract, introduction and so on. This allows the classifier to already know that it is classifying a research paper and so can use heuristics that are specific to research papers. Using HHMs can help further break down the classifications of documents by looking at their structures.

[38] [39] built upon the works of Miller and Frasconi using HMMs to categorize Dissertations and Medical Documents. Yi replaced the document and query formula with target category and relevant document. For each category a TC model or HMM was created. The output from these models was used to determine relevant documents. While Yi used predefined terms and information sources, this thesis creates the HMMs based on a hierarchical approach instead of information sources as well as focusing on a smaller more diverse environment.

2.4 Similar Systems

Filtering is an important part of any information organization and retrieval system. A filter or a query is applied to information with the result of specific documents or files related to the filter's purpose. This is useful because classification and routing techniques are what compose filtering [6]. Clack creates an

automatic document classification system for businesses by using filtering. This system reroutes information from a central database to multiple users with different profiles, by using evolving classifying agents that filter the data. These agents are initially trained (supervised learning) by looking at the user's current information focus. It examines web bookmarks, emails, and certain directory structures to form a stable first solution classifier. After that Genetic Programming is employed as a learning mechanism for some time before the user can access the data. This is a similar approach to ACDF in terms of the initial learning setup and approach to longevity by using evolving classifiers. It also seeks to incorporate natural learning. Similarly, they encountered difficulties with documents containing text information, that both distracts and misleads the learning mechanisms.

In an effort to overcome these issues, Cognitive Assistant that Learns and Organizes (CALO) [3], a project lead by SRI International, focused on development of a smart desktop. The CALO project goal was to automate interrelated decision making tasks that have resisted automation and allow them to react appropriately to situations that are unusual. While CALO seeks to create a desktop environment which assists users in many different information handling, this thesis focuses on the particular aspect of assisting the automatic filing of documents for the user.

Email classification systems [35] are similar to the ACDF system. They are continually receiving new text-based documents and working to classify and extract important information out of them. That particular system also uses supervised learning on a large set of training emails and then turning them into vectors. The classification system is built using T-Route and T-Trans vector comparison algorithms, several term weighting schemes, Latent Semantic Anal-

ysis (LSA) transformations and some post processing. T-Route and T-Trans are two similar ways of comparing vectors that use a term to document matrix [9]. T-Route gathers terms that are destined to have the same route in the matrix and places them together. T-Trans instead, first gathers the groups of terms that are duplicates and discards the extras before places them in the matrix. Then comparisons are made using the matrix to determine similar routes. LSA is based on using the technique of singular value decomposition (SVD) to find the lowest error representation of the matrix in a compact subspace. This can be seen as smoothing the term or document [9]. This system discusses that for email classification, the vector based classifiers did much worse than simple Naive Bayesian classifications with Bellegarda word weights. Bellegarda weights combines the importance of a word globally within a set of documents, with the local importance of that word in a document.

Work relating to organizing and classifying into hierarchies [4] involved separating feature words from noise words at each level of the hierarchy. This multi-level classification allows the searching of topics instead of just keywords. At each level a context sensitive signature and feature selection is created and then focused to cut out noise and stop words. Again they find Naive Bayesian techniques to be almost as accurate as more complex methods and faster.

Other systems with similar purpose are DEVONThink [10] which seeks to make an all inclusive information gatherer and organizer. It gathers all text based documents and gives one interface for viewing and grouping. It sorts, classifies and shows relationships between documents automatically, but it falls short of learning from the user or inferring any sort of context from the documents. The NepoMuk [29] semantic desktop system seeks to combine personal data management and collaboration into a tool for social relation building and knowledge

exchange. It annotates and links information on the desktop across media types, file formats and applications using semantic web data structures. While this thesis does not seek to build a full fledged semantic or "smart" desktop, it does seek to create a useful part of that goal. Elimination of tedious work is often a helpful and useful goal [6].

3 Development

The ACDF system is composed of two parts: the structure which keeps track of the meta data of the documents and the classification algorithm. This classification algorithm is the most important aspect since it does all the work. HMMs provide a novel approach to the situation of classifying documents in a small environment. This is compared to Naive Bayesian TC, a popular classification method that is often used as a comparison for testing. The structure of the system is built around using nodes (or directories) as the categories themselves. For example, a user might have a folder labelled Cars, and within that folder, two other folders labeled: Ford and Honda. The system creates a node for each folder and treats them as three separate categories. While this removed the ability for items to have multiple tags or classes, this allows simple classification in a hierarchical manner. Using the previous example, when a user is attempting to store an article about a Jeep, the system will place it in the Cars folder. If the user wants to start filing more articles about Jeeps, they only need to create a new folder and place the Jeep article in the folder. The system now has 4 categories and any new Jeep articles submitted for filing, should be placed now in the Jeep folder. Using the nodes as a hierarchical method of classification allows for the classifier to focus specifically on each section of the file tree. While there are advantages to both to branch classification and this version of node classification, using the node classification allowed for faster running times since less data is necessary for each classification [21].

3.1 Hidden Markov Models

Hidden Markov Models (HMM) [13] [38] [39] [31] [26] are often used as statistical models for speech processing, pattern matching and more recently text classification. These HMMs are also known as probabilistic functions of Markov chains. A set of Markov chains can be described as a set of states, each with output values, and each of these states has probabilities of movement between them. For example, consider a system composed of N states which is in one of those states at a give time q_t where t = 1,2... and so on. At regular intervals the system changes state (although its change my be staying in the same state) according to the probabilities within the system. Since each change of state is dependent only on the previous state, it can be shown as:

$$P[q = S_j | q_{t-1} = S_i = q_{t-2} = S_k = ...] = P[q = S_j | q_{t-1} = S_i]$$

This system is stochastic, meaning probabilistic and non-deterministic, so that the output given is randomly determined on each pass through the system. Using the above equation allows the construction of the state transition probabilities a_{ij}.

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i] 1 \leq i; j \leq N$$

The probability of going to state j comes from where it was in the previous iteration of state i. In a regular Markov model the states and the outputs are visible. An example that [31] gives is that of a 3 state Markov Model using the weather. The three states are:

State 1: rainy (or snowy)

State 2: cloudy

State 3: sunny.

The probabilities of movement between these states is described as this matrix:

$$A = a_{i,j} = \begin{pmatrix} .4 & .3 & .3 \\ .2 & .6 & .2 \\ .1 & .1 & .8 \end{pmatrix}$$

A prediction can now be made for the probability of the weather over the next t days. For example, if the start state at $t = 1$, is sunny then we can predict the likelihood of the next 5 days being sun-rain-rain-cloud-sun. This is shown as $\pi = 1(thestartingstate) * A_{33}(probabilityofstatestayingsunny) * A_{31}(statemovingtorainy) * A_{11} * A_{12} * A_{23}$. The probability is: $1 * (.8) * (.3) * (.4) * (.2) * (.1) = 0.00192$. While these models can be used to predict the probabilities of events happening in certain orders, the practical use of these models comes from when the states are hidden from the user.

These are known as Hidden Markov models which have invisible states, but visible outputs. A *HMM = (N, M, A, B, π)* contains five components [38]: 1) N is the number of states in a model. Each of the documents in a category is treated as a state. 2) M is the set of output symbols. These are the words contained in each document with stop-words removed. 3) A is the probabilities of transitioning between states or documents. The probabilities of transitioning between documents are considered equal, i.e. if there are 4 documents in a category, then the transition probability is 1/4. 4) B is the emission probabilities of each word at that state or document. That probability is defined according to how much usage that word has in the document.

$$B_{ij} = \frac{sum(b_{ij})}{total(B)}$$

5) π is the starting probabilities which are also equal for each document. These are more useful in applying to real world situations where only the outputs are visible as they allow for the statistical predictions of sequences of events. An example from the GHMM (General Hidden Markov Model) libraries tutorial uses HMMs to figure out if dice are loaded or unfair [17]. The states of the dice are either fair or loaded. The observation symbols are their faces: 1,2,3,4,5,6. A fair die will have a 1/6th chance for the rolling of each of these numbers, while an unfair die will have a higher chance for certain numbers to be rolled. Using a Viterbi-path, a sequence of states Q maximizing the probability $P[\frac{Q}{observations, model}]$, a sequence of events can show if the die rolls are natural or if it is likely that they are loaded.

