NOTE TO USERS

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Quintin Armour

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

The Role of Named Entities in Text Classification

TITRE DE LA THÈSE / TITLE OF THESIS

N. Japkowicz

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

S. Matwin

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

F. Oppacher

P. Turney

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

THE ROLE OF NAMED ENTITIES IN TEXT CLASSIFICATION

QUINTIN ARMOUR

THESIS SUBMITTED TO THE
FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE M.A.SC. DEGREE IN ELECTRICAL AND COMPUTER ENGINEERING

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING
FACULTY OF ENGINEERING
UNIVERSITY OF OTTAWA

Your file Votre référence
ISBN: 0-494-11206-9
Our file Notre référence
ISBN: 0-494-11206-9

To my brother Christian whose advice I can now appreciate

Abstract

Named entities, typically associated with names of people, places and organizations, constitute a group of textual elements present in almost any type of document. The general techniques used to extract them and their variable-length property also makes them an attractive type of attribute to study in text classification. In this thesis, several datasets are characterized as being either dependent or independent of named entities with a Naive Bayes based ranking technique. Using this characterization, results are presented which find named entities to be in fact useful in classification tasks, and that accuracy can be improved by considering them as a special type of attribute. Namely, the inclusion of regular terms, named entity representation and the frequency with which a classifier is retrained all have an impact on the classification of documents where named entities are important.

Acknowledgments

I would like to acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC), Communications and Information Technology Ontario (CITO) and the University of Ottawa. I would also like the thank my supervisors Nathalie Japkowicz and Stan Matwin for their excellent guidance and feedback throughout the course of this work. Thanks as well to my defence examiners Peter Turney and Franz Oppacher whose dutiful comments and review were most appropriate. Last but not least, thanks to my family and friends for their encouragement and advice.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Given a large number of documents, comes the need to classify them. This classification ensures that despite the high number of documents, those of interest can still be found expeditiously. If these documents are provided in *electronic* format, the possibility exists to automatically classify them. When efficiently automated, the classification process can be completed with minimal human effort. Studying and finding new methods for sorting electronic documents is the problem of *text classification.* For example, some documents typically considered for classification are online newspaper articles and personal emails. These documents would be categorized into several pre-defined groups such as newspaper sections and email folders respectively.

An area of current research is *named entity recognition.* The process involves taking a document and identifying all of the *named* elements such as people, places and organizations. These entities are semantically rich, multi-word elements and occur in almost every type of document of interest to those performing text classification. The recognition of these named entities relies on the fundamental techniques of natural language processing and machine learning. For the purposes of this thesis, anything which is not

a named entity is called a *regular term.*

Any method for text classification must decide which elements or features to use from the documents in order to categorize them. The most common choice for these discriminating elements is single words. Since the simplicity of such terms is often blamed for classification errors, the main motivation for this work comes from the need for more informative features in text classification. Named entities would offer to classification such a type of feature. Single words or *bag-of-words* (BOW) representation destroys such things as word order, inflection and case, along with any semantic content they might be able to offer to text classification. It is intuitive to believe that named entities would be adversely affected by such a BOW representation. The question of whether the preservation of the lexical and semantic properties of named entities would help improve classification accuracy is subject to experimentation.

The goal of this thesis is twofold. The first goal is to identify the situations in which named entities are the key features needed for accurate text classification. The second goal is to study how, in these same situations, named entities can be used to achieve better classification accuracy. The situations which are hypothesized to improve the utility of named entities include the following scenarios:

- Named entities used in the absence of other elements
- Named entities are represented as a single entity instead of by their component words
- Named entities in classification tasks where the effect of time is reduced by retraining the classifier more often

The first point will discover whether or not named entities are negatively affected by the

presence of other non-named entity types of features. The second point will investigate which feature representation is best for named entities in those tasks where they are identified to be useful. The final point will show whether or not named entities are more significantly affected by the passage of time in comparison to other types of features. If so, the possibility of retraining the classifier to remedy the loss in accuracy over time will be studied.

The methods for conducting and evaluating text classification require machine learning algorithms such as Naive Bayes, support vector machines and decision trees. These algorithms are used to determine which documents, given a corpus, are best classified with named entities as compared with more general types of features. The different classifier types are needed to discover if there is any dependence between the task and the algorithm used.

In the remaining sections of this introduction, the following topics are discussed in general: text classification, named entity extraction and the experimentation methodology. The chapters ends with a summary of the main contributions of this thesis. In Chapter 2, all the related background material is discussed. Chapter 3 provides a method for dataset characterization and Chapter 4 presents the experiments, results and discussion. The conclusions and future work are given in Chapter 5.

1.1 Text Classification

Text classification is a process that facilitates the management of large amounts of textual documents. When documents are classified, other tasks related to the field of information retrieval are improved in terms of speed and utility. If this classification can be successfully automated, many man-hours of repetitive work can be avoided.

The mechanism for performing automatic classification is unique to each task. As such, it must be customized for every new task. This customization can be performed manually, but it is a time-consuming task. A better alternative is to use machine learning techniques to learn the mechanism needed to accurately sort the documents. One of the general approaches for learning how to classify documents relies on pre-classified or pre-labeled documents. In this approach, the document labels are used by a *supervised* learning algorithm to induce a classifier. It is this classifier that is used to categorize previously unlabeled documents. By comparison, there are also *unsupervised* learning algorithms which induce classifiers based only on unlabeled documents. These types of classifiers will arrange the documents into a specified or automatically determined number of categories. Unsupervised text classification does not apply human-supplied labels to documents. The labels can be applied after classification has been performed.

The general technique for text classification first requires a representation for the documents. In order to create this representation, the list of the elements needed by a classifier must be created. The elements of this list are called the attributes for classification. From the list, the documents are represented by the presence or absence of these attributes. The attributes for text classification are typically textual constructs from the documents themselves. These terms can be for example, words or phrases. Also of interest are the particular attributes selected for a given classification task since the quality of the classifier trained is a direct consequence of the attributes used. Current research deals with trying to remove redundant or harmful members from the attribute list by using either filter or wrapper techniques [1]. A filter is a preprocessing technique relying on things such as statistics. Wrappers use machine learning techniques to search for the features which perform the best. Often filters must be applied in order to make

the problem computationally tractable.

Working from the method for document representation, a collection of labeled documents is used to train the classifier. This collection of documents is called the *training set.* The training set is then mapped into a different format using the predetermined document representation. The training set acts as input to a machine learning algorithm. The output of this process is a text classifier. If the classifier is trained to separate the documents into two classes, it is called a binary classifier. Typically the training documents are labeled with positive (the target concept) and negative labels. The different machine learning algorithms will be discussed in detail in Section 2.1.1. A small example of a transformed training set, as it is provided to a learning algorithm, is provided in Figure 1.1. In this figure, the documents are shown on the left. To the right is the translated representation. In this representation, there are N columns and k rows corresponding to the features and individual documents respectively. If a feature appears in a document k_i, then a T is entered, otherwise F is specified. In the figure, the first $N-1$ columns correspond to the actual features and the Nth column is the class label (binary since $L1$ and $L2$ are the only labels) used in training.

In the research domain, especially when performing supervised learning, data for experiments comes from existing sources of classified documents. The data can then be segmented into the different pieces needed in order to perform the desired evaluation. The manner in which the data is divided depends on the type of evaluation being conducted. For example, in a study such as in [2], the data may be divided into two parts: training and testing. The size of each of these sets depends on the task at hand. Other studies use classifiers with parameters that need to be set, require that the dataset be divided into three: training, parameter setting and testing. In such a case, a separate piece of

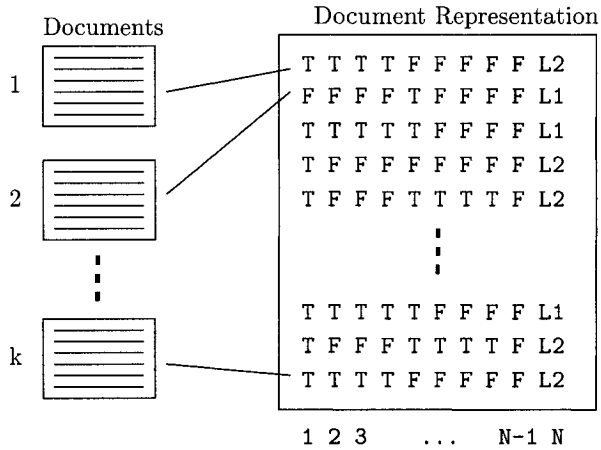


Figure 1.1: Example of document representation

data is needed to set the customized parameters of the classifier prior to testing.

No matter which type of study is being conducted, the method for making these divisions is important. In a case such as in [3] the approach is to create the sets randomly by performing k-fold cross-validation. This method divides the data into k parts, using $k-1$ parts for training and 1 part for testing. It repeats this procedure k times using a different part for testing each time. The procedure ensures that all the data will be at some point for training and at another point for testing. For certain datasets, such as news items, the data may be structured as a stream of documents. In such a case, the individual documents have a timestamp representing when the data was introduced into the set. For such a dataset, an appropriate method for segmentation would respect the ordering of the documents. The first half of the documents could be used for training and the remainder for testing.

In terms of the methods for training a classifier, there are several. Each has a different theoretical background. These include information theory, statistics, and linear algebra. The corresponding algorithms are called decision trees, naive Bayes, and support vector

machines. The different algorithms will be discussed in detail in Section 2.1.1. The associated tools of c5.0, WEKA, and libSVM will also be discussed. Other algorithms exist such as neural networks, instance and rule-based methods, but these were not selected due to the fact that they are not as popular for text classification tasks.

1.2 Named Entity Recognition

A survey of the progress of named entity recognition (NER) is provided in [4]. Research into NER has been ongoing since 1995. The main objective for researchers in this field is to extract from an unstructured document the names of individuals, locations and organizations as well as other elements such as dates and monetary amounts. NER systems endeavour to extract these types of elements independent of the problem context. This separation from context is in contrast with information extraction methods which are customized to the target domain. The main utility of NER has traditionally been in question answering systems where real-world references must be understood by the system.

An example of a successful NER system is presented in [5]. Here the system developed was an entry into the 1997 MUC (Message Understanding Conference). It used a method that relies more on the properties of the natural language than on the presence of entities in a precompiled list or gazetteer. The paper mentions the interesting property of named entities: the more well-known the entity, the less external evidence is present to help in its recognition. The authors determine that it is with these entities where gazetteers are most useful.

The types of named entities being recognized in the above example are people, locations and organizations. These named entities may or may not hold any importance

when it comes to classification. It all depends on whether the classification task is dependent on them or not. For example, if a particular organization is a good identifier for a particular class, it is important to ensure that it is correctly identified and not confused with other terms. By recognizing the term as a named entity, the risk that it can be confused with others is reduced since it relies on the properties of natural language. In other words, the named entity has lexical properties (word order and uppercase terms), as well as grammatical and contextual properties (its location in a sentence) which make it less likely to match a different term. For example, suppose the following two sentences have two different class labels:

- The Ottawa Senators participated in an outdoor practice today.
- Today in Ottawa, senators were given a 10% pay increase.

Here, although the sequence "ottawa senators" appears in both, NER is able to identify the first as an organization and the second as a sequence of a location and a noun. If these were used in a text classification task, the features generated from NER would be more useful than those produced by a BOW representation since there would be less ambiguity.

```
<TEXT>
<TITLE><Person>Galasek</Person> scores winner as
<Organization>Otters</Organization> beat
<Organization>Generals</Organization> 1-0</TITLE>
<BODY>
<Person>Tomas Galasek</Person> scored the lone goal
of the game to give the <Organization>Erie
Otters</Organization> a 1-0 win over the
<Organization>Oshawa Generals</Organization>.
</BODY>
</TEXT>
```

Figure 1.2: Example of named entity recognition

The popular NER systems in the research community are GATE and Alembic Workbench. Additionally, there is a commercial tool from Inxight that performs NER. The system used in this research is GATE 2.2 available from the University of Sheffield [6]. It was selected based on its successful use in other research and availability. It allows NER process to appear as a black box. The input is the document provided in SGML and the output is the tagged document. An example of such output is provided in Figure 1.2. The software allows for a variety of named entities to be recognized. These named entities include: Date, Job Title, Organization, Person, Location, to name the most important. It provides a graphical interface to the recognition process as well as an API for the batch processing of larger tasks. For this research, the process was automated for large corpora using the programmer interface provided.

1.3 General Methodology

In this section the general procedure used for the experiments will be described. When the actual experiments are described in Chapter 3 and 4, further details will be provided.

Since one claim of this thesis is that named entities can be useful in text classification, experiments must be conducted to test this claim. As such, the general procedure involves the following steps and is depicted in Figure 1.3:

1. Named entities must be identified and extracted from the training documents
2. All regular terms must be identified from the same set of documents
3. A classifier must be trained for each type of document
4. The performance of each classifier must be determined for each attribute type

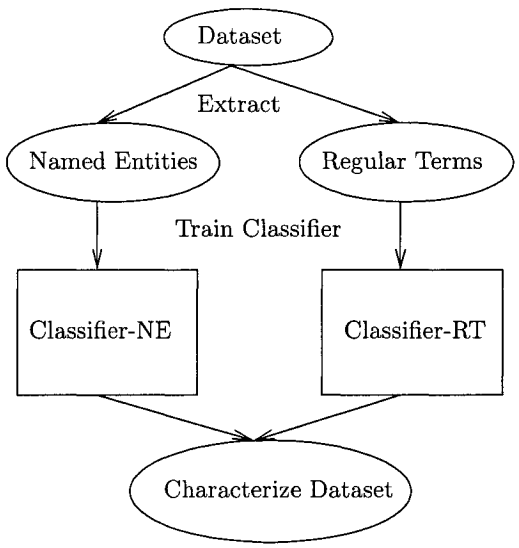


Figure 1.3: General methodology

After accomplishing the final step, dependence of a dataset on a particular attribute type can be determined. If the dataset is deemed to be named entity dependent, then further experiments can be conducted to test if there are any ramifications associated with this dependency. For example, named entity representation would be more important in terms of how it impacts classification accuracy if the dataset is named entity dependent.