The ACDF system is influenced by Yi [38] [39] who uses a HMM with 4 states: start, end, classified documents, and subject specific documents. The first two states are dummy states.

Each of these HMMs in each category produce a string of words based on their probabilities which are used for comparison. A visual is shown in Figure 3.1. The three documents compose the category, and all the information is used to produce a HMM. The document being classified is then compared with the HMM produced (relevant) document for similarities.

An visualization of a classification HHM for each category is shown in Figure 3.2. Each document in the category is represented as a state in the HMM. The probabilities of moving between the states ($S_{i,j}$) are equivalent. The output probabilities ($O_{i,j}$) are shown pointing to different words with X standing for all non-shown words. These different states each have a probability of producing one of these words when the counter is moved to them. If a document does not contain a word, then its probability of producing that word is zero. The probabilities of

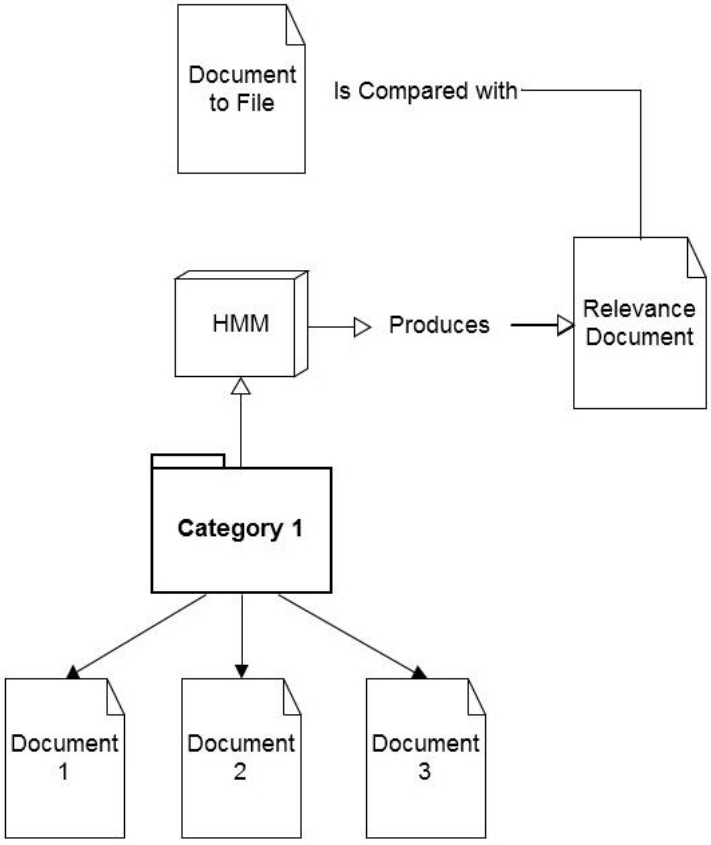

Figure 3.1: HMM Usage

using a word are determined by how often it appears in the document. To produce the output document the counter is moved N times, which creates a document of N length. Using this method of HMMs can be more accurate in smaller environments than NB because of the need for less training data. For example, consider a folder filled with music. The meta-data on the music contains the name of the artist, album title, producer etc. The NB classifier has only a small amount of information to test similarities with, while the HMM model takes the small amount of data and produces a larger weighted bag of words for testing similarity.

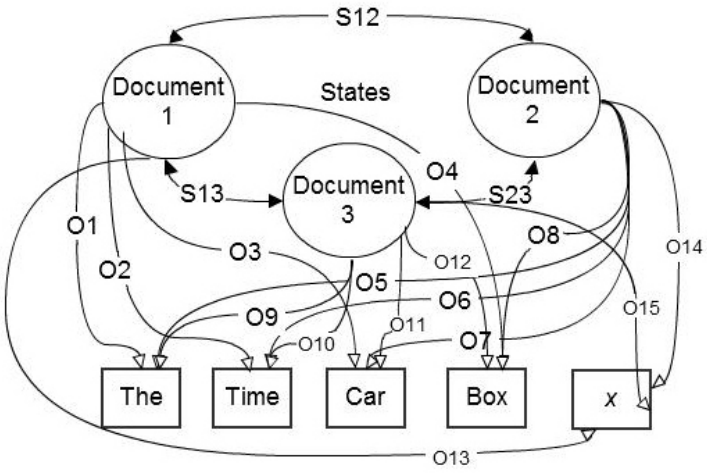

Figure 3.2: Example of Classification HMM

3.2 Other Classification Techniques

This section discusses other modern classification techniques and explains why only NB and HMM classifiers are selected for this thesis. There are several different classification techniques which are described in the related works section. Some of these types are support vectors machines [19], rules-based [7], agent based [6] [16] and neural network based classifiers [1].

Support Vector Machines use supervised learning to analyze data and detect patterns. A formula is created which accepts the training data and conceptually plots each item on a N-D plane. The SVM classifies documents by taking the input of a document and puts it through an algorithm which plots it as a point. Lines can be drawn through this plane to determine which category the item belongs. The SVM is an optimization procedure to find the best line at each step. For multiple categories the SVM runs multiple times as each time it determines the best line and splits the remaining items determining the closest class. Figure 3.3 shows different lines which the SVM draws to split the categories. H3 fails to

determine a class while H1 and H2 both properly split that class. H1 does it with a smaller margin, meaning that the line splitting the two classes is very close to some of the plotted items. The line H2 has the highest margin as it is furthest away from both classes.

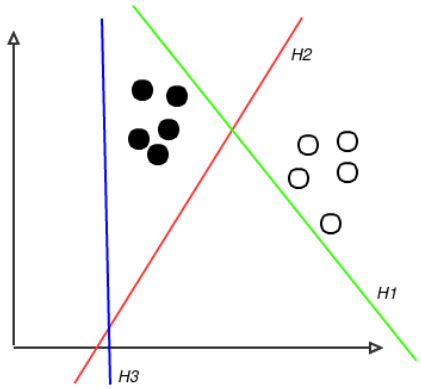

Figure 3.3: Classification with Support Vector Machines

Often the support vectors did worse than the NB classifier such as in the case of email categorization [35]. NB also often showed better results than other complex classifiers [21] [4]. Many of the other classifiers used the basis of the NB logic to create their classifiers [2] [14] [11] [38]. The HMM classifier used here is also based on the NB logic. The NB classifier is chosen since it is often used as a baseline or standard, and is the basis of the HMM classifier.

Rules based classification methods use rules to determine usefulness of words for classification. One paper used rules to create a "context" for words [7]. The context was a group of words that were related to that word, and when they were used, increased the importance of that word. The results in the end showed it to be slightly more useful then NB. For this project though, the increase in

accuracy is not great enough to warrant usage. Also a much larger scale program is required to properly use a rules based classifier [7]. Increasing the complexity of the system for minimal increase in accuracy is unnecessary; implementing rules based classifiers as a test comparison was deemed not useful for the purpose of this thesis.

Agent based classification uses agents to classify documents. An agent is a software program which operates autonomously, emulates intelligent behaviours and completes tasks for its user. Multi-agent systems are used in [16] for classification by giving each agent a feature selection ability. Then documents in the process of being classified are turned into document vectors with terms weighted using the Term Document Frequency/Inverse Frequency (TDF/IF) model. This model seeks to figure out the important words be looking at how many times a word appears in a document making it important, versus its frequency making it less important. An example would be the word: 'and' which appears quite a lot in this paper, but its frequency in almost every sentence makes it unimportant. In contrast the term: "HHM" would appear much more important since it occurs multiple times, but with limited frequency. The agents then run their feature selections on the document vectors and if a similarity score exceeds the threshold set, then the documents are classified. This project does not include an agent classifier since often they are a combination of agents and the usual classifying techniques. A future work idea is the combination of agents with HMM powered feature selection.

Neural networks are computational models of biological networks and have been adapted for use as classification tool [1]. They use artificial nodes or neurons as feedback connections allowing the networks to process sequential information which is especially useful for grammar. Since this thesis does not focus on gram-

matical structure or in depth context, neural networks are not used as a test case.

4 Implementation

The ACDF system is implemented using the Python language on a Intel Core Duo 2.2 GHz laptop. Ubuntu 9.04 is the version of Linux being operated. The structure of the system is a tree node system. The nodes, called FileNodes, contain the information of each directory including, parent directory, all files in directory, all children directories and any information necessary to create the classifiers for each node. Figure 4.1 shows the structure of each node.