The methodology for the experiments in Chapter 4 follows standard text classification techniques. In other words, different feature lists will be constructed given the experiment at hand and a classifier trained and evaluated. Further details are provided in the setup description for each experiment.

1.4 Main Contributions

The first contribution of this thesis is the detailed analysis which is presented of the utility of named entities as attributes in a variety of text classification tasks. Named entities are

believed to be a semantically rich type of attribute suitable for document representation. By the evaluation of named entities in the context of these tasks, it is demonstrated that they are useful in text classification. In this evaluation, three important considerations when employing named entities in classification tasks are investigated. These considerations are: the lower frequency of named entities relative to other types of attributes; the choice of named entity representation; and the greater sensitivity named entities have with respect to time. These considerations are important since each helps to increase the impact of named entities on classification. The first consideration shows that due to the frequency difference between attribute types, named entities are overshadowed in tasks where they are the best type for document representation and thus should be used in isolation. The second consideration demonstrates the potential negative effect of using a word-based named entity representation as opposed to one which retains the full entity description. The third studies how named entities, if used in text classification, need special attention, with more frequent classifier retraining, due to their greater dependence on time than regular terms.

The second contribution of this thesis is the method developed to characterize a dataset according to how well it is represented by its attribute list given a classification context. This method is able to infer when there is a relationship between attributes and dataset. In a case with two attribute lists, the dataset can be categorized as being dependent on one or the other, both or neither type of attribute. This characterization can be used to evaluate an attribute list, simplify and improve classification by reducing the size of a combined attribute list and help to target experiments to datasets which are dependent on a particular type of attribute. This method is applicable for any attribute list and dataset and is better than accuracy since it rewards classified examples in terms

of how easy they were to label.

The third contribution is the collection of two news article datasets from Yahoo! News. These were collected over the span of seven months in 2004 and consist of about 4000 documents each. The first dataset presents itself as a two-class classification problem where the articles deal with events in professional and junior hockey. In this thesis, these articles were found to be well represented by the named entities they contain and therefore the dataset can be used in the future development of techniques that incorporate named entities in text classification. The second dataset, also with two classes, offers articles related to television and movie news items. This dataset is useful in that it offers articles which are semantically close, since they deal with a similar topic and therefore it can be used to evaluate new text classification methods.

Chapter 2

Background

This chapter will present the background material and justification for the experiments conducted in Chapters 3 and 4. This material consists of the machine learning algorithms used to train text classifiers and the tool used to perform named entity recognition. The chapter will also present a review of the current related literature in order to demonstrate the relevance of this work by placing it in the context of other research.

2.1 Resources

In this section, an introduction to the theory behind the machine learning algorithms used in this thesis is presented, followed by a description of the software used to conduct the named entity recognition required.

2.1.1 Machine Learning Algorithms

The core component of text classification methods is the algorithm that trains the classifier. The algorithm analyzes the training data and outputs a method for determining the

class of an unlabeled document. The three algorithms and the names of their associated implementations are provided in the following list:

1. Naive Bayes (implemented by author)
2. Support Vector Machine (libSVM)
3. Decision Tree (C5.0)

This section will describe each of these algorithms in turn. The description will include a discussion about the mathematical background and how it is used in text classification. These algorithms were selected in particular because of their current popularity in text classification tasks. Part of this popularity can be attributed to their availability and the ability of their associated implementations to deal with high-dimensional data.

Naive Bayes

The Naive Bayes algorithm is built upon the principles of Bayesian probability. The algorithm will compute the conditional probability that a document belongs to a class given the evidence that an attribute has a particular value. It is by using the conditional probabilities for all the attributes occurring or not occurring in a document that the algorithm is able to classify documents.

In general terms, this conditional probability can be expressed as:

$$P(c|A) = \frac{P(c \cap A)}{P(A)} \tag{2.1}$$

Since $P(c \cap A) = P(A \cap c)$ and $P(A \cap c) = P(A|c) \cdot P(c)$, $P(c|A)$ can be computed from

the probabilities $P(A|c)$ which are known:

$$P(c|A) = \frac{P(A|c) \cdot P(c)}{P(A)} \tag{2.2}$$

In text classification terms $P(c|A)$ is the probability that a class label c is correct for the attribute evidence A. From the training data, the probability that an attribute A occurs given the class label c is used to compute $P(A|c)$.

For the optimal Bayes theorem, which is the foundation of the Naive Bayes classifier, the probability that a document belongs to class c is dependent on the joint probability $P(c \cap (\bigcap_{i=1}^{n} A_i))$. For the situation where $n = 2$, $P(c \cap A_1 \cap A_2) = P(c) \cdot P(A_1|c) \cdot P(A_2|c \cap A_1)$. The optimal Bayes theorem can therefore be expressed (for $n > 2$) as:

$$P(c|\vec{A}) = \frac{P(c \cap A_1 \cap A_2 \cap \ldots \cap A_n) \cdot P(c)}{P(A_1 \cap A_2 \cap \ldots \cap A_n)} \tag{2.3}$$

For even larger values of n, this expression is even more difficult to compute since it is hard to get accurate estimates given data which is of limited size.

The solution to this problem was to consider all attributes to be independent. Then Equation 2.3 becomes:

$$P(c|\vec{A}) = \frac{P(A_1 = v_1|c) \cdot P(A_2 = v_2|c) \cdot \ldots \cdot P(A_n = v_n|c) \cdot P(c)}{P(\vec{A})} \tag{2.4}$$

The values on the right hand side of Equation 2.4 are conditional probabilities computed from the frequency of occurrence of the attributes. For a given attribute and training set this probability is:

$$P(A_i = v|c_j) = \frac{C(A_i = v, c_j)}{N(A_i)} \tag{2.5}$$

where $C(A_i = v, c_j)$ is the number of times the attribute A_i occurs with the value v in documents labeled with the class c_j in the training set. The denominator, $N(A_i)$, is the total number of occurrences of A_i for all values and for all training set documents.

In order to make a classification decision on an unlabeled document, Equation 2.4 is evaluated for each possible class c. This process can be represented as follows:

$$\hat{c} = \underset{all\ c}{\text{argmax}} \left[\left(\prod_{i=1\ldots n} P(a_i = v_i|c) \right) \cdot P(c) \right] \tag{2.6}$$

The class that returns the highest value for Equation 2.6, $\hat{c}$ is suggested for the unlabeled document. Note that the denominator from Equation 2.4 is left off since it is common for all classes.

For an attribute that does not occur for a class of documents, $C(A_i = v, c_j)$ from Equation 2.5 is zero. For example, consider a binary classification problem where $P(A_1 = v|c_1) = 1$. This fact implies that $P(A_1 = v|c_2) = 0$. If the attribute A_1 appears in an unclassified document, $P(c_1|\vec{A})$ will be > 0 but $P(c_2|\vec{A})$ will be zero because $P(A_1 = v|c_2)$ was zero and $P(c_2|\vec{A})$ is calculated by multiplying the conditional probabilities for all A_i for $i = 1 \ldots n$ (as in Equation 2.4). To avoid this problem, zero-valued conditional probabilities are assigned a small non-zero value. The method applied here is Laplace smoothing where Equation 2.5 can be rewritten for each class as:

$$P(A_i = v|c_j) = \frac{C(A_i = v, c_j) + 1}{N(A_i) + k} \tag{2.7}$$

where k is the number of classes.

For the experiments presented in later chapters, access to the internals of the algorithm is necessary. Existing implementations, such as WEKA [7], would have had to be

modified. The WEKA version also had an issue with how it dealt with smoothing. It assigns conditional probabilities which have the value of zero with a very small constant value. A better way to do handle smoothing is to assign zero probabilities with a value that factors in the total number of occurrences for that particular attribute, as is the case with the method presented above. For these two reasons, and since the algorithm is fairly simple to implement, a customized version was written and used for this thesis.

Support Vector Machines

Support Vector Machines (SVMs) are a class of algorithms built on the theory of linear classifiers. Linear classifiers will take a set of numeric attribute vectors and map them into a set of two or more values. The mapping to classes is performed by finding a dividing hyperplane in N-dimensional space. The specific hyperplane is described by a linear combination of weighted attribute values. SVMs apply the same principle except they operate in a space of higher dimensionality in order to allow for non-linear classification boundaries. For simplicity, only binary classification will be considered in this section since multi-class SVMs are an extension to this theory.

A linear classifier is derived from the principle of linear regression. Linear regression is the process of finding a function to correspond as closely as possible to a set of points in space. This function f can be represented as:

$$f = b + w_0 a_0 + w_1 a_1 + \ldots + w_n a_n \tag{2.8}$$

where a_i are the numeric attributes for each example and w_i are the weights learned by the algorithm to allow for the greatest separation of the two classes. The parameter b is a bias factor to shift the hyperplane over by the specified amount. The weights of the

attributes are learned so as to ensure the best possible fit to the set of points provided. The best hyperplane is therefore the one that minimizes the average error between the points and the itself. To use linear regression as a binary classifier, the function output is restricted to two values: ± 1. The classifier would then take a new example and apply the function. If the result is negative, a corresponding label of -1 is assigned. Conversely if the result is positive, the example is labeled with $+1$.

For SVMs, the function used to classify the examples is different from linear regression in that it is expressed in terms of *support vectors*. These support vectors are special labeled points in space that are closest to the dividing hyperplane. The support vectors are depicted graphically in Figure 2.1. The use of support vectors allows the classifier function to process fewer elements when classifying examples, rather than calling upon all training examples. Another interesting aspect to the function is that it uses the dot product to compute the distance between the unlabeled point and the support vectors to evaluate which label is most appropriate. The dot product is equivalent to evaluating the distance between two points, except it is applicable in N-dimensional space. The classifier can be represented by the following equation:

$$f = sign\left(\sum_{i=0}^{n} w_i(\mathbf{x_i} \cdot \mathbf{y}) + b\right) \tag{2.9}$$

where w_i are the weights set in training, $\mathbf{x_i}$ is the attribute values of a support vector and $\mathbf{y}$ is the unlabeled instance.

The problem with linear classifiers is that they can only learn how to label linearly-separable classification problems. The majority of classification problems are not linearly-separable. The general idea of SVMs is to apply the simplicity of a linear classifier, but to non-linear classification problems. The trick is to take the problem as specified and

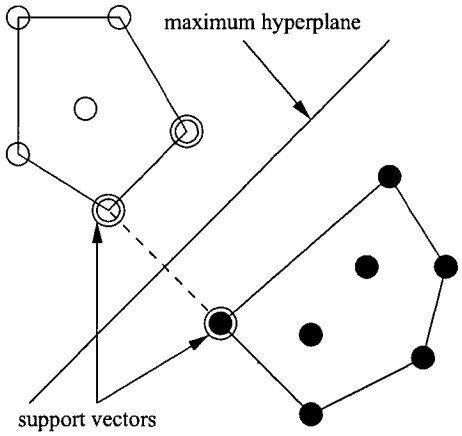


Figure 2.1: Support Vector Machine

map it into a *feature space* of higher dimensionality. The belief is that in feature space, the problem will be linearly-separable and the regular method can be applied. Formally, the mapping from vector space to feature space is defined as:

$$\Phi : X \rightarrow F \tag{2.10}$$

$$x \rightarrow \Phi(x) \tag{2.11}$$

The disadvantage to performing this mapping to feature space is that the number of features will balloon into an unmanageable amount. The dot-product from Equation 2.9 required to assign labels to new examples would be too computationally expensive to perform. Fortunately, kernel functions exist which have the property of returning the same result for a dot-product independent of the dimensionality of the vectors:

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}') = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')) \tag{2.12}$$

If such a kernel function is used there is no additional computational cost for considering

the space of higher dimensionality. The problem is never actually considered in feature space. The kernel function allows the examples to be classified by a linear hyperplane in this feature space which is non-linear in vector space.

Some examples of kernel functions that allow for linear class separation in feature space while actually creating non-linear boundaries in the original vector space are:

- Polynomial

$$k(x, y) = ((x \cdot y) + \theta)^d \tag{2.13}$$

- Radial Basis Function (RBF)

$$k(x, y) = e^{-\gamma \|x-y\|^2} \tag{2.14}$$

- Sigmoid

$$k(x, y) = \tanh(\kappa(x \cdot y) + \theta) \tag{2.15}$$

where θ, d, γ and κ are training parameters. Each of these kernels will create a corresponding boundary in the original vector space. Usually since for a particular problem, there may not be any intuition for which kernel to use, several or one that works well in general would be applied.

The implementation of SVMs used in the experiments is called libSVM. This software is a library for performing support vector machine classification and regression. It provides several types for each, but the one used in this thesis is C-Support Vector Classification (C-SVC). The input to the software is the usual vector representation of examples but in sparse format where zero-valued attributes are removed. The software was run with default parameters, $k(x, y) = RBF$ and with $\gamma = \frac{1}{k}$, where k is equal to

the number of attributes.

Decision Trees

The final machine learning algorithm considered is that of decision trees. The name refers to the structure used to visualize the classifier after it has been trained. The method applied to construct the tree is derived from the principles of information theory. In high-level terms, the tree is a description of the data requiring the least amount of information. An example is shown in Figure 2.2.

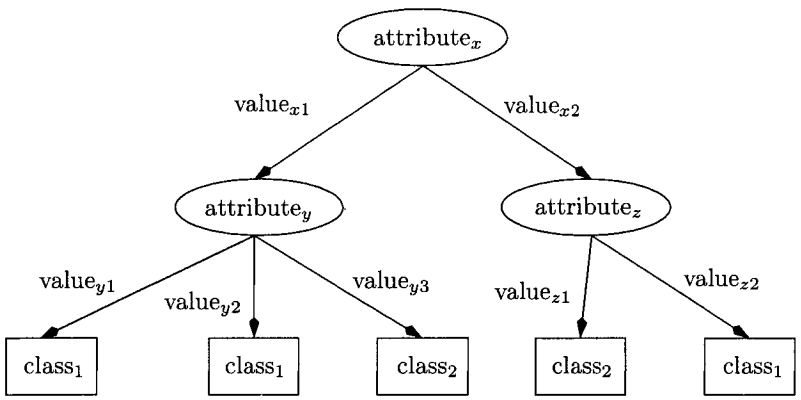


Figure 2.2: Decision Tree

A decision tree is built from a hierarchy of decision nodes. Each decision point in the tree is represented as an internal node. The attribute selected for a node is selected based on how much information is required to describe its addition to the tree. The best attribute is the one where the tree has the most to gain by its use. The information required to describe a branch of a split is given by:

$$\text{info}([c_1, c_2, \ldots, c_n]) = \text{entropy}(\frac{c_1}{T}, \frac{c_2}{T}, \ldots, \frac{c_n}{T}) \tag{2.16}$$

where $T = \sum_{i=1}^{n} c_i$ and entropy is a measure of disorder. It is large when the purity of a potential subtree is low and small when the purity is high. Entropy is the following expression:

$$\text{entropy}(p_1, p_2, \ldots, p_n) = \sum_{i=1}^{n} -p_i log p_i \tag{2.17}$$

where $n = 2$ for binary class problem. The information gain is computed for each attribute:

$$\text{gain} = \text{info(Parent)} - \text{info(Child)} \tag{2.18}$$

For the root of the tree, *gain* is defined to be equal to $info(Child)$ since it has to parent.

Decision trees also have a measure to give preference to attributes that create fewer children. For this, the information required to represent the split is computed in a similar way to Equation 2.16 except the c_i refer to the example counts in the each of the subtrees of the candidate node:

$$\text{split info} = \text{info}([e_1, e_2, \ldots, e_n]) \tag{2.19}$$

Having this definition allows for the actual measure used at each decision point in the tree to be defined. It is in fact the attribute which produces the highest gain ratio measure that is selected. Gain ratio is defined as:

$$\text{gain ratio} = \frac{\text{gain}}{\text{split info}} \tag{2.20}$$

The splitting of the tree by gain ratio proceeds until the subtrees compute a gain of zero or the tree cannot be divided further. Additionally, the tree could be *pruned*, a process where leaves are combined to shorten the tree.

The decision tree implementation used is C5.0. It is a well-known piece of software

produced by RuleQuest Research. The input to this software is at least two files. The first is a *names* file describing the attributes and class labels of the data. The second is the *data* file, which is a list of all the attribute values. Additionally, a *test* file is required if a specific test set is built. C5.0 also allows the built tree classifier to be linked to external programs. The public source code provided on RuleQuest's website requires another file, *cases*, to be specified as well. This *cases* file is identical to the *test* file, but the class labels are optional.

2.1.2 Named Entity Tagging

A necessary pre-processing step in this thesis was to tag the named entities in each document of the datasets. This step is required in order to be able to perform the experiments and analysis in Chapters 3 and 4. The tool used to perform the named entity tagging is called GATE: General Architecture for Text Engineering. GATE 2.2 is available through the University of Sheffield [6]. This piece of software provides the framework to accomplish a lot of tasks related to natural language processing. These tasks include such things as parsing and part-of-speech tagging.

GATE also provides a larger tool called ANNIE: A Nearly New Information Extraction system. This system deals with many of the sub-problems of natural language processing in order to perform named entity recognition. For example, the main components of ANNIE are the text tokenizer, lemmatizer, sentence splitter, part-of-speech tagger and grammar parser. A complete description of the procedure for tagging named entities in text is given in Figure 2.3.

The method for tagging named entities is therefore described as a pipeline where the output of one block is the input to the next block. For example, and referring to

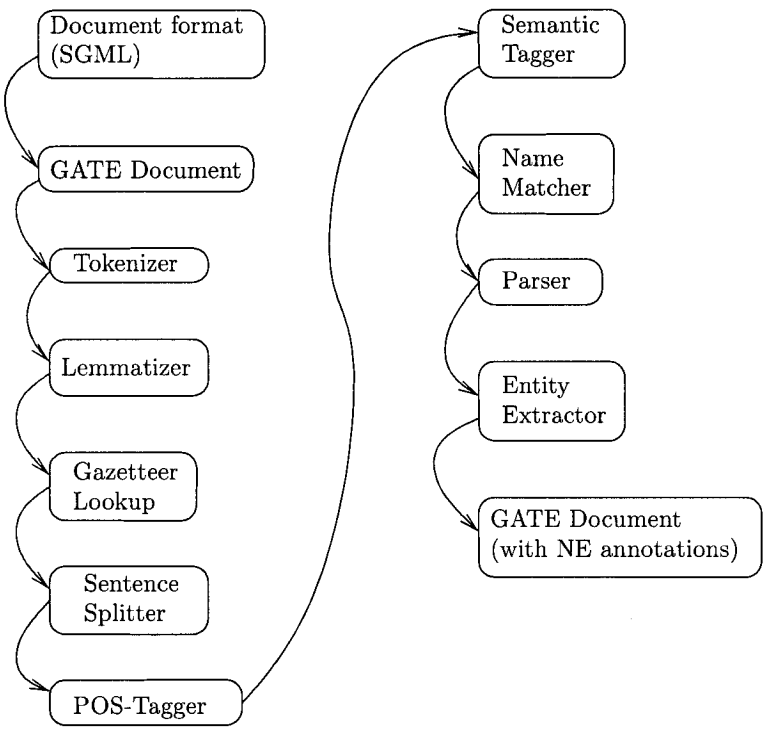


Figure 2.3: Pipeline used to perform named entity recognition by GATE

Figure 2.3, the source document is first fed to a tokenizer which will split everything into words and characters. The words will then be converted into their roots or lemmas. A gazetteer, or a list of well-known entities, is used to identify its member elements in the text. Next, sentences are split and words are tagged with part-of-speech tags. Some semantic knowledge is tagged to aid in identifying the types of the named entities present. Next, names are matched and grammar rules are used to identify those for which the gazetteer cannot tag. Finally, the entities are extracted and tagged according to their most likely types. The following named entities are some of the most commonly tagged: Person, Location, Organization, Date, Address, Money, Percent or Job Title.

The entities of interest in this thesis are those of Person, Location and Organization. These particular entity types were selected based on the view that they are the most interesting [5]. Also, for the datasets being considered, these entities hold the most semantic value. For instance, in news articles, the who (Person and Organization) and where (Location) are very important.

2.2 Literature Review

This section presents a review of the current literature related to the topics of text classification and document representation methods. The review will look at the different kinds of approaches used by different researchers to try and improve text classification. This review will provide the research context for this thesis.

The majority of research in text classification involves some kind of evaluation with the Reuters-21578 dataset. Bekkerman et al. in [8] argue that the Reuters dataset in particular, since it was manually labeled, favours the development of keyword-based text classifiers. In other words, the people labeling the documents relied on the occurrence

of keywords to help them in their task. Since the Reuters dataset would seem to only work well with keyword features, then any more sophisiticated approach probably will not seem too successful when tested on this *standard* dataset.

This conjecture regarding the Reuters dataset is supported by work by Lewis [9] which shows that a phrasal and word-cluster approach to text classification performs poorly on the Reuters dataset. The main reason for this decay in accuracy, according to Lewis, is the sparseness of the document-feature matrix. The reasoning behind this argument is that if content words, like nouns and verbs, are not very frequent by nature, then sequences of content words are even less likely to occur since there are greater number of possible combinations.

Such failures have led to the development of information extraction (IE) based techniques. These IE-based approaches try to extract pertinent elements from a dataset instead of a brute-force like approach such a n-grams. Riloff has conducted such experiments to show that role-based text classifiers can achieve high accuracy [10]. In this author's work, features or extraction patterns such as "<victim> was *murdered*" and "<perpetrator> attempted to *kill*" are used to detect news articles related to terrorism. The italicized word is referred to as a *trigger word* and is used to "fire" the specific extraction pattern. The rest of the extraction pattern is used to form the *case role*, hence making the classifier role-based. This approach is successful, but suffers the same difficulties as with all IE-based solutions: the need to write a custom extractor for each separate task. Riloff has developed a tool to reduce the cost of doing this task, but it still must be bootstrapped in order to perform well.

In another paper by Croft et al. [11] the authors mention that "words may be associated and co-occur, but not be part of the same phrasal concept. NE (named entity)

extraction preserves the concept." In other words, if something can be distinguished as a named entity then it is known that the words belong to the same phrasal concept and the semantics are maintained. This research provides an indication that named entities are useful elements to extract from text because they are essentially like multi-word nouns.

Cooley in [2] endeavours to classify news articles with SVMs. The author introduces the idea of using named entities as a way to reduce the feature space in this domain. He argues that since named entities are present in almost all text documents and certainly in all news articles, that this may be a successful tool in adding some semantic meaning to the classification process. In his results, he finds out that the classifier trained only on named entities, although it reduced the feature space, does not produce sufficiently high accuracy to be of any use. The limitation of the work is that the results are based only on one dataset and do not investigate the reasons for which named entities produced poor results. There are several possible reasons why named entities did not perform well. These will be studied in this thesis.

In a later work [12], Cooley and Clifton develop a tool to perform *Topic Detection and Tracking* (TDT). This tool makes good use of named entities in using them as the cornerstone of their algorithm. In the algorithm, named entities are grouped to form frequent itemsets using a method from association rule mining. These itemsets are used in a technique to cluster documents sharing a common topic. This method demonstrates how named entities can be linked by a common topic. Based on the success of this technique, named entities can be considered to be linked to topics in news articles. This fact suggests that named entities would be useful in a text classification task.

Inxight Corp. produces commercial software for text discovery. This application, as

described in a white paper [13] helps to traverse and sort text documents. A core component of this tool is a module that performs named entity recognition. At the user's request, named entities can be added to the list of features used in the categorization of documents. The missing component in this tool is one which can automatically determine which named entities are useful for classification, instead of relying on the user to manually add them. The work presented here is a step in developing such a method. This thesis is presented as a study of how named entities behave when used in a variety of text classification tasks.

As mentioned in [4], named entities offer a general method for extracting predetermined multi-word features from text. This generality is not present in the more difficult problem of information extraction which requires a customized extraction engine for each classification task. From the research presented in other papers, their use in text classification seems to be a research-worthy endeavour.

As evidence of the current topicality of named entities in text classification, Ron Bekkerman in a recent (2004) technical report [14], presents an evaluation of text classification methods on the Enron corpus (described in Section 3.1.5). He mentions in his future work that "[n]amed entities may be highly relevant features. It would be desirable to incorporate a named entity extractor into the foldering system." The incorporation of named entities into text classification tasks is exactly the purpose of this thesis.

Chapter 3

Dataset Characterization

The objective of this thesis is to determine the role of named entities in text classification and how this role if any can be exploited to achieve better classification and a better understanding of this type of feature. To this end, a method to measure the overall influence of an attribute list (i.e. named entities) on a dataset is first required. Having such a measure will enable datasets to be characterized according to their attribute dependence. Once a dataset is characterized, the results obtained from experiments have more meaning since they are presented within a context. For example, showing that a particular named entity representation is superior to another is more significant if the dataset as a whole is known to be dependent on named entities. If named entities were not critical for classification then it would not matter in the final system how they were represented. The goal of this chapter is therefore to obtain characterizations for each of the datasets being studied. The following categories are introduced as the different characterizations possible:

1. Named entity dependent

2. Regular term dependent

3. Dependent on both named entities and regular terms

4. Independent of named entities and regular terms

The usual way to test the performance of an attribute list on a dataset is to classify a collection of test documents and obtain a value such as accuracy, precision/recall or F-measure. The problem with a type of measure like accuracy is that it is not representative of the facility with which classification is performed. The authors in [15] raise this issue and suggest an alternate measure, namely the *area under the curve* (AUC) score of an ROC (Receiver Operating Characteristics) plot. The method will reward an attribute list if it produces a ranking where all the correct classifications have higher rank than the incorrect classifications. The method used in this thesis, weighted ACC-COV, is related to the ROC-AUC measure, only different weights are applied to different regions of the curve. Since it is a related measure, it shares in the improvements of ROC-AUC over accuracy. As such, it is also not only concerned with how many examples were correctly or incorrectly classified, but also with how easily they were classified. An attribute list can then be evaluated using this method because it measures the quality of the ranking and whether the top ranked examples produce high accuracy. This method improves upon ROC-AUC, in terms of being able to characterize datasets, by providing greater discrimination between easy and hard to classify examples. The details of these measures will be discussed further in Sections 3.3.1-3.3.3.

These measures to evaluate attribute dependence (with the exception of accuracy) rely on the fact that the ranking is performed by an algorithm which is good at ranking. In other words, if the algorithm used is poor at ranking examples, then even when there is an attribute dependence the score produced would be low. The method presented

Section 3.2 will look at how rankings can be produced by each of the three machine learning algorithms considered. It will then describe and evaluate the ability of each algorithm to rank classifications. The method will display the rankings in a manner similar to [14] using accuracy-coverage curves.

This chapter will first present a description of the datasets used in these characterizations and in the experiments in Chapter 4. Next, the chapter will study how the examples from these datasets can be ranked using the three different machine learning algorithms. The datasets will then be characterized using the measures of accuracy, ROC-AUC and weighted ACC-COV and the best used to determine the final characterizations.

3.1 Datasets

This section describes the general nature of the datasets used in this thesis as well as the method used to prepare the data for experimentation. There are four sources for the seven datasets used. These sources can be labeled as follows:

- Yahoo! News, a collection of online news articles from January-July 2004
- Reuters-21578, a collection of news articles from January-October 1987
- 20 Newsgroups, a collection of newsgroups postings from April 1993
- Enron Email Corpus, a collection of emails exchanged by Enron employees during the Enron scandal of 2002

Since the datasets presented in this section come from discrepant sources, they must first be converted into a standard format. The following section describes this format. In the subsequent subsections, the datasets themselves are presented.