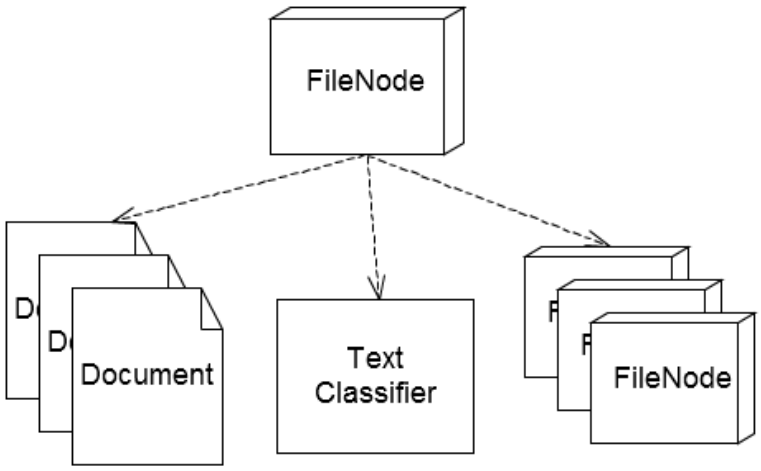

Figure 4.1: Structure of FileNode

4.1 Structure of each File Node

While each FileNode has the same basic structure, there are differences depending on the type of classifier being used. When a node is being created, the data and objects relating to their classifier are created. Each node always gathers a representation of each document by gathering a list of words from the docu-

ment. This list is sorted first by the number of times each word is used. Then all the words that are shorter than 3 letters are removed. This is to remove words such as "to" or "a", which add little to the context of a document. After that all the stop-words, which are words that similarly don't add anything to the context, are removed. The stop-words come from an English list and a stop-word list provided from the Reuters corpus [36] [28]. Lastly, all punctuation and numbers are removed. From this final list the several hundred most seen words are then used. For the simple classifier, this is all the data that is necessary, as when classifying, each node just returns the amount of words found in this list. The NB classifier uses this list of words in a similar way, where each category is trained using the list of words from each document in that category. This NB classifier is created at the root node for the whole tree. The HMM classifier creates and adds the HMM to each FileNode, while the Hybrid both creates the root level NB classifier and a HMM at each FileNode. The difference here is the the NB classifier trains using the data provided from the HMM.

4.2 Hidden Markov Model

This project uses the HMMs simulator developed and provided by Alexander Schliep's group for bio-informatics at Rutgers University [36]. The HMM at each node is created by first grabbing all the documents at that node. Each document has access to the words list, which was previously discussed, which has words which are descriptive of that document. The HMM's alphabet consists of every word in the node. The probabilities of moving between the states in the HMM are set equal for each document. This means that with 4 documents, the chance of moving to another state or staying on the same state is 1/4. Then the emission probabilities for each state are created by using the frequency distribution on each

word. For example, if the word "zebra" was used 15 times in a document which had 600 words, then the emission probability would be 15/600 or 0.025. The last part of the HMM is the π, or starting probabilities which are also set to equal for each document. When the HMMs are created, in theory it is the possible to produce an output that contains the words most related to those documents.

4.3 Usage of the System

When a node is created at a certain folder, it collects all the information and then creates nodes recursively for each of the children directories. A background process checks to see if any changes happen to a directory, and when they do, a new file node is created in that branch which automatically refreshes the branch. As files are being categorized, the tree adapts its classifiers at each location accounting for the new information.

Files and categories/directories can be added manually like normal. To use the automatic filing, a particular directory is selected as the bin folder. Items placed in this folder are classified if possible, and moved to the appropriate location on the hierarchy. Before this happens a daemon must be run in which the type of classification is selected. As explained before, when a node is being created, the type determines the kind of data that is saved. There are four types of classifiers being compared. The simple version finds the words that are at least a certain length and appear most often in each category. When classifying the same top selection of words in the comparing document is tried against every category and the category with the most words in both is selected. The NB version creates one classifier for the whole tree structure. This uses each file as a bag of words with the category being the directory name. When a new file is added it is run against

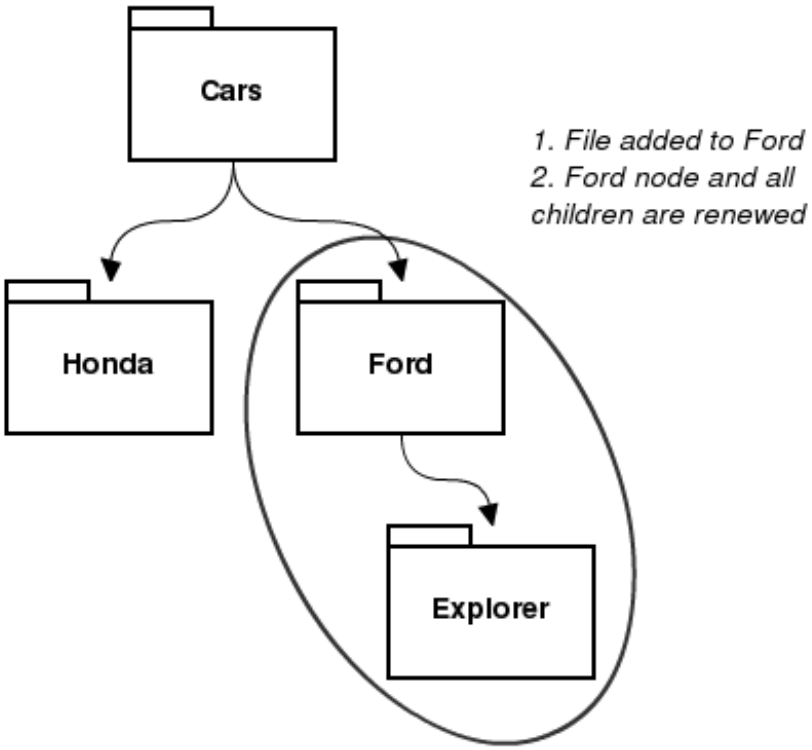

Figure 4.2: Adding Files causes a Branch Renewal

the classifier and the category with the greatest percentage is selected. The HMM classifier contains a HMM at each node. This HMM creates states from each of the documents in the node, and uses the bag of words as output symbols. When classifying a document is produced at each node and compared to the comparing document in a manner similar to the first method. The selected category has the most similar wording. Lastly the HMM Bayesian classifier combines the two, by using the HMMs to create a document at each node and then using the Naive Bayesian classifier on the created documents.

4.4 Testing

The structure of the testing takes two folders, one which contains the root directory with all the training files and a mirrored directory that contains the files for testing. When the test is run, the program goes through each file in the test directory, categorizes it and then compares its destination with where it should end up in the mirrored directory. All this information is sent to log files which then tally the results at the end and compare the different methods of classification.