3.1.1 Data Formatting

The standard format developed allows for the data to act as input to the subsequent stages of named entity recognition and machine learning. The format is expressed in SGML and uses specific tags as shown in Figure 3.1. It separates the document title from the body in case either one is undesired. Allowance for document timestamps is also made. The document id tag is there in order to help track the documents in the preprocessing stages. The topic tag specifies the category for the document.

```
<TEXT>
<TOPIC> . . . </TOPIC>
<ID> . . . </ID>
<DATE> . . . </DATE>
<TITLE> . . . </TITLE>
<BODY>
. . .
</BODY>
</TEXT>
```

Figure 3.1: Standard document format for input files

For news articles, the mapping is straight forward as they have titles, topics, and timestamps. For the other document types such as newsgroup postings and emails, the subject component maps into the title element. In terms of how the body is constructed for email documents, header information is discarded as it could be the source of some unwanted regularity in the results.

3.1.2 Yahoo! News

Yahoo! is a popular web-based news and entertainment service. The benefit to using Yahoo! news articles is that they are already sorted into categories. The categories are arranged hierarchically, but the different levels can be flattened if desired. The data for these categories was collected semi-automatically, and therefore articles are not necessarily available for all dates. The code for the downloaded web pages is easily parsed to extract the article text. In general, the text is well-written in that it has acceptable grammar and few spelling mistakes.

Hockey

This dataset consists of 3943 documents from January 1st, 2004 until July 18th, 2004. It is well-suited for use in a binary classification task. The two categories are formed by dividing the documents into those about professional hockey and those related to junior hockey. The articles deal largely with summaries of games played and also of any stories about hockey players or hockey teams at the time. The dataset was selected because it is surely reliant on named entities. Regular terms would be of limited use since the categories deal with the same general topic of hockey. For this dataset, 2914 documents belong to the professional hockey class and 1029 are related to junior hockey.

Movies

This dataset is comprised of 4211 documents from January 1st, 2004 to July 18th, 2004. The data lends itself to being classified into two categories: those articles related to television and those related to movies. The classification problem associated with this dataset is an interesting one because the concepts are similar in nature. The nature of the

Category	No. Articles
earn	3987
acq	2448
money-fx	801
grain	628
crude	634
trade	551
interest	513
wheat	306
ship	305
corn	254

Table 3.1: Document frequencies for top 10 categories of Reuters-21578

articles revolve around reviews of the shows and movies. Also accounts of events, such as the academy awards or any controversy surrounding a particular film or TV show, are present. The fair amount of data available for each category was also a factor in selecting these articles for classification. For the dataset, 2171 documents are in the category of movies and 2040 are related to television programs.

3.1.3 Reuters

This dataset has been used in a lot of text classification research [16] [17]. Its categories and characteristics are well-known. The dataset consists of 21578 articles from 1987. The standard way to split the data, named the ModApte split, is to use the articles from January to May for training and then use the data from October to November for testing. For the experiments conducted in this thesis, only the top ten categories were used. The data in Table 3.1 gives a list of the categories and the number of articles for each.

From these categories, binary classification problems can be created. One class of articles can be classified against all other articles in the top ten categories as in [16]. For example, the articles for the *earn* category would represent the positive class and all

other document classes would belong to the negative class.

For the experiments in this thesis, only the two largest document classes were used as positive classes. Namely, the *earn* and the *acq* categories. The rest of the documents in the top ten were used as the negative class in each of these cases. By using the largest categories, other problems such as having too little training data and class imbalance can be mitigated. The datasets were selected because they are known to be keyword-based [8] and therefore will serve as a useful benchmark for the findings.

The Reuters-21578 dataset is provided by David Lewis and is available online at http://www.daviddlewis.com/resources/testcollections/reuters21578/.

3.1.4 20 Newsgroups

Newsgroups are another source of pre-classified documents for text classification. Like news articles, they are structured hierarchically, and the categories can be collapsed to create larger datasets. The difference between these and news articles, however, is that these texts tend to be more informal. For example, the texts are not written in a professional manner. There is also a greater chance that an article is posted to the wrong category.

The 20 Newsgroups dataset consists of about 20000 newsgroup articles from April 1993. In other words, there are roughly 1000 postings per category. The categories and their respective topics are presented in Table 3.2. The dataset is available at http://people.csail.mit.edu/u/j/jrennie/public_html/20Newsgroups/.

For the experiments, a subset of this dataset is used. This subset is described in Table 3.3 and consists of about 5000 newsgroup postings. Several subcategories were combined to translate this problem into a binary classification task. The topics were selected in

Category	Topic Covered
alt.atheism	religion
comp.graphics	computer graphics
comp.os.ms-windows.misc	microsoft windows
comp.sys.ibm.pc.hardware	computer hardware
comp.sys.mac.hardware	computer hardware
comp.windows.x	X Windows
misc.forsale	item sales
rec.autos	cars
rec.motorcycles	motorcycles
rec.sport.baseball	baseball
rec.sport.hockey	hockey
sci.crypt	cryptography
sci.electronics	electronics
sci.med	medicine
sci.space	space
soc.religion.christian	religion
talk.politics.guns	politics
talk.politics.mideast	politics
talk.politics.misc	politics
talk.religion.misc	religion

Table 3.2: Category to topic mapping for 20 Newsgroups dataset

Category	Label
rec.sport.hockey	positive
alt.atheism	negative
soc.religion.christian	negative
talk.politics.misc	negative
rec.sport.baseball	negative

Table 3.3: Subset of the 20 Newsgroups dataset used in experiments

terms of their proximity in subject matter. Only five categories were selected as the base set so that if one category is compared to all others, then there is no problem of data imbalance [18]. Table 3.3 describes the labels assigned to the categories used in a classification experiment.

3.1.5 Enron Email Corpus

In late 2001, the Enron company filed for bankruptcy protection. The company admitted to doctoring their accounts to make itself appear to be more profitable than it really was. The Enron dataset comes from a set of emails made public in the wake of the resulting scandal. According to its current distributor, William Cohen, the dataset was cleaned up (removal of attachments, verification of email address consistency, etc.) by Melinda Gervasio at SRI International. The version of the dataset used in this thesis is a different one, available from Ron Bekkerman. In this version, non-topical folders have been removed (such as inbox and sent-mail), in order to make it more directly used in text classification tasks. This "flattened" version contains the emails from seven Enron employees. The dataset is available at http://www.cs.umass.edu/~ronb/datasets/enron_flat.tar.gz.

A subset of the complete flattened set is used in this work, specifically from two of the seven employees. The employees are referred to by their last name and the first name initial. The two employees referred to in this thesis are: *lokay-m* and *farmer-d*. Table

Dataset Id	Positive Class	Count	Negative Class	Count
farmer-d	farmer-d\logistics	1192	farmer-d\tufco	609
			farmer-d\wellhead	339
			farmer-d\personal	321
lokay-m	lokay-m\tw_commercial_group	1159	lokay-m\corporate	407
			lokay-m\articles	243
			lokay-m\personal	190
			lokay-m\enron_t_s	179

Table 3.4: Summary of data from the enron-flat dataset

3.4 gives a summary of the two datasets in terms of how many emails are available and how they are categorized. This table also shows how a two-class classification problem was crafted from the multiple classes available. The largest section of documents is taken for the positive class and the next n sub-categories are used for the negative class. The value n was chosen such that the total number of documents for each class is roughly equal.

3.2 Document Ranking

This section presents the method for obtaining the *certainty* measures from the three machine learning algorithms from Section 2.1.1. The certainty measure reported is a value representing how certain a classifier is that its classification is correct given the different class labels available. The certainty measure is therefore indicative of the distance between the predicted class and the next most likely class. It is derived from the numeric outputs for each of the different learning algorithms. The intuition behind using this measure is that if an example, given the attributes used, should obviously be assigned to a particular class, then the certainty value should be high. Such certainty values can be used to rank the examples of a dataset. With high certainty comes high

rank and with high rank comes the hope for a correct classification. The purpose of the experiments in this section is therefore to determine which algorithms are best suited for ranking classifications, given a dataset and the attributes specified. Such an appropriate ranking will enable suitable characterizations of the datasets to be made. The degree to which such characterizations are possible will now be evaluated for the different machine learning algorithms under consideration, namely, Naive Bayes, SVM and Decision Trees.

For each algorithm, a series of figures generated using the *movies* and *lokay-m* datasets is provided to demonstrate graphically how well the documents are ranked. These datasets were selected in particular because they are among those for which it is difficult to ascertain attribute dependence prior to classification, making them interesting from the point of view of ranking. The preferred algorithm will be the one which is best able to rank the examples given their underlying dependencies. In Figure 3.2, an annotated sample graph is provided to demonstrate how the figures can be interpreted. In this figure, two curves are presented. The solid line represents a sequence of accuracy calculations. The initial accuracy value, on the left-hand most side of the figure, is calculated using the entire test set. For each subsequent value on this curve (from left to right), examples are excluded from the accuracy calculation. Since the test examples are ordered by certainty, the accuracy is being calculated on smaller and smaller subsets of the test set. The minimum certainty value for these subsets increases for each subset. Accuracy is therefore expected to increase from left to right since higher certainty should mean higher accuracy. For the dashed-line curve (1-coverage), a similar procedure is followed, except the value reported is the percentage of total examples which have been excluded from the accuracy calculation. This curve is included in the figure in order to show how successful the classifier is at assigning errors with a low certainty value. The

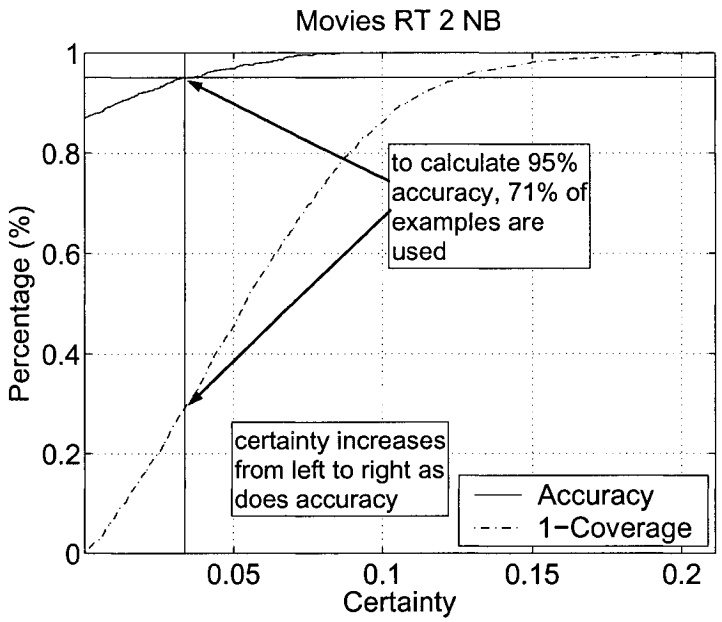


Figure 3.2: Annotated characterization curve

logic is that if the classifier is going to make mistakes, then at least they should have low certainty and rank.

In other words, for a successful ranking of examples (given an attribute list), the following equations have to be true:

$$P(e_1) > P(e_2) > \ldots > P(e_n) \tag{3.1a}$$

$$cert(e_1) > cert(e_2) > \ldots > cert(e_n) \tag{3.1b}$$

$$E\left[acc([e_1])\right] > E\left[acc([e_1, e_2])\right] > \ldots > E\left[acc([e_1, e_2, \ldots, e_n])\right] \tag{3.1c}$$

where $P(e_x)$ is the true probability of correctly classifying example e_x; $cert(e_x)$ is the certainty of classification of example e_x; and $E[acc(S)]$ is the expected accuracy for the

set of examples S. Equation 3.1a should hold if the examples are correctly ranked. Since true probabilities are difficult to obtain in classification, Equation 3.1b is used to capture the order of classification certainty. Depending on the measure of certainty used, this measure may or may not capture the ranking between classification probabilities. If the ranking is correct, then Equation 3.1c will be true since the accuracy calculated using more likely correct examples is going to be greater than if it is done using those which are less likely to be correct. If all three equations are true, then the curves produced will look like those in Figure 3.2.

The desired behaviour in these graphs is, therefore, to achieve maximum accuracy while still considering a maximum number of examples. In other words, it is desirable to cover as many errors as possible in as few of the test examples as possible (optimally, all the errors are covered first). It is also desirable to have as few classification errors present in high certainty ranges. If these properties are present in the graph, then it means that the ranking method employed is successful in placing the difficult-to-classify examples to the left of those which are easier to classify. If a classifier misclassifies high certainty examples, then it will only achieve high accuracy on a small percentage of the examples, and it would therefore not be a suitable choice for characterization. The curves produced by two different machine learning algorithms can be compared by the degree to which the scores they produce can rank examples. If one method ranks examples better than another, then a larger percentage of examples will be used in calculating a particular accuracy value. This comparison can be made by observing the graphs produced. Examples will be provided for each of the learning algorithms under consideration in the following subsections.

3.2.1 Naive Bayes Scores

As was presented in Section 2.1.1, the Naive Bayes algorithm computes a "probability" for each potential class when classifying texts. Zadrozny and Elkan state in [19] that Naive Bayes scores, although not accurate probability measures, can be used to order examples well. In other words, if $score(C_1) < score(C_2)$, then $P(class = C_1|x) < P(class = C_2|x)$, where x is the example to classify and $P(class = C_1|x)$ is the true probability that example x belongs to class C_1. The score, $score(C_1)$, is the Naive Bayes probability for which the assumption of attribute independence is made.

Naive Bayes scores can be used as follows to obtain a relative ranking of examples. For an instance i, in the context of a binary classification problem, the following probability scores are obtained from an intermediate step in the algorithm:

$$classC_1 : score(C_1) = p_1 \tag{3.2a}$$

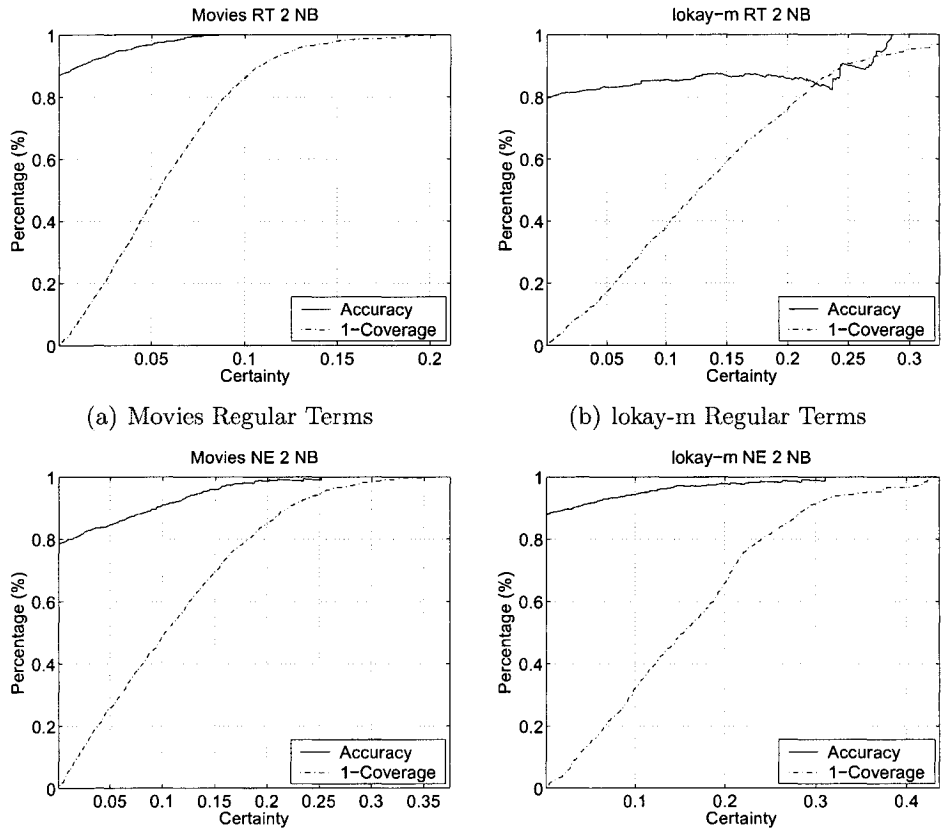
$$classC_2 : score(C_2) = p_2 \tag{3.2b}$$

The next step in the algorithm is to predict the class C_1 or C_2 based on the result of $max\,[p_1, p_2]$. These intermediate values, however, can be used to obtain the certainty measure. If class C_1 has a score greater or equal to that of class C_2, then the certainty of i is given as:

$$certainty(i) = 1 - \frac{1}{\frac{p_1}{p_2}} \tag{3.3}$$

This measure takes into account the magnitude of the separation of the scores. The ranking works since if the ratio of $p1$ to $p2$ is high then the separation is large. By using a ratio, as opposed to a difference, a comparative measure is obtained. For example, if $score(C_1) = 3$ and $score(C_2) = 1$ then class C_1 is 3 times as likely than class C_2.

Using a ratio also ensures that the measure is not dependent on the number of attributes appearing for an example. This property allows for examples to be compared. The ability of the measure to compare examples is the property which enables them to be ranked. The score will produce values between 0 and 1. If the score is 1 it means that $p1$ is infinitely larger than $p2$ and if the score is 0 it means that $p1$ and $p2$ are equal.



(a) Movies Regular Terms

(b) lokay-m Regular Terms

(c) Movies Named Entities

(d) lokay-m Named Entities

Figure 3.3: Characterization of the *movies* and *lokay-m* Datasets with Naive Bayes

This claim that Naive Bayes scores can be used to correctly rank examples is confirmed by the experimental results in Figure 3.3. The results show the accuracy increasing as the

certainty increases for the given expected dependencies. For example, the *movies* dataset and the regular terms curve, the line representing the coverage of examples increases from left to right demonstrating the percentage of examples used to calculate the accuracy value. The coverage line shows that 95% accuracy is achieved when considering 71% of the examples. For the *lokay-m* results, the ranking is also done well and shows a better ranking for named entities than for regular terms. The best plots produced are for the *movies RT* and *lokay-m NE* suggesting that these datasets are regular term and named entity dependent respectively. In these curves, the coverage curve increases rapidly initially, as the accuracy also increases at a faster than linear rate, suggesting that errors are being discarded in ranges of low certainty. These results show that the certainty measure for the Naive Bayes algorithm is successful in ranking the examples given the expected dataset dependencies.

3.2.2 Support Vector Machines (libSVM) Scores

The implementation of the SVM algorithm, libSVM, includes an option to output class probabilities when it makes its classification. These probabilities are computed using a method discussed in [20]. Here, the probabilities for each class are computed using the approximation that:

$$p(y = i|\mathbf{x}) = \frac{1}{1 + e^{Af(x)+B}} \tag{3.4}$$

where A and B are solved by an optimization algorithm. This algorithm uses a subset of the training data and their labels $f(x)$.

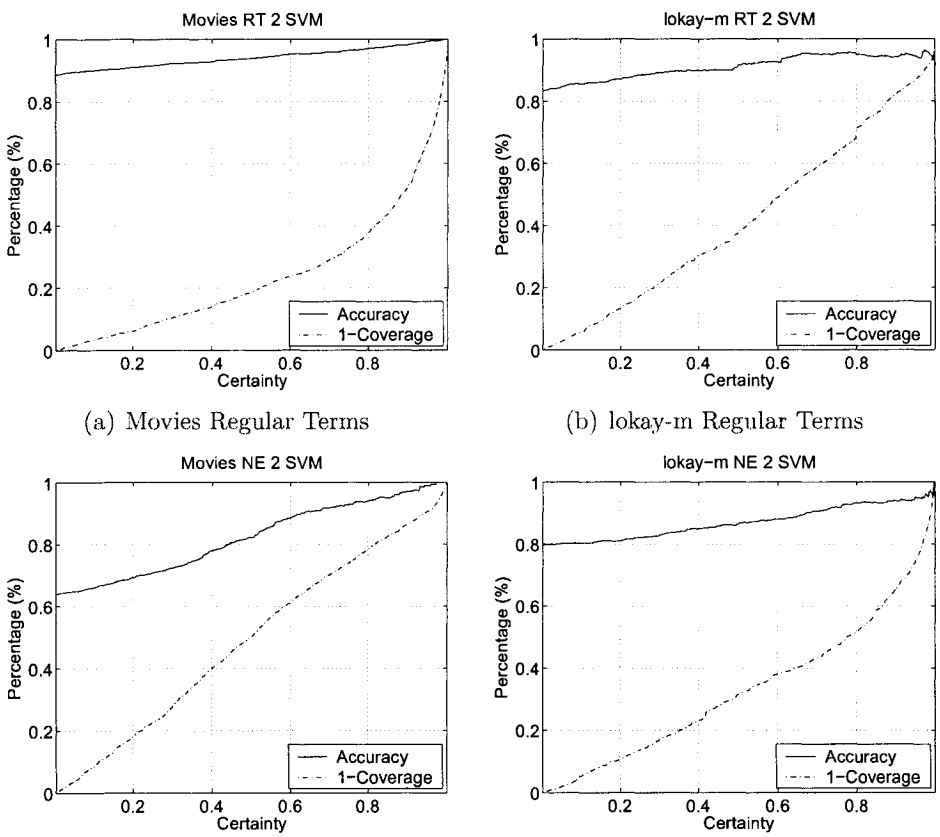
The probability estimates output are between 0 and 1. As a result, for a particular test case in a binary classification task, one class may get a score of 0.8 and the other a score of 0.4. Given the fact that the values output are closer to true probabilities than for

Naive Bayes, small values (such as 0.001) are more common. Thus, it was not possible to use a ratio in order to compute a certainty. The resulting range of values would be too large. A measure that considers the separation of the class probabilities and restricts the range to between 0 and 1 is to simply take the difference between probabilities. The certainty is therefore given as:

$$certainty(i) = score(C_1) - score(C_2) \tag{3.5}$$

The accuracy curves, using the same datasets as in Section 3.2.1 are given in Figure 3.4.

This method of certainty estimation using SVM probability measures, can be judged by observing these results in Figure 3.4, and comparing them with the desired properties. From the figure, the curves agree with the requirements stated in Equations 3.1a-3.1b. However, if the curves are compared with those from Naive Bayes, the ranking produced for the named entities attributes is of lower quality. For the *movies* regular terms graph in Figure 3.4, accuracy increases linearly while the coverage line increases exponentially demonstrating that there are misclassifications produced with high rank. As a result, the measure is not entirely successful at judging if one example is more likely to be correctly classified versus another. This curve implies that errors are distributed throughout the range of certainties instead of being grouped in the low certainty range. For the *lokay-m* plots, the property where the coverage curve is increasing faster than the accuracy curve in the high certainty ranges is also present for both types of attributes. This property demonstrates, as it did with the *movies* plots, that errors are being highly ranked. As a result, the separation of easy- and hard-to-classify examples is not taking place. SVMs will nevertheless be retained as a classifier used in this thesis since they were still able to rank examples, albeit poorly. The results from SVMs will serve to compare with those



(a) Movies Regular Terms

(b) lokay-m Regular Terms

(c) Movies Named Entities

(d) lokay-m Named Entities

Figure 3.4: Characterization of *hockey* and *rt-earn* datasets with SVM

obtained from the Naive Bayes classifier.

3.2.3 Decision Tree (c5.0) Scores

Following the same logic from Sections 3.2.1 and 3.2.2, a certainty measure will be derived for decision trees and the output of c5.0. For decision trees, however, the topic of probability estimation is more of an open research topic. Zadrozny and Elkan argue in [19] that decision trees do not give proper measures, even when pruning is used because this pruning is performed in order to reduce the number of errors and not improve the accuracy of probability estimates. Typically the probability given for a test case is computed as:

$$P(class = C_i|(x)) = \frac{k_i}{n} \tag{3.6}$$

where k is the number of number of examples in a leaf with the label C_i and n is the total number of examples.

They mention a modification that does a little to smooth out the problems of high bias* and high variance† associated with these estimates. The modification implements Laplace smoothing (as proposed by Domingos and Provost in [21]) on the probability estimate so that the estimate is now calculated as:

$$P(class = C_i|\mathbf{x}) = \frac{k_i + 1}{n + 2} \tag{3.7}$$

This new estimate is still likely to have the same problems, but hopefully not to the same degree. This estimate, however, is unlike the others obtained for Naive Bayes and SVM. For a particular leaf, the sum of the probabilities add up to 1 and therefore there is a

*Decision trees try to have homogeneous leaves so the probabilities tend to close to 0 or 1

†The number of training examples in a leaf is small so the probabilities are not statistically reliable

dependence between the two values (since one probability can always be expressed in terms of the other). As a result, a certainty measure using subtraction or division would always be proportional to the majority class in a particular leaf. Therefore, a solution is to take the probability measure obtained by that majority class. This measure, for a binary classification problem, can be expressed as:

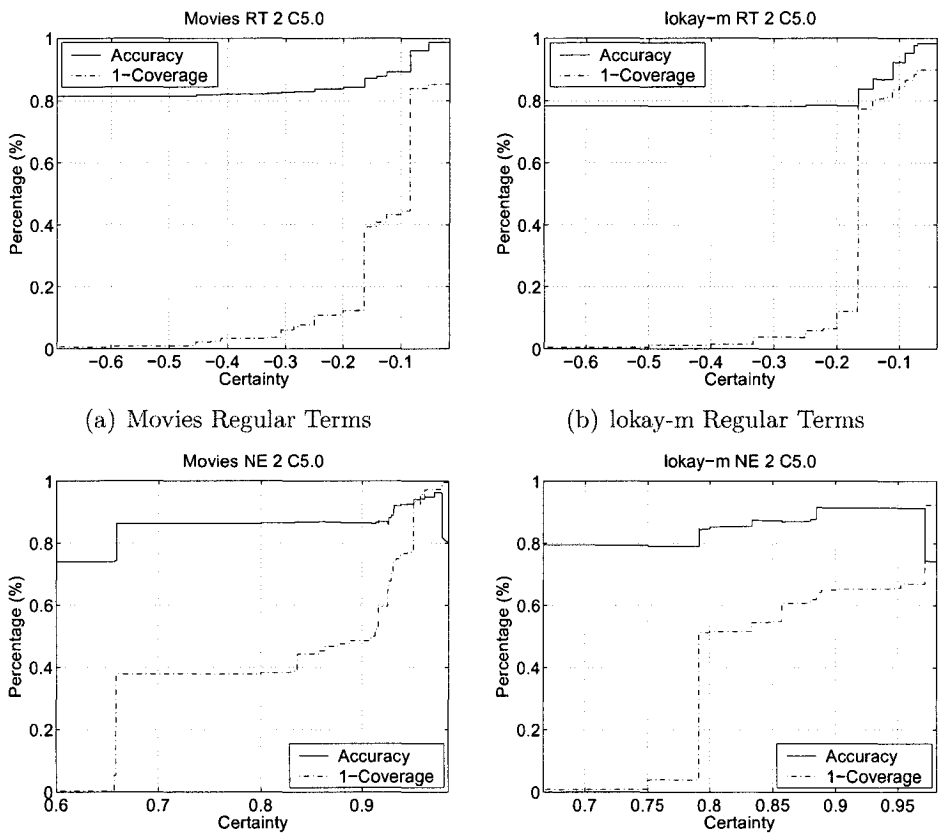
$$certainty(i) = max\left[P(class = C_1|\mathbf{x}), P(class = C_2|\mathbf{x})\right] \tag{3.8}$$

This value will rank examples which fall into high probability leaves as more probably right. It is likely, however, to suffer from the problems of high bias and high variance. The accuracy curves, using the same datasets as in Section 3.2.1 are given in Figure 3.5.