5 Usage Scenario

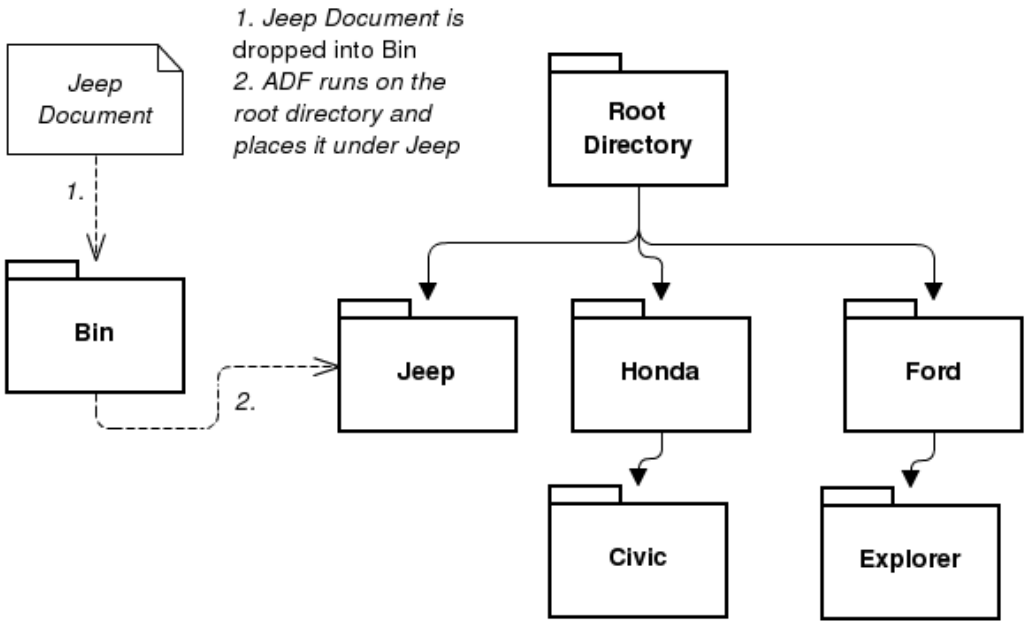

Figure 5.1: Diagram of Usage Scenario

The final system (ACDF) supports a user by automatically classifying and categorizing files. The following is an explanation of how the typical user interacts. First, the user sets up the ACDF system by choosing the root directory, i.e. C drive or whatever folder area they want document classifying. This directory's folders are now seen as categories and incoming documents will be stored accordingly. Next, the user chooses an empty folder and their bin. This is where files are dragged for classifying in the root directory. After setup, the user runs the ACDF program in the background. With ACDF running, the user then drags newly received files and drops them onto the bin folder. The files are then clas-

sified by their content and placed in the location that the user would most likely have placed them, based on the root directories current structure.

Figure 5.1 shows the example of placing a document about a Jeep into the bin and then having the file moved. If the system does not find a reasonable location to place, the system is meant to ask the user to specify a new category or show it where it should be placed. This part of the program was not implemented since it was not useful for testing the classifying capabilities of the HMMs.

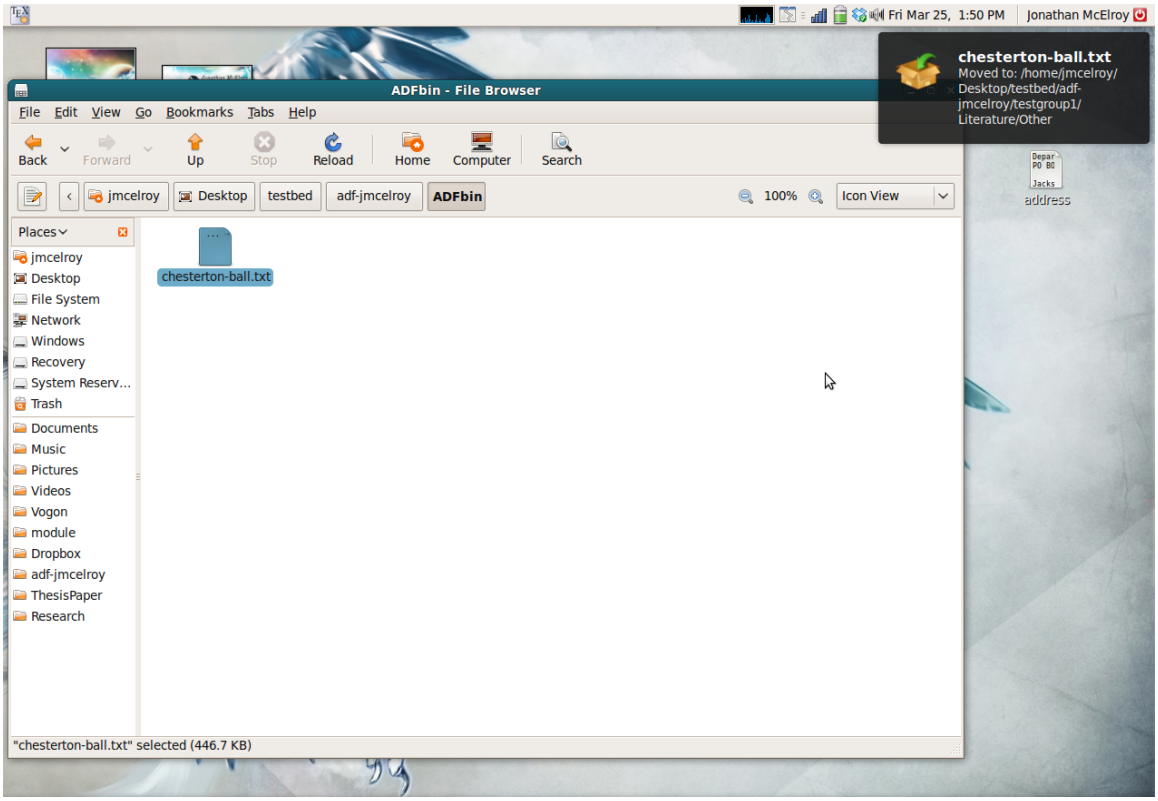

Figure 5.2: Screen Shot of File Being Placed

Figure 5.2 shows a screen shot of a filing being placed in the filing bin. The bin is called ADFbin and a notification displays to tell the user where the file has been placed.

6 Results and Validation

For testing, one of the primary focuses of this thesis is to see how HMMs contribute to increased classification accuracy in the smaller environment of a desktop. There are three main test groups. The first is the small desktop environment. The second group is made up of positive and negative movie reviews which tests if the classifiers can determine words that are positive or negative. Lastly, the system is tested on a large amount of Reuters documents that have defined categories. Both the Movie Reviews and the Reuters test groups are run with test sets of differing sizes. The point of the different sized subsets is to see how well the HMMs performed based on the size of the training data. For each test group the tests are run 20 times, since the HMM is non-deterministic and returns slightly different results each time. This allows the formation of an average result for each of the test groups and classifiers.

6.1 Small Desktop Environment

The first test is on a smaller environment set up to represent a normal desktop, filled with documents and music meta-data. This test group contains documents relating to vehicles, government documents, classic literature, Reuters news reports, as well as some documents in other languages. Musical meta data was represented in text files with the data that could be taken from music files such as artist, album, producer, etc. Since there is a smaller amount of data to train the classifiers with, the hypothesis is that the the HMMs will be more successful than the NB classifiers.

A graph of the results are shown Figure 6.1. The HMM performed better than the other classifiers by correctly categorizing the files 96.25% of the time.

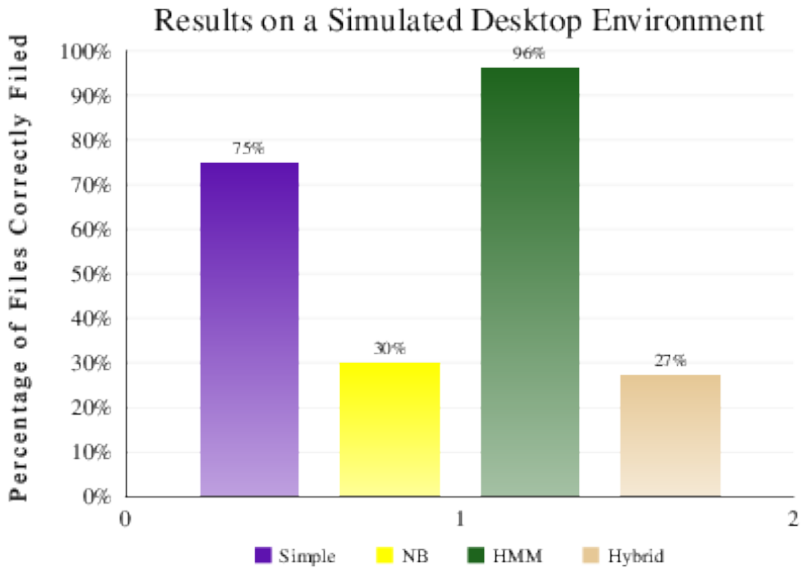

Figure 6.1: Desktop Environment

Both of the NB classifiers did poorly. This is probably due to the smaller and much more varied amount of training data. The simple classifier did well, but the HMM did better since it attributed a stronger weight to the words that appear more. The NB classifier seems to need a much larger set of data as well as a more intelligent word chooser. The hybrid suffers from the same problem as the NB classifier, in that there is too little training data.