As with the other measures, the method of certainty estimation, this time using leaf probabilities, is evaluated by studying the results. The results in Figure 3.5 show that the certainty measure for decision trees does not share the same desired properties as did the others. Since the decision tree algorithm tries to create pure leaves, as a way of maximizing accuracy, the probability measures do not correspond closely to the true probability of classification. For example, in Figure 3.5, the coverage jumps quickly at times as many examples were given the same certainty despite not sharing any relation in classification confidence. Due to the fact that decision tree probability measures do not facilitate the ranking of examples, this algorithm will not be used in later experiments.

3.3 Characterization

With the ability to rank documents comes the ability to characterize the datasets according to their attribute dependencies. In this section, a method to score a machine



(a) Movies Regular Terms

(b) lokay-m Regular Terms

(c) Movies Named Entities

(d) lokay-m Named Entities

Figure 3.5: Characterization of *hockey* and *rt-earn* datasets with C5.0

learning algorithm given a particular dataset and attribute list will be presented. The general method for characterization is performed by noting the number of documents that receive a high ranking at high accuracy for a particular dataset. If the number of highly ranked documents is large, then the dataset can be said to be dependent on that type of attribute. Conversely, if the number of highly ranked documents is low, then the dataset is said to be independent of the attribute type. The dataset can then be categorized into one of the four groups outlined in the introduction to this chapter. These results will then serve as reference material for the set of experiments presented in Chapter 4. These experiments will determine the effect and reveal the properties of named entities in classification tasks. For example, in order to study the contribution of named entities in regular term dependent classification tasks, the appropriate characterizations must first be performed.

In order to best determine these characterizations, three methods for evaluating attribute dependence will be considered.

1. Accuracy
2. ROC-AUC
3. Weighted ACC-COV

For each of these measures, the values obtained for each of the datasets will be discussed in turn and evaluated in terms of their ability to characterize a dataset. The goal is to find which scoring function performs this task the best. Each will be discussed from a theoretical point of view as well as practically, using the datasets as a reference. Scores will be provided using Naive Bayes and SVM output and will be evaluated in terms of their ability to clearly and correctly distinguish dataset attribute dependence. From

Range	Label
0.0 - 25.0	none
25.0 - 50.0	low
50.0 - 75.0	medium
75.0 - 100.0	high

Table 3.5: Labels assigned to different percentage ranges

the findings in Section 3.2, the results obtained for Naive Bayes will be treated as more representative, and those for SVM included for comparison. The datasets of *hockey* and *rt-earn* will be used in particular to judge the quality of the score. It is necessary that the score clearly reflect the fact that the *hockey* dataset is named entity dependent and that the *rt-earn* dataset is regular term dependent. For the last two measures, a particular score will be preferred if it is able to provide a clear distinction between datasets with comparable overall accuracy for a particular learning algorithm. It is also important that the score reflect the dependence of the entire attribute list as provided. In other words, it should reflect certain properties of the attribute list. For example, if there are unnecessary attributes or if the attributes provide conflicting evidence. The desired outcome is that for a dataset heavily dependent on an attribute type, a value close to 1 is returned. If the dependence is weak, the value is close to 0. This range of values will be divided into four equal parts and labels assigned according to Table 3.5. In order to assure attribute dependence, the values should be a full range apart for each type of attribute, or a full 0.25 or 25%. Such a wide margin is necessary because this work is primarily interested in datasets which are clearly named entity or regular term dependent. The different scores will now be considered as methods to determine attribute dependence.

Dataset	NE	RT	$\|\Delta\|$
Hockey	91.0	78.0	13.0
Movies	78.3	86.9	8.6
20NG-5	92.8	88.3	4.5
Rt-earn	72.9	87.3	14.4
Rt-acq	80.4	97.3	16.9
lokay-m	85.3	79.1	6.2
farmer-d	68.8	76.2	7.4

Table 3.6: Accuracy achieved with NB classifier for different attribute types

3.3.1 Accuracy

The measure of accuracy is well-known to be very general. The authors in [15] provide a good theoretical explanation for why accuracy is not appropriate to measure the quality of a ranked set of classifications. To illustrate this point, consider that a single accuracy value can have a large number of possible rankings. Consider a ranking of 20 classifications. The classifications are either correct or incorrect. If the accuracy is 80%, then there are $\binom{20}{4} = 4845$ possible different rankings. An example of such a ranking of classifications is shown in Figure 3.6.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
+	−	+	+	+	+	+	−	−	+	+	+	+	+	+	+	+	+	−	+

Figure 3.6: Ranked examples

The accuracy scores obtained for each of the datasets under consideration for both Naive Bayes and SVMs are reported in Tables 3.6 and 3.7. With these values, for both Naive Bayes and SVM, there is a sense as to which datasets do better than others on the different attribute lists, but no value is far enough apart that any is solely dependent on one type of attribute. All the values are within 0.25 of each other and hence no clear

Dataset	NE	RT	$\|\Delta\|$
Hockey	92.7	91.4	1.3
Movies	64.1	88.7	24.6
20NG-5	93.9	93.8	0.1
Rt-earn	76.6	97.9	21.3
Rt-acq	85.4	96.4	11.0
lokay-m	79.6	83.3	3.7
farmer-d	70.1	79.4	9.3

Table 3.7: Accuracy achieved with NB classifier for different attribute types

dependence that can be determined from the score of accuracy alone.

It is interesting to note that the SVM classifier does a good job with the *hockey* dataset using regular terms. This result completely goes against the expectation that this dataset is only classifiable with named entities. It demonstrates the capacity of SVMs to find regularities in data which may only be observed in this particular subset of hockey news articles. A classifier trained on named entities would be expected to still do well on data where such coincidental regularities are absent. The Naive Bayes algorithm, however, does not face the same problem. This result demonstrates the ability of Naive Bayes to reflect the suitability of the entire attribute list for the dataset. With the *hockey* dataset there are a lot of regular term attributes such as "goaltender" and "referee" which serve to confuse a learning algorithm such as Naive Bayes.

3.3.2 ROC-AUC

In this section, scores will be computed using a method designed to evaluate the ranking quality of set of classifications. Namely, the area under an ROC curve will be computed. An ROC curve, in this case, is a plot of how many successful classifications are performed for every misclassified example. Thus, it will plot the *true positive rate* (TPR) versus the *false positive rate* (FPR). The method used to calculate these values is illustrated by

considering the ranked set of classifications from Figure 3.6 along with every possible split t of the ranked set. The TPR is then equal to the fraction of all correct classifications that are above the split point and the FPR is equal to the ratio of incorrect classifications to the total number of classifications above the split point. Therefore, Equations 3.9 and 3.10 can be written and used to compute these values.

$$TPR = \frac{TP}{(TP + FN)} \tag{3.9}$$

$$FPR = \frac{FP}{(TP + FP)} \tag{3.10}$$

where TP is the number of correct classifications above t, FP is the number of incorrect classifications above t and FN is the number of correct classifications below t.

To demonstrate how the values of TPR and FPR are calculated, consider a split formed after the data point 5 as shown in Figure 3.7. At this point, $TPR = \frac{12}{16} = 0.75$

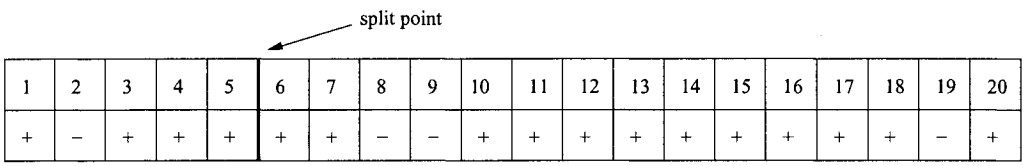


Figure 3.7: An example split point in a set of ranked classifications

and $FPR = \frac{12}{15} = 0.8$. If these values are calculated for every split point of a complete dataset characterization task, an ROC curve like is shown in Figure 3.8 is obtained. The area under this curve, which is generated from the *movies* dataset, can then be calculated. This measure improves upon simple accuracy by considering the ranking of the examples. If the ranking is good (all misclassifications at low rank) then the TPR is continually high (except at low rank split points) across the continuum of values for FPR.

It is this ideal ranking of classifications which is desired and which would demonstrate high dependence on a particular attribute list.

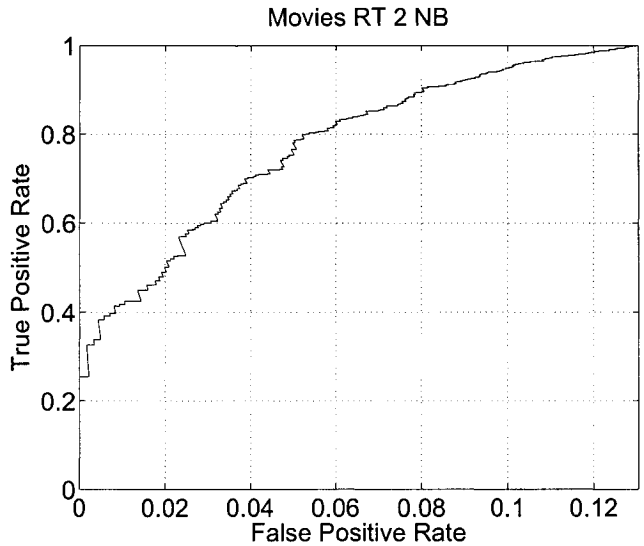


Figure 3.8: An example ROC curve produced by a set of ranked classifications

The AUC values obtained from the ROC curves generated from Naive Bayes and SVM certainties are provided in Tables 3.8 and 3.9 for each of the datasets under consideration. In these tables, the differences between the regular terms and named entities scores are also provided. From this difference, the Naive Bayes score notes the *hockey* and *lokay-m* datasets as named entity dependent. No dataset is convincingly regular term dependent. For the SVM score, only the *rt-earn* dataset is noted to be regular term dependent. From the viewpoint of this score, no datasets are named entity dependent. With these results, the scores have improved over those of accuracy. For Naive Bayes, the differences between the scores for named entities and regular terms has increased, especially for datasets like *hockey*, *movies* and *lokay-m*. For SVM, the separation in the scores has not increased as much. The scores do, however, reflect the difficulty required in ranking classifications

Dataset	NE	RT	$\|\Delta\|$
Hockey	81.5	32.7	48.9
Movies	62.3	77.3	15.0
20NG-5	78.1	79.3	1.2
Rt-earn	58.2	76.4	18.2
Rt-acq	69.6	91.6	22.0
lokay-m	69.7	37.1	32.6
farmer-d	52.7	55.3	2.6

Table 3.8: Percent AUC scores with NB classifier for different attribute types

Dataset	NE	RT	$\|\Delta\|$
Hockey	86.3	63.0	23.3
Movies	61.8	77.6	15.8
20NG-5	80.0	90.1	10.2
Rt-earn	60.8	87.6	26.8
Rt-acq	66.4	86.2	19.8
lokay-m	57.9	58.4	0.5
farmer-d	56.5	61.3	4.8

Table 3.9: Percent AUC scores with SVM classifier for different attribute types

such as those obtained with the *hockey* dataset using regular terms.

Computing ROC-AUC to determine dataset attribute dependence scores does present certain problems. The problems arise since the score evaluate the quality of the ranking regardless of the context of accuracy. In other words, it becomes possible to obtain a high score by producing a perfect ranking. Such a score would be independent of the overall accuracy. As mentioned in Section 3.3.1, a large number of possible ranking patterns can produce the same overall accuracy. Of these rankings, one is the best (all correct classifications ranked above the incorrect ones) and one is the worst (all correct classifications ranked below the incorrect ones). It is therefore possible to obtain a high score in the presence of low overall accuracy since the AUC values are normalized according to these best and worst scores. This phenomenon is not observed directly in the

datasets under consideration, but it still remains possible from a theoretical standpoint. The normalization, however, also affects the AUC score when the overall accuracy is high. When the accuracy is high there are fewer permutations of the number of possible rankings, and so an out-of-place misclassified example becomes more important when the score is normalized. As a result, the few misclassifications cause a poor score to be produced, when a higher one is expected for attribute dependence. The values reported in Tables 3.8 and 3.9 show that for the experiments yielding high overall accuracy, namely *hockey NE*, *rt-earn RT*, *rt-acq RT*, and *20NG-5 RT and NE*, the corresponding ROC-AUC score is lower than expected. Evaluating rankings, as captured by this scoring function, is primarily useful for datasets with mid-range accuracy. With such datasets, the ROC-AUC measure should be low if the ranking is poor. The datasets where this property is observed are *hockey RT*, *movies NE*, and *lokay-m RT* (for Naive Bayes). For these sets, the overall accuracy is around 80%, but the score produced is much less. This evidence underlies the utility of using ranked classifications to determine attribute dependence.

This scoring function, however, could be improved for the purpose of determining attribute dependence if the problems with it could be solved. By incorporating the context of overall accuracy into the computed score (while still retaining the virtues of rank evaluation) an improved function could be derived. Such a scoring function is discussed next in Section 3.3.3, where weights are assigned to different regions of an ROC-like curve according to the overall accuracy for that region.

3.3.3 Weighted ACC-COV

The scoring function derived in this section aims to characterize attribute dependence for a given dataset and classification algorithm. The scoring function applies a weighting function to an ROC-like curve to compute a value in a similar way as compared to the ROC-AUC discussed in 3.3.2. As a result of using this weighting function, more weight or importance is assigned to attribute lists which are able to produce large numbers of highly accurate classifications. In order to be more directly interpretable, this function makes use of a variant of the ROC curve. With this variant, *accuracy* (ACC) and *coverage* (COV) above the threshold t are used (as in Section 3.2) instead of TPR and FPR. These values are in fact closely related to each other. Accuracy is simply equal to $1 - FPR$ and *coverage* is similar to TPR except for the fact that it considers all the cases above the split point instead of just the correct ones. The formal equations for *accuracy* and *coverage* are provided in Equations 3.11 and 3.12.

$$accuracy = \frac{TP}{(TP + FP)} \tag{3.11}$$

$$coverage = \frac{(TP + FP)}{(TP + FP + FN + TN)} \tag{3.12}$$

where TN is the number of misclassifications below t (the rest of the variables retain the definitions stated in Section 3.3.2). Any graph comparing the trade-off between accuracy and coverage is therefore also appropriate for evaluating the ranking ability of a machine learning algorithm. The similarities between the ROC curve in 3.8 and the accuracy-coverage curve in Figure 3.9 are clear. One curve is obtained from the other by simply rotating it.