6.2 Movie Review Testing

The second test focus on a set of documents from the nltk corpus which are movie reviews [28]. This is an interesting set of data because there is a lack of keywords that really explain whether a document is positive or negative in nature. An example is that the word 'good' could be a key word in both positive and negative since it could be preceded with a 'not' for negative. Although the

different classifiers are just looking at keywords, this test is checking to see if they can classify the tone and meaning of files. Three sets of differing sizes are used for testing. The smaller test set consists of 14 files being tested against 24 training files, with the larger test consisting 36 files being tested on a set of 600. The full set is 1,918 files with 70 files used as tests. The files were previously classified based on the first rating perceived from the reviews. Anything over approximately 65 out of 100 was placed in the positive category.

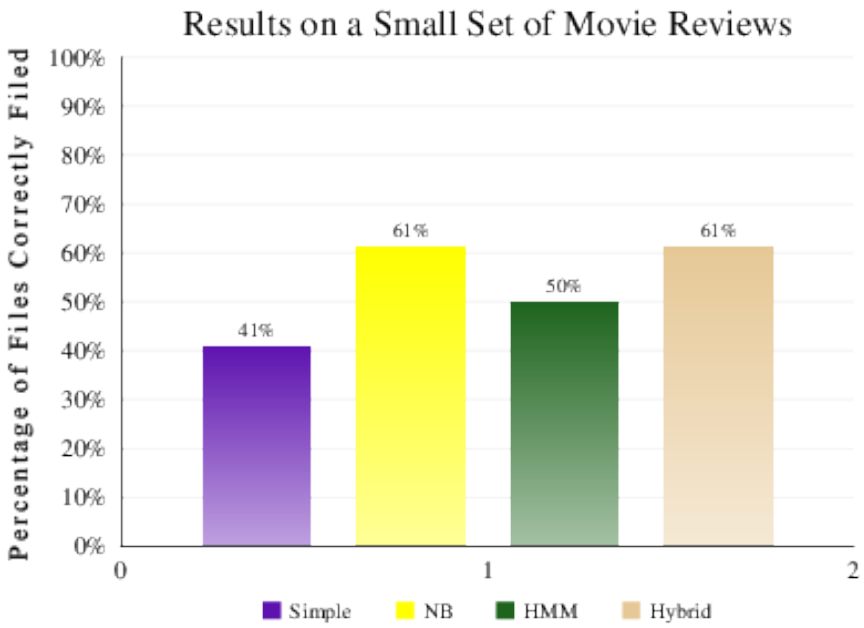

Figure 6.2: Smaller Set Movie Reviews

The results for the smaller set of Movie Reviews are shown in Figure 6.2. The Large and Full sets are in Figure 6.3 and Figure 6.4.

In this experiment the results in the smaller and larger sets for the NB classifiers were much better averaging around 61% correctness while the HMM classifier was around 50% correct. Interestingly, these stats remained almost the same be-

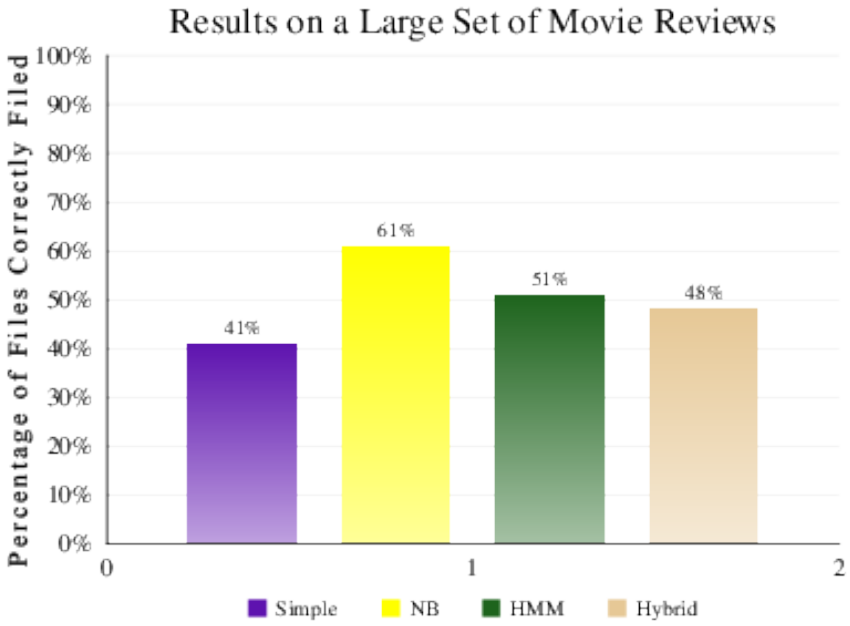

Figure 6.3: Large Set Movie Reviews

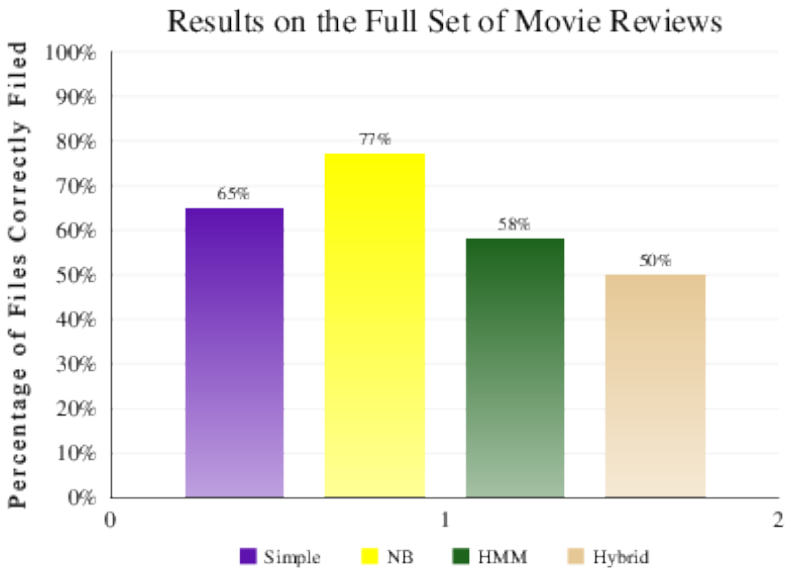

Figure 6.4: Full Set Movie Reviews

tween the small and large sets. Only the hybrid had a different result. In the larger set it had almost the same number of correctness as the regular NB classifier. The simple classifier also stayed the same. The results changed more for the full set, which had all the classifiers doing better with more training data. The NB classifier, again, came out on top with 77% while the simple classifier increased past the HMM to 65% correctness. The HMM and Hybrid received 65% and 50% correctness.

6.3 Reuters News Corpus Testing

The Reuters corpus has been used as a standard for text classification for many years [32] [34] [21]. A modified format of the Reuters corpus provided by the University of Torento, is used as this test set [27] [36]. Again, the test were run on both a large and small set of the corpus.

The smaller set had 85 testing documents, with 197 training documents. There were four categories overall. As seen in Figure 6.5, the HMM once again did much better than the others scoring a 75% correctness rating. The NB and Hybrid had 51% and 47% respectively. These results seem to show that the HMM is much more useful when there is a smaller amount of training data.

Figure 6.6 shows that the results for the large set of the Reuters corpus are much different. The simple classifier does the best at 25% followed by the NB and HMM at 25% and 18%. The Hybrid did exceptionally terrible at just 1% complete. Some of the possible issues with this corpora [36] arise from the fact that some of the categories are much more weighted then others. There are 91 categories in this set with 7,862 files total, but 7 of the categories contain around 50% of the number of files.