If this curve is then sampled at regular intervals and multiplied with a weighting

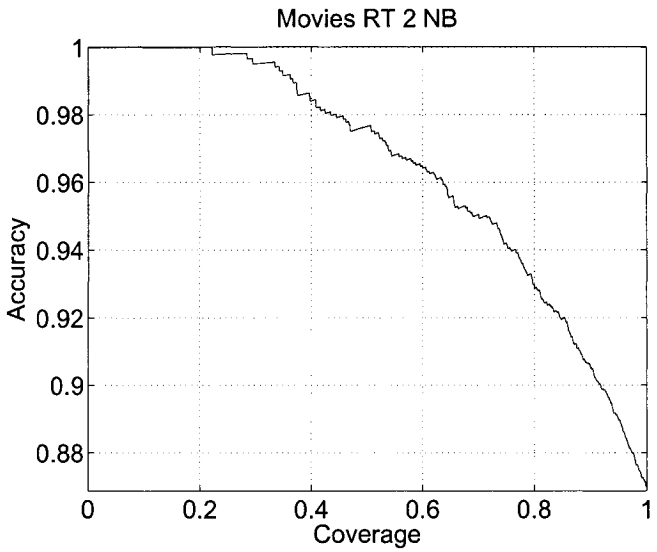


Figure 3.9: An accuracy-coverage curve produced by a set of ranked classifications

function, a score is obtained which can be used to characterize the dataset. In this case, the curve is sampled at intervals of 5% accuracy and the coverage at each point is weighted as specified by the function in Equation 3.13. This weighting function appropriately assigns higher weight to the high accuracy sample points and low weight to the low accuracy ones. The scores produced are then normalized to those produced by the best and worst possible rankings such that the choice of w_1 no longer matters.

$$\text{Weighted ACC-COV}^* = \sum_{i=1}^{20} w_i \cdot C_{(100-5i)} \tag{3.13}$$

where C_x is the coverage or percentage of examples used in computing a percent accuracy value of x. The weights w_i decrease exponentially with x such that $w_{i+1} = \frac{w_i}{2}$. This scoring function makes intuitive sense since if high accuracy can be computed using a

*score prior to normalization

Dataset	NE	RT	$\lvert\Delta\rvert$
Hockey	89.7	18.5	71.2
Movies	57.3	83.9	26.7
20NG-5	97.4	89.4	8.0
Rt-earn	44.9	79.4	34.5
Rt-acq	74.5	100.0	25.5
lokay-m	77.7	25.4	52.3
farmer-d	46.0	39.8	6.2

Table 3.10: Percent Weighted ACC-COV scores with NB classifier for different attribute types

large set of ranked classifications, then the score will be high. If high accuracy can only be computed using a small set of highly ranked classifications, then the score will be low.

The values produced for this scoring function on the datasets under consideration are provided in Tables 3.10 and 3.11 for both Naive Bayes and SVM rankings. The values in these tables show improvement for experiments with high overall accuracy. Namely, the scores for *hockey NE*, *rt-earn RT*, *rt-acq RT* and *20NG-5 RT and NE* are higher than those for ROC-AUC. Also important is that for the datasets with expected dependence (*hockey*, *rt-earn*), the separation in the values is more clearly defined for both Naive Bayes and SVM rankings. This score also preserves the benefit which comes from evaluating rankings as opposed to accuracy as discussed in Section 3.3.2. The datasets of *hockey RT*, *movies NE*, *lokay-m RT* all have around 80% accuracy for Naive Bayes, but produce bad rankings and low dependence scores with weighted ACC-COV.

This scoring function demonstrates an ability to provide a good separation of values which enables the datasets to be characterized. The values obtained using this scoring function for the *hockey* and *rt-earn* datasets provide the best distinction of the attribute dependencies. Such characterizations can be used to reason about document classification with different attribute types as will be done in Chapter 4. An as example of the type of

Dataset	NE	RT	$\|\Delta\|$
Hockey	97.2	84.6	12.6
Movies	44.8	87.7	42.9
20NG-5	99.0	98.8	0.3
Rt-earn	45.1	100.0	54.9
Rt-acq	69.7	100.0	30.3
lokay-m	53.1	64.8	11.7
farmer-d	41.0	54.8	13.8

Table 3.11: Percent Weighted ACC-COV scores with SVM classifier for different attribute types

reasoning possible, consider the following example. If a dataset is highly dependent on attribute type A and less dependent on attribute type B, then it is possible to improve classification by discarding attribute set B. Such a claim is difficult to make based on an accuracy measure alone since it does not look at the distribution of the classified examples. With this measure, accuracies obtained on different attribute types can be relatively close (as with the *lokay-m* dataset), but produce highly different characterization scores, demonstrating the true nature of the relationship between data and attributes. For the example here, by virtue of discarding an entire set of attributes, training and classification time for the dataset can be reduced. Classification accuracy could be improved as well.

3.3.4 Results

The formal characterizations as derived from the Naive Bayes weighted ACC-COV scoring function are provided in Table 3.12. The numbers in the dependence column correspond to the different types presented in the introduction to this chapter. The results show that of the seven datasets, two are clearly named entity dependent. The first is the *hockey* dataset where there is a marked advantage in using named entities. From the classification task (to distinguish news articles about professional hockey from those of

Dataset	NE	RT	Dependence
Hockey	high	none	1
Movies	medium	high	2
20NG-5	high	high	3
Rt-earn	low	high	2
Rt-acq	medium	high	2
lokay-m	high	low	1
farmer-d	low	low	4

Table 3.12: Final characterizations for the datasets

junior hockey) it is apparent that regular terms are not of much use since both classes deal with the same general subject matter. The best way to discriminate between the two classes is with attributes such as the names of teams, players, administrators, and cities. The second is the *lokay-m* dataset where people names and organizations play an important role in deciding which label to assign.

The Reuters datasets are all regular term dependent. This fact is in agreement with the research conducted by [8]. The work suggests that since the data was hand-labeled, the underlying classification process is keyword-based. The results presented here confirm these results as the *rt-earn* and *rt-acq* are reported to be regular term dependent. The same is the case for the *movies* dataset where keywords appear regularly enough to help in the classification. Named entities also occur very frequently, but it is usually in combination with the more statistically significant keywords.

The dataset, *20NG-5*, holds the unique position of being the one where both regular terms and named entities are significant in classification. This result makes sense considering the characteristics of the data. This particular dataset has well-defined categories and was only collected over the span of a month, and therefore there was not much opportunity for the creation of sub-topics. In contrast, the *farmer-d* dataset is lightly dependent on both named entities and regular terms. For such a dataset, the low

scores are evidence that neither document representation is able to adequately classify the test set. The dataset can still serve a useful purpose since the performance of the two attribute types is comparable.

The results for the SVM classifier show a difficulty in characterizing named entity dependent datasets. At the root of this result are its problems with document ranking as presented in Section 3.2. This classifier does a good job at weighting features in order to produce the best accuracy results. Unfortunately for characterization, a general *impression* of the data given the attribute list is what is desired. If the classifier is performing its own feature selection, then this will tend to destroy the nature of the representation of the data as it is presented to the classification algorithm. For the regular term dependent datasets considered in this thesis, a high weighted ACC-COV score is largely due to the high overall accuracy obtained with regular terms. For the named entity dependent datasets (as determined by Naive Bayes), there is not such an overwhelming difference accuracy and the ability of the algorithm to rank examples becomes more important. In these cases, SVM does not produce a good characterization since its rankings are not as good. For this reason, the task characterizations produced by the Naive Bayes classifier are preferred over those produced by the SVM classifier.

These results confirm the ability of this Naive Bayes Weighted ACC-COV method to correctly characterize datasets as either named entity dependent or regular term dependent. The characteristics were found to be applicable to the dataset itself, and did not apply to dataset genres. The email data, for example, produces different results for the different sets due to the fact that classification labels were assigned based on the user's personal preference. Also demonstrated by the above results is that these datasets serve as a useful data repository since they cover the full range of possible characterizations.

This diverse array of datasets will enable the meaningful study of named entities, which is presented in the next chapter.

The main benefit to these characterizations is that it provides both a reference and a justification for the experiments conducted in Chapter 4. It allows datasets to be argued about in terms of their attribute dependence. The method used for determining attribute dependence relies solely on the performance of the attribute itself. It is only when the dataset as a whole, given a set of different possible attribute lists, is characterized that the results from the lists are compared. The method also successfully shows that Naive Bayes is a better algorithm to use for doing characterization because it ranks the documents in a manner which is more representative of the attributes specified.

Chapter 4

Effect of Named Entities

This chapter provides an analysis of how named entities can be used successfully in text classification tasks. Three different factors limiting their utility in classification have been identified. These are:

1. The relative frequency between named entities and regular terms
2. The different possible representations for named entities
3. The greater dependence on time for named entities than for regular terms

The experiments presented in this chapter will test each of these factors in turn. For the first factor, the accuracy obtained from individual classification with named entities and regular terms will be compared with that from a combined feature list. The belief here is that named entities, because they are less frequently occurring in documents than regular terms, are overshadowed by them in classifier training. The second factor will investigate whether single word representation is sufficient for named entities. The hypothesis for this factor is that given a particular task it is possible to reduce ambiguity

and improve classification accuracy by representing named entities by a single element rather than by their component words. The third and final factor of interest will show how named entities as an attribute type are more dependent on time and must receive special consideration when deciding to retrain a classifier. Such a decision has a direct impact on maintaining classification accuracy over time.

For all of these experiments, the characterizations from the previous chapter will be used to help guide the experiments. Based on the results obtained, methodological changes can be made to enable named entity classification to overcome its limiting factors and improve accuracy. For completeness, each experiment will also provide weighted ACC-COV scores to demonstrate attribute list dependence as discussed in 3.3.3.

4.1 Attribute Frequency

Machine learning algorithms depend heavily on the relative frequency of attribute values in classifier training. In other words, an algorithm, especially one based on statistics, favours attributes that occur often in the training documents as opposed to those which are seen rarely. The difficulty with this type of approach is that it could potentially ignore attributes that do not occur frequently, but are all the same useful for classification. The experiments conducted in this section will aim to show that the relative low frequency of named entity occurrence, as compared with regular terms, directly and negatively impacts their influence in text classifier training. For example, if a classification task is named entity dependent, classification accuracy on a test set is likely to be reduced by the inclusion of regular terms in training because they are more statistically important.

4.1.1 Experimental Setup

The *de facto* standard for attributes in text classification makes no distinction between the different types of attributes by combining them all together. The experiments here will determine the contribution of each type of attribute in a general classification task. The objective for these experiments is to show that for named entity dependent classification, named entities need to be considered in isolation from regular terms. This separation is needed in order to ascertain whether or not they are being overshadowed by the more frequent regular terms. If this is the case, then classification accuracy can be improved by training on named entities alone. There is also the possibility of a performance gain since having fewer features reduces training and classification time. The method requires three different classifications to be performed:

1. General classification (control experiment)
2. Regular term classification
3. Named entity classification

For each experiment, a different attribute list is constructed. The named entity and regular term lists are built first and then they are combined to create the list for the general classification task. With the data available, half will be used for training and half will be used for testing. The findings will report the accuracy achieved using each attribute list. The results of the general classification will be compared with those of named entity and regular term classification. There are two different expected outcomes:

1. Named entity classification accuracy is better than that of general classification
2. Regular term classification accuracy is comparable to that of general classification

The first case implies that the contribution of the regular terms was enough to lower the accuracy of general classification. This claim is expected to be supported by the evidence that the dataset under experimentation is named entity dependent. The second case should apply to regular term dependent classification where named entities have little effect in general classification. The two cases demonstrate the reliance of machine learning algorithms on the frequency of attribute occurrence. The conclusion that this effect is due to the frequency imbalance between named entities and regular terms (that regular terms occur more frequently than named entities) will be confirmed with a comparison of the attribute densities from the different lists. This comparison is provided in Section 4.3.5.

The above method will be carried out on the datasets found to be either regular term dependent or named entity dependent according to the results from Chapter 3. These datasets are those where the frequency of attribute occurrence can be best studied. In addition, it is for these datasets that ignoring a whole class of attributes can lead to performance gain (shorter training and classification time) since the number of attributes can be significantly reduced.

4.1.2 Results and Discussion

The first results provided are those for the regular term dependent classification tasks. In Table 4.1, the top three rows present the overall accuracy obtained for named entities, regular terms and general classification where the two lists are combined. The results, for the Naive Bayes classifier, show that the accuracy of the combined list roughly tracks the accuracy of the regular term classifier. This result demonstrates two things. The first is that these datasets are in fact regular term dependent. The second is that named entities

Dataset	NE	RT	Combined
Movies	78.3	86.9	86.6
Rt-earn	72.9	87.3	87.2
Rt-acq	80.4	97.3	97.5
Hockey	91.0	78.0	82.6
lokay-m	85.3	79.1	81.8

Table 4.1: Accuracy achieved with NB classifier for different attribute types

occur less frequently and are therefore correctly ignored in the general classification task. This fact implies that named entities could be removed from the attribute list with no loss in classification accuracy.

The second set of results are those for the named entity dependent datasets. These results are provided in the last two rows of Table 4.1. Here, the results for the combined list are similar to those for the regular term dependent datasets. The accuracy of the combined classifier is again closer to the accuracy of the regular term classifier. It is the more frequent occurrence of regular terms with respect to named entities which causes the classification accuracy to be lower for the combined attribute task. Hence, named entities are ignored to a certain extent even in datasets where they are the most important. Therefore in these datasets, better performance is achieved when the most frequent attributes, the regular terms, are ignored and named entities are not overshadowed. The total reliance on named entities for classification is only something which can be done if the dataset is first characterized as being named entity dependent. Otherwise, there would be no indication that named entities would be useful in classification.