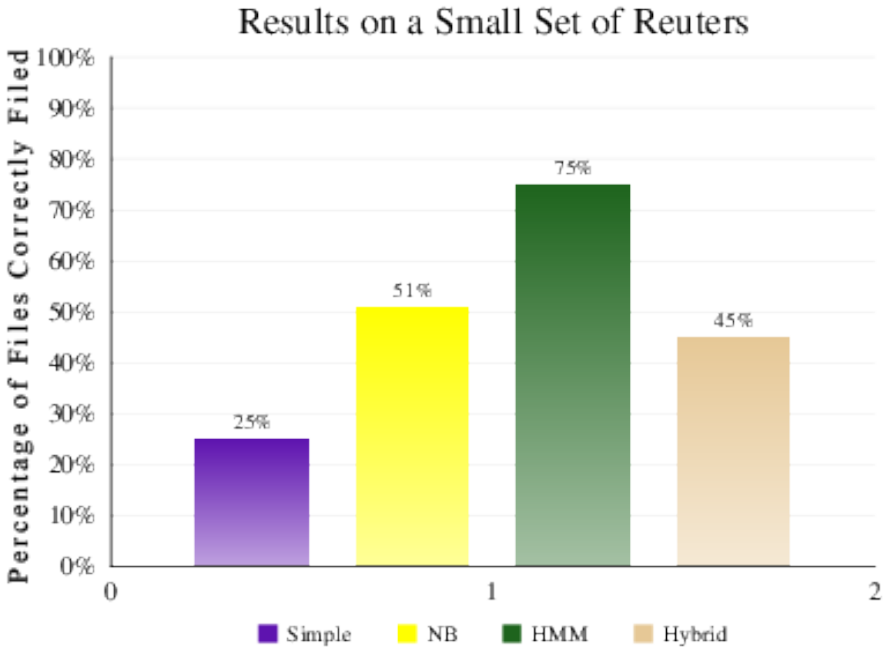

Figure 6.5: Smaller Set Reuters

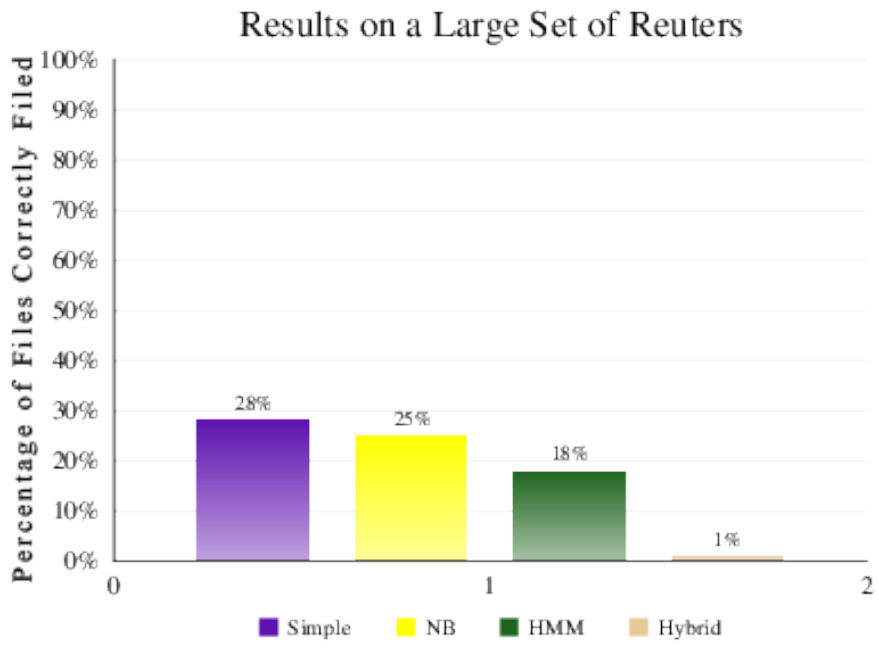

Figure 6.6: Large Set Reuters

7 Analysis

This section covers the different aspects of the results requiring analysis. The first being, the advantage/disadvantage of using the HMMs. Secondly, the NB Hybrid's failure is discussed. Lastly, the most important learned ideas from this project will be discussed.

In current scholastic news, the categorization and search on the Internet is often the most important focus of TC [32] [35]. This comes from the incredibly large amount of documents and meta information needing classification for increased usefulness. This thesis focused on finding classification methods that work better in a much smaller environment, with a purpose of contributing toward a more useful semantic desktop. Although large advancements are being made in search, classification and automation of findings on the Internet, local machines for the most part are still being integrated into this idea of increased automation. This could be for several reasons, even psychological reasons [37], where users like to remain feeling in control. One of the possible issues for local machines is that there is less information locally than on the Internet, which could contribute to machine learning or training.

7.1 Using HMMs

The main purpose behind this thesis was to create a system that would allow users to file (classify) their documents automatically with little manual attention. The HMM is proposed as a useful tool to assist with this task. One of the main differences of using the HMM, as discussed earlier, is its non-deterministic output. Since this probabilistic output does not guarantee the same output every time, the issue arises that results could be inconclusive from the range of responses.

However, the results mostly show that the HMM classifier performed similar classifications each time, nullifying this issue. HHM classifiers also act differently than other classifiers by using produced data for their classification methods, instead of using the actual information as training data. This is attributed as a strength and contributes as to why it works so well on the smaller environments.

The HMM performed the best in the small desktop environment, with much larger margin of correctness compared to the NB classifier (96% to 30%). This result most likely comes from the difference in training data necessary for each classifier to make a more accurate classification. With less words in each category, the NB was unable to make most of the correct classifications. The NB classifier works by taking the document to classify and calculating the probabilities of each word existing in each category versus existing in the whole file tree. Since there were less words to differentiate each category, the NB classifier's results are much worse. The advantage for the HMM classifier comes from producing a document that intrinsically contains what are deemed the most important words, i.e. those words which appear the most within a category.

For example, Figure 7.7 shows a scenario where we classify a small document C using both NB and HMMs. Document A contains the text: "The dog was riding in the car. This dog was named Bo." and is in a category about short stories. The word "dog" appears in 1/6th of the document. Lets compare this to another document B, in a category about dogs, which contains the word "dog" 7 times. B has more words overall leading to the word "dog" only appearing in 1/15th of the document. Now let's look at a third document, C, which contains the word "dog" and requires comparison to the first two documents. The NB classifier might incorrectly classify the document C as being in the same section as A, since the word "dog" has a higher percentage of appearances in A. With the HMM

producing text though, it is likely that document B would produce a comparison document with the word "dog" more than the other words. This comes from B containing a higher ratio of the word "dog". The produced comparison document would correctly place document C with document B. This is one explanation of why the HMM's were so successful in the smaller environment.

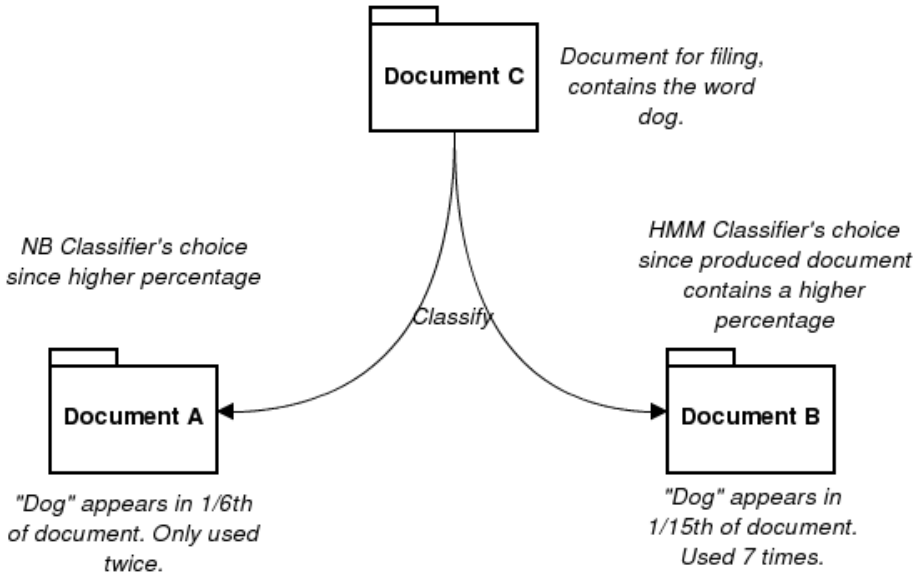

Figure 7.7: Example of differences between possible HMM and NB classifications

7.2 The Failure of the NB Hybrid

The hybrid model of NB and HMM was an obvious failure. In each test case, the hybrid model did worse than the NB classifier and only bested the HMM classifier once. It particularly failed in the test case featuring the large test set of Reuters documents. When testing the NB hybrid, one of the variables that was tested was output size. The output size is the number of words that was produced by a HMM and then used as training data for the NB classifier. Initially, it was

thought that having a much larger output size would lead to a greater accuracy as is usual with any kind of training classifier. In testing though, it appeared that having a smaller output size worked much better. Again, in future work, a more robust and less naive feature selection might lead to better results for the hybrid model.

7.3 Important Ideas

The success of the HMM in a smaller environment might seem unnecessary in a web-centric technology age, but there are some practical uses and adaptions. The main idea is to take the usefulness of the HMM, and adapt other classification ideas to suit that strength.

One example is to combine the HMM classifier with a clustering algorithm [12]. Clustering algorithms take in a group of documents or files and use a similarity metric to determine clusters or groups of relevant or similar documents. After the clusters are formed a version of the HMM classifier could be used on the clusters to then classify within that smaller set of documents. The results might allow for an increase in speed since the clustering would find similar categories and then the HMM would run on that subset. Figure 7.8 shows a visualization of this adaption.

There is also a need for search and classification within websites that have large amounts of user generated pages [24]. Most websites are hosted on servers which are in essence dedicated computers which wait for other computers to request information [8]. These smaller environments could benefit from using the HMM classifiers for classification of generated pages. The classifier would treat each page as a document and the users would create the initial categories. This

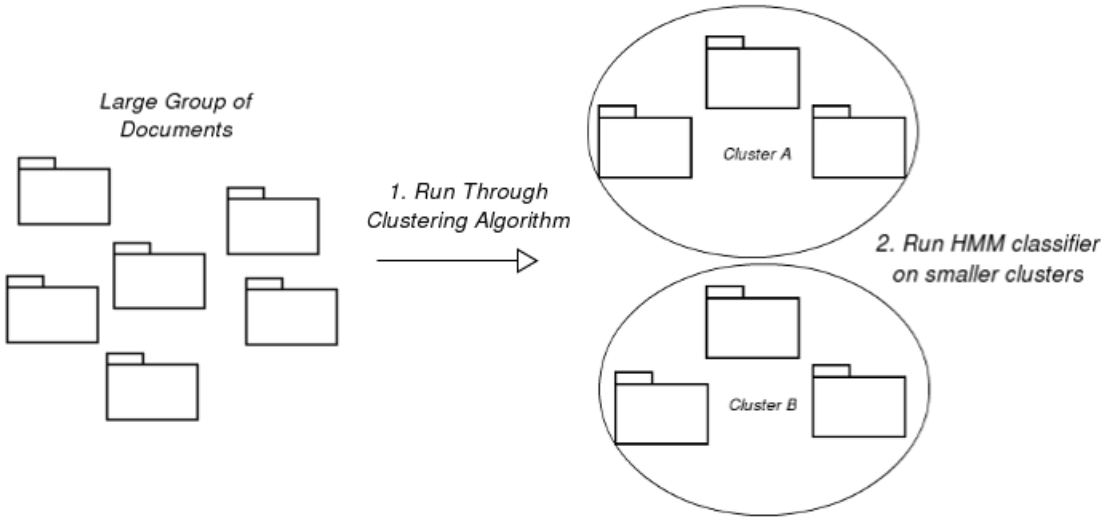

Figure 7.8: Classification using Clustering

would allow for automatic and adapting classification and take into effect users editing, adding or commenting on each page.

8 Future Work

There are many different improvements and ideas that currently have not come to fruition in this thesis. This project seeks to focus on a smaller part of creating a better desktop environment, and the following are some ideas for expanding this system and increasing its usefulness.

8.1 Moving Past Human Interaction

One of the goals in this thesis is to contribute to HCI. This is accomplished by adding more automatic functionality to something that users do rather well: categorizing or classifying. While users can easily view documents and decide how they should be stored, a computer has a much more difficult task of correctly guessing at what the users want. The computer though has the ability to do this much faster and to run all day and night. The current system still requires someone to create the basic structure and to start the classification themselves. They are also required to keep track of the end results of the classification and correct any misplacements as well as adding any new categories manually. Future work for this project can focus on the system accurately determining when a new category needs creation. This is a very large issue by itself. First, the system would need to measure the similarity of documents [22] [18] and determine when a document meets a low enough similarity to all the other documents, that it requires a new category. Then it needs to determine an appropriate name for this new category. After that there is the issue of storing the new category properly within the hierarchy. The issue is when a document should create a new sub category within a new category. Overall, the future work required here is very significant, and success in this area would be greatly beneficial.

8.2 Adding More Hierarchical Structure

Using a hierarchical system for test classification is an important issue [22] [21] [24] [34] for future consideration. In the current system the hierarchical structure is only a slight focus as each node of the tree is seen as an independent category. One possible idea that makes sense is to get a similarity rating for each branch from the root and then choose the most similar. At the top level, each node returns a score which is produced from viewing that node and the children documents as one document. The classification then runs on that branch, possibly increasing speed as well as accuracy. That change would be reasonably easy, but is outside the scope and focus of the currently thesis. The downfall for using this simple fix, is that documents would need to compare their children nodes against themselves. A car example will help illustrate this issue: A folder labeled Cars has two child folders labeled Ford and Jaguar. Any file which should land in the child nodes will likely do so, but the problem comes when a file is meant to be labeled just Cars since it is either not referring to a Ford or Jaguar or it is speaking of something relevant to all cars. The children nodes will be compared to the Car node, which will likely contain more files leading to the possibility of misplacement. In the current system each of the three nodes are valued equally and there is a higher likelihood of correct placement. The issue of hierarchical placement is an important one in smart systems and could use more focus.

8.3 More Intelligent Word Weighting

The word feature selection for this thesis was relatively naive. After stripping stop words and non-alphabetic characters, the selection was based on which words were used the most in each document. There are several different word

weighting techniques that could be applied to this thesis for future work. Other papers experiment with Inverse Document Frequency, Carpenter weighting, Mutual information scoring, Bellegarda weighting [35] as well as Information gain [13]. TD-IDF or term-document inverse frequency, seeks to find terms with rare frequency and increase their weight since they are more likely to be specific to a class. Mutual Information looks at two different objects or classes in this case and infers which keywords being used to classify one of them will also lead to a misclassification of the other one. Bellegarda weighting combines a global weight of a word from all words with the weight of the word in the local document. Although these techniques would certainly increase the accuracy of the current classifiers, they were not necessary for the extent of this project. Having better feature selection would have increased all the classifiers accuracy. The primary goal though, was to observe the usefulness of HMMs for classification in a smaller environment, which is appropriately tested by checking the HMM's performance against the standard NB classifier.

8.4 Expanding Beyond Just Text Files

Since the end goal of this thesis is to contribute to a smarter desktop environment, a very important area for future work is branching away from text files. Although musical meta data was included in this thesis, there are many other files, such as images and other media which need to be included in the classification. Much of that future work though, involves converting those files into meta data which accurately describes how they should be classified. This is beyond the scope of the current work and into a very different area of focus. Once those files are represented in text, the current system would be able to classify as normal.

9 Conclusion

Creating an adaptive and helpful desktop is an important goal. Software makes people's lives easier by removing tedious work and assisting user's skills. This thesis contributes to this goal by examining a useful method for automatically filing documents. By creating a self adapting structure that is able to classify documents and place them accordingly, the burden of filing documents by hand is removed. The search for a smarter desktop continues as more advances in technology continue.

In many ways, the HMM classifier is proven useful in a smaller environment. This was the original theory, based on the HMM's ability to produce documents using the probabilities of words within a larger set of documents. The results showed that the smaller the amount of documents total, the more correct the classifications were for the HMM's classifier. This does not indicate that less categories are necessary, only that less documents overall. Also another area of usefulness of the HMM is shown when it comes to filing documents with limited meta-data, such as music or images. Since these items contain much less information, the other NB classifiers were unlikely to correctly label them. The HMMs produced larger documents from this limited meta-data leading to better categorizing. The future work section describes how this system could be easily extended or bettered. The main idea of using HMMs in smaller environments behind this thesis is successful.

In the larger scale, such as the Internet or an intra-net, the currently implemented HMM classifier requires more work to see an increase in usefulness. The simple NB classifier did much better when it came to large documents being categorized. This indicates that if there is a way to break a classification problem

down into smaller classification problems, that HMMs are a good possibility to approach for an increase in accuracy.

Bibliography

[1] Arevian, G. 2007. *Recurrent Neural Networks for Robust Real-World Text Classification.* In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI '07). IEEE Computer Society, Washington, DC, USA, 326-329. DOI=10.1109/WI.2007.91 http://dx.doi.org/10.1109/WI.2007.91

[2] Baker, L. D. and McCallum, A. K. 1998. *Distributional clustering of words for text classification.* In Proceedings of the 21st Annual international ACM SIGIR Conference on Research and Development in information Retrieval (Melbourne, Australia, August 24 - 28, 1998). SIGIR '98. ACM, New York, NY, 96-103. http://doi.acm.org/10.1145/290941.290970

[3] *Cognitive Assistant that Learns and Organizes* http://caloproject.sri.com/

[4] Chakrabarti, S., Dom, B., Agrawal, R., and Raghavan, P. 1998. *Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies.* The VLDB Journal 7, 3 (Aug. 1998), 163-178. http://dx.doi.org/10.1007/s007780050061

[5] Charniak, E. *Statistical Language Learning.* Cambridge, MA: MIT Press. 1993.

[6] Clack, C., Farringdon, J., Lidwell, P., and Yu, T. 1997. *Autonomous document classification for business.* In Proceedings of the First international Conference on Autonomous Agents (Marina del Rey, California, United States, February 05 - 08, 1997). AGENTS '97. ACM, New York, NY, 201-208. http://doi.acm.org/10.1145/267658.267716

[7] Cohen, W. W. and Singer, Y. 1999. *Context-sensitive learning methods for text categorization.* ACM Trans. Inf. Syst. 17, 2 (Apr. 1999), 141-173. http://doi.acm.org/10.1145/306686.306688

[8] Comer, Douglas E.; Stevens, David L. (1993). Vol III: Client-Server Programming and Applications. Internetworking with TCP/IP. Department of Computer Sciences, Purdue University, West Lafayette, IN 47907: Prentice Hall. pp. 11. ISBN 0134742222.

[9] Coz, S. and Shahshahani B. *A Comparison of some Different Techniques for Vector Based Call-Routing* Proc. Eurospeech, 2001.

[10] DEVONThinkhttp://www.devon-technologies.com/products/devonthink/index.html

[11] Fan, H. and Ramamohanarao, K. 2003. *A Bayesian approach to use emerging patterns for classification.* In Proceedings of the 14th Australasian Database Conference - Volume 17 (Adelaide, Australia). K. Schewe and X. Zhou, Eds. ACM International Conference Proceeding Series, vol. 143. Australian Computer Society, Darlinghurst, Australia, 39-48.

[12] Fang, Y.C., Parthasarathy, S., Schwartz, F. 2001. *Using Clustering To Boost Text Classification.* Ohio State University.

[13] Frasconi, P., Soda, G., and Vullo, A. 2001. *Text categorization for multi-page documents: a hybrid naive Bayes HMM approach.* In Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries (Roanoke, Virginia, United States). JCDL '01. ACM, New York, NY, 11-20. DOI= http://doi.acm.org/10.1145/379437.379440

[14] Frasconi, P., Soda, G., and Vullo, A. 2002. *Hidden Markov Models for Text Categorization in Multi-Page Documents.* J. Intell. Inf. Syst. 18, 2-3 (Mar. 2002), 195-217. http://dx.doi.org/10.1023/A:1013681528748

[15] Freitag, D. and McCallum, A. 2000. *Information Extraction with HMM Structures Learned by Stochastic Optimization.* In Proc. of the 12th AAAI Conference, Austin, TX (pp. 584589),

[16] Fu, Y., Ke, W., and Mostafa, J. 2005. *Automated text classification using a multi-agent framework.* In Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (Denver, CO, USA, June 07 - 11, 2005). JCDL '05. ACM, New York, NY, 157-158. http://doi.acm.org/10.1145/1065385.1065420

[17] *General Hidden Markov Model.* http://ghmm.sourceforge.net/

[18] Helmer, Sven. 2007. *Measuring the Structural Similarity of Semistructured Documents Using Entropy.* Vienna, Austria. VLDB 07, September 23-28.

[19] Joachims, Thorsten. 1998. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features.* In European Conference on Machine Learning (ECML), pages 137 142, Berlin, Springer.

[20] Karanikolas, Nikitas. 2010. *A Parametric Methodology for Test Classification.* Journal of Information Science August vol. 36 no. 4 421-442.

[21] Koller, D. and Sahami, M. 1997. *Hierarchically classifying documents using very few words.* Technical Report. Stanford InfoLab.

[22] Lakkaraju, P., Gauch, S., and Speretta, M. 2008. *Document Similarity Based on Concept Tree Distance.* Pittsburgh, Pennsylvania, USA. HT08, June 1921.

[23] Langley, P., Iba, W., and Thompson, K. 1992. *An analysis of Bayesian classifiers.* In Proceedings of the tenth national conference on Artificial intelligence (AAAI'92). AAAI Press 223-228.

[24] Liu, N. and Yang, C. 2007. *A Link Classification Based Approach to Website Topic Hierarchy Generation.* In Proceedings of the 16th international conference on World Wide Web (WWW '07). ACM, New York, NY, USA, 1127-1128. DOI=10.1145/1242572.1242728 http://doi.acm.org/10.1145/1242572.1242728

[25] Losada, D. E. and Barreiro, A.*Embedding term similarity and inverse document frequency into a logical model of information retrieval.* Journal of the American Society for Information Science and Technology - Volume 54 Issue 4. Wiley Subscription Services, Inc., A Wiley Company http://dx.doi.org/10.1002/asi.10209

[26] Miller, D., Leek, T., and Schwartz, R. 1999. *A Hidden Markov Model Information Retrieval System.* In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '99). ACM, New York, NY, USA, 214-221. DOI=10.1145/312624.312680 http://doi.acm.org/10.1145/312624.312680

[27] Moschitti, A. and Basili, R. 2004. *Complex Linguistic Features for Text Clas-*

sification: a comprehensive study. In proceedings of the 26th European Conference on Information Retrieval Research (ECIR 2004), Sunderland, U.K.

[28] *Natural Language Toolkit.* http://www.nltk.org/

[29] NepoMuk. nepomuk.semanticdesktop.org/

[30] Pachet, Francois. 2005. *Knowledge Management and Musical Metadata.* In Encyclopedia of Knowledge Management, Schwartz, D. Ed. Idea Group.

[31] Rabiner, L.R.; , *A tutorial on hidden Markov models and selected applications in speech recognition.* Proceedings of the IEEE , vol.77, no.2, pp.257-286, Feb 1989 DOI= 10.1109/5.18626 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=18626&isnumber=698

[32] Sebastiani, F. 2002. *Machine learning in automated text categorization.* ACM Comput. Surv. 34, 1 (Mar. 2002), 1-47. http://doi.acm.org/10.1145/505282.505283

[33] Sebastiani, F. *A tutorial on automated text categorisation.* In Analia Amandi and Alejandro Zunino (eds.), Proceedings of the 1st Argentinian Symposium on Artificial Intelligence (ASAI'99), Buenos Aires, AR, 1999, pp. 7-35.

[34] Sun, A., Lim, E. 2001. *Hierarchical text classification and evaluation.* ICDM 2001, Proceedings IEEE International Conference on , vol., no., pp.521-528.

[35] Tailby, R., Dean, R., Milner, B., and Smith, D. 2006. *Email classification for automated service handling.* In Proceedings of the 2006 ACM Symposium on Applied Computing (Dijon, France, April 23 - 27, 2006). SAC '06. ACM, New York, NY, 1073-1077. http://doi.acm.org/10.1145/1141277.1141530

[36] University of Torento, Italy *Downloadable Corpora* http://disi.unitn.it/moschitti/corpora.htm

[37] Wang, W., Battocchi, A., Graziola, I., Pianesi, F., Tomasini, D., Zancanaro, M., and Nass, C. 2006. The role of psychological ownership and ownership markers in collaborative working environment. In Proceedings of the 8th international conference on Multimodal interfaces (ICMI '06). ACM, New York, NY, USA, 225-232. DOI=10.1145/1180995.1181041 http://doi.acm.org/10.1145/1180995.1181041

[38] Yi, K. 2005 *Text Classification Using a Hidden Markov Model.* Doctoral Thesis. UMI Order Number: AAINR12966., McGill University.

[39] Yi, K. and Beheshti, J. 2009. *A hidden Markov model-based text classification of medical documents.* J. Inf. Sci. 35, 1 (Feb. 2009), 67-81. DOI= http://dx.doi.org/10.1177/0165551508092257

[40] Zhang, H. 2004. *Optimality of Naive Bayes* American Association for Articial Intelligence.