The above findings are supported by the weighted ACC-COV scores reported in Table 4.2. The results are presented as percentages in order to keep the same format as used for the accuracy results. Here, the scores show that when the attributes are combined a lower value is obtained for the named entity dependent datasets. For the regular term

Dataset	NE	RT	Combined
Movies	57.3	83.9	79.4
Rt-earn	44.9	79.4	79.4
Rt-acq	74.5	100.0	100.0
Hockey	89.7	19.7	74.9
lokay-m	77.7	25.4	29.3

Table 4.2: Percent weighted ACC-COV score with NB classifier for different attribute types

Dataset	NE	RT	Combined
Movies	64.1	88.7	88.8
Rt-earn	76.6	97.9	97.5
Rt-acq	85.4	96.4	96.2
Hockey	92.7	91.5	91.3
lokay-m	79.6	83.3	83.4

Table 4.3: Accuracy with SVM classifier for different attribute types

dependent datasets, there is little or no effect.

In Table 4.3, the results obtained using an SVM classifier are presented using the same format as in Table 4.1. For all of the datasets, the results resemble those observed when using the Naive Bayes classifier. The results demonstrate the fact that regular terms are more statistically important than named entities when they are combined. In Table 4.4, the weighted ACC-COV values are displayed. Similar to those in Table 4.2, they show that the *hockey* dataset negatively affected by the inclusion of regular terms. The result for *lokay-m* is different in the respect that for SVMs, regular terms perform better. The improvement in the weighted ACC-COV measure is likely due to the fact that including named entities to the attribute list improved the certainty of correct classifications. In other words, the same number of errors were made but the correct classifications were given higher rank.

From these results, a weakness of statistical text classification methods is exposed.

Dataset	NE	RT	Combined
Movies	44.7	87.7	87.1
Rt-earn	45.1	100.0	100.0
Rt-acq	69.7	100.0	100.0
Hockey	97.2	84.6	92.3
lokay-m	53.1	64.8	69.3

Table 4.4: Percent Weighted ACC-COV scores with SVM classifier for different attribute types

Classification is sometimes best performed by attributes which are not necessarily the most statistically significant. This result favours the development of a form of feature selection unlike those of filters and wrappers, since it is based on the semantics of the constructed attribute lists. If the dataset is found to be named entity dependent, then several thousand attributes can be ignored. The reverse is true for regular term dependent datasets. An interesting fact about constructing named entity attribute lists is that there is no loss in generality since they can be found in almost any kind of text. The findings presented here are therefore potentially useful for text classification in general.

4.2 Attribute Representation

The question of how to represent named entities for text classification tasks is key since it directly impacts on classifier accuracy and interpretability. In this section, two possible representations will be considered. The first representation consists of using the component words of named entities as attributes, a typical choice for most representations. The second representation is to retain the named entity as a sequence of words. The objective here is to determine whether or not single word representation is an acceptable form of document representation in named entity dependent tasks. Single words benefit in terms of simplicity and generality. If, for example, single word representation is adequate for

named entities, then this representation could help to reduce the number of attributes.

This experiment requires datasets which are dependent on named entities. In using these datasets, the results are assured to be useful and interesting. If a regular term dependent task were to be considered, the representation used for named entities would be sure to have no effect on the final classification. This fact is in agreement with the findings presented in Section 4.1. In other words, the impact of named entity representation will be evaluated on datasets where it could actually make a difference in the ultimate performance of the classifier.

4.2.1 Experimental Setup

The setup for this experiment is similar to the one described in Section 4.1. The data will be separated into two equal parts. One for training and one for testing. Since the datasets must be dependent on named entities, the *hockey* and *lokay-m* sets will be used. The single word representation (NE-BOW) will be constructed from the attributes of the complete named entity list (NE). Classifiers will be trained and tested using the two forms of attributes. The results for this experiments are presented next.

4.2.2 Results and Discussion

The accuracy results for this experiment are presented in Tables 4.5 and 4.6. Also available for the datasets, in Table 4.7 and 4.8, are the weighted ACC-COV scores. These values give a better idea of how suitable an attribute list is for classifying the documents of the dataset.

In terms of accuracy, conflicting results are obtained. Whereas the *hockey* dataset shows little (slight decrease) or no difference in accuracy, the *lokay-m* dataset shows a

Dataset	NE	NE-BOW
Hockey	91.0	91.8
lokay-m	85.3	78.2

Table 4.5: Accuracy for named entity representation comparison for NB classifier

Dataset	NE	NE-BOW
Hockey	92.7	93.1
lokay-m	79.6	75.2

Table 4.6: Accuracy for named entity representation comparison for SVM classifier

Dataset	NE	NE-BOW
Hockey	89.7	93.4
lokay-m	77.7	70.8

Table 4.7: Percent Weighted ACC-COV scores for named entity representation comparison for NB classifier

Dataset	NE	NE-BOW
Hockey	97.2	96.8
lokay-m	55.1	56.0

Table 4.8: Percent Weighted ACC-COV scores for named entity representation comparison for SVM classifier

marked one. The weighted ACC-COV scores for Naive Bayes in Table 4.7 confirm these findings. In this table, the NE-BOW score is slightly better for the *hockey* dataset whereas it is worse for the *lokay-m* dataset. If there is a difference in the values it implies that the ranking procedure was not as successful because the classifier did not have quality attributes. The results imply that the classifier trained on NE-BOW data produced more lower certainty classifications. The weighted ACC-COV scores reported for the SVM classifier in Table 4.8 can be disregarded since on these datasets, the algorithm showed no ability detect named entity dependence. The Naive Bayes algorithm is preferred in this case. Further investigation into the characteristics of the datasets reveals the reason behind these results, in terms of why for the *hockey* dataset, NE-BOW performs slightly better and for *lokay-m*, the NE representation is best.

For the *hockey* dataset, the scope of the classification task is of significant importance. If this binary classification task were to be considered as part of a hierarchical classification task, it would result in the formation of two leaves in the classification tree. In other words, given the task, the clear context of *hockey* articles exists. Therefore, when the named entities are analyzed, for the most part those which consist of multiple words, are in fact unambiguous if a single word representation is used. For example, consider the attribute of "Martin Brodeur." This person is a well-known personality in the hockey world. There is only one player with this last name. Therefore if this named entity is represented by the single word token of "Brodeur," then there is no loss of information, given that the classification context remains as hockey. It seems logical that well-known players would have unique names since there are fewer of them and they would need to stand out. The same applies for other attributes like team names and cities. Consider the team "New Jersey Devils". An acceptable representation is that of "Devils," because

in the hockey world, this word only has a single interpretation. It is as a result of how the dataset was constructed that no articles on theology, for example, are included. For place names, consider the attribute "Medicine Hat." The word "hat" may be ambiguous in hockey, but not if the attributes consist only of named entities. Here, either component word is an acceptable representation of the multi-word attribute.

In contrast, the *lokay-m* dataset constitutes a much more general classification task. It deals with documents in the email domain where there is no restriction on the topics discussed. As such, there is no implied context and named entities are more likely to be ambiguous if they are represented by their component words. This dataset therefore has two interesting properties. The first, as just mentioned, is that it presents itself as a general classification task. The second is that it is dependent on named entities. In such a dataset, the attributes can not always be represented by single words since this introduces ambiguity across attributes. It is therefore better in such a case to represent named entities with their full multi-word representation.

These results show the usefulness of determining the sensitivities of datasets. Some datasets may not require special attention for their attributes because of how the classification task is defined. In such a case, single word representation is acceptable. In other classification tasks, this representation may introduce class ambiguity and decrease accuracy. These experiments presented here, serve as evidence that it is important to determine whether the attribute representation selected is acceptable. This decision can be made by testing the alternatives and comparing the accuracies obtained for each.

4.3 Attribute Time Dependence

The concept of time is important in text classification since data is often collected and classified as it becomes available. Named entities in particular could be more vulnerable to the effects of time. Their occurrences are reported to be clustered in time, and as a result found to be particularly useful in *topic detection and tracking* (TDT) tasks [12]. Such a dependence on time, is expected to cause classification accuracy to decrease as time moves forward. This effect of time is not always given much attention in research projects due to data and time constraints and is typically left as future work as in [2]. Nevertheless, it is necessary to address time if text classification methods are to be used in practice. Classifiers are, after all, trained to categorize previously unseen and unlabeled documents. This section presents two goals. The first goal is to show that named entities are made less useful in certain classification tasks as a result of their dependence on time. The second goal will show that classifier retraining is a suitable solution for the negative effects of named entity time dependence. The motivation for these goals is to identify a difference between named entities and regular terms and to further specialize how named entities are used in text classification.

This section will first present an introduction into the causes of accuracy decay over time. Given a classifier, there are three main reasons for why its ability to classify would decrease over time:

1. Classifier overfitting
2. Concept drift
3. Novelty introduction

Each of these causes will be briefly introduced in Sections 4.3.1-4.3.3 along with any sub-causes which may exist. These causes will then be evaluated in practice by comparing how classifiers trained on named entities and regular terms behave as new data arrives. Next, classifiers for each attribute type will be retrained at regular intervals in order to get an idea of how accuracy is affected. The expectation is that by retraining the classifier, named entities will be less affected by time and accuracy will be improved. Finally, an experiment will be presented which aims to show that named entities are primarily affected by concept drift and novelty introduction. This experiment is important because it shows that a retraining scheme is allowable since it is not encouraging overfitting.

4.3.1 Overfitting

Overfitting is an interesting concept for classification. It is caused by one of two things. The first is the over-specialization or over-generalization of a classifier, which occurs when the classification algorithm is not able to properly learn a concept. The second is when the classification algorithm gets confused by noisy data and includes this noise in the trained classifier. For the first cause, assuming a perfect machine learning algorithm, the classifier mislabels the document due to a lack of training material. For example, say a document is classified as *true* if it contains the word "banana." If the learned concept classifies a document to be *true* if the sequence "yellow banana" appears and *false* otherwise, the classifier has overfit the training data. If the sequence "green banana" were to appear, it would be misclassified. For noisy data, a more difficult problem to deal with, the hope is also that with more data, the noise could be ignored and a proper classifier can be trained. The interesting fact about overfitting is that the current method to detect it is to test the classifier on data occurring after those used for training and to

check for a drop in performance. It is interesting because none of the causes for overfitting has anything to do with time directly. The causes are more related to the quality of the training set than to any pathological manifestation of time.

4.3.2 Concept Drift

The second major cause of classifier performance decay is due to what is known as concept drift. To illustrate the idea, consider the following example. A set of training examples are collected during the summer months in a seasonal climate. The trained classifier would rely on terms associated with summer. If this same classifier is used on examples obtained during the winter months, then the context has changed and performance is expected to be poor. Consider the term "hockey" as being important to classification. During the summer it would be associated with field hockey and in the winter with ice hockey. Here the performance would suffer since examples containing the word "hockey" would be classified as ice hockey documents when they should not be. The implied context provided by each season therefore plays a role in the meaning of the attributes. This example demonstrates the effect of attribute-class dependency shift. In other words, the concept shifts because the attribute values in the training set were generated by a different process from the one operating in the testing set.

Another type of concept drift happens when the attributes in the training set occur less frequently as the classifier "ages." Not to be confused with overfitting, which relates to the generalization/specialization of the trained classifier, the concept learned becomes obsolete as the terms used to describe it change. In this case, the classifier needs to be retrained with the new terms needed. This problem is related to the one discussed next, which tackles the difficulties introduced when the best terms for classification do

not appear in the training set.

No matter which type of concept drift presents itself in the classification task, the primary method for detecting it, as is the case with overfitting, is through monitoring the classifier's performance on the testing data.

4.3.3 Novelty Introduction

The issue of novelty in text classification tasks is an interesting one as well. It is related to the problem of novelty detection, but here only the negative effects of novelty are discussed. Novelty is something easily studied in the area of text classification since language is not a bounded environment. New words are invented all the time to describe new ideas or concepts. The difficulty with the current train-and-test method of text classification is that these words cannot be included in the training phase for the classifier. If the occurrence of these new terms becomes significant and the words are useful for classification, concept drift is said to occur*.

As with the other time dependent issues with text classification, classifier performance is the main criterion for deciding when novelty or anything else has become serious enough to warrant classifier retraining.

4.3.4 Experimental Setup

The main goal here is to investigate what part time plays on the effectiveness of attributes to correctly classify documents. The question to answer here is, given a dataset which attribute type leads to a faster decay in performance over time. If time is less of an issue

*The problem is presented separately here to highlight the two-sides to the problem. The one side is that the terms in the training set are occurring less frequently and the other side is that other terms are replacing them.

for a particular attribute, then it can be considered to be a better attribute for that dataset. If on the other hand, better performance can be achieved by using a different attribute type and regularly retraining the classifier, then maybe it is possible to use these attributes. The difficulty with this type of experimentation is that the same measures are being used to test several different and interdependent issues in text classification. There is no way to be 100% sure that the reason for classifier performance decay is due to any one in particular. If attribute classes, such as regular terms and named entities, are considered however, then better insight into the problem can be achieved.

The experiments will make use of all the datasets and their properties. The data will be sorted in time (as with the other experiments), and split into four parts or blocks. The amount of data for each of the blocks will be made as equal as possible. The advantage with this type of split of the data is that equal amounts are available for training and testing. The downside is that for some blocks, the range in the time value (from when the block starts and ends) may be greater. For the majority of the datasets, this is not a problem since the data is distributed uniformly over time. For those datasets like *rt-earn* and *rt-acq*, where there is a big gap in the data, the separation in time was respected and some blocks, as a consequence, were given unequal amounts of data. The differences were not too dramatic, however, since in the worst case, one set had a few hundred more examples (the total set has about 10,000) than another. For the first experiment, a classifier will be trained on the least recent data and tested on the three remaining sets, simulating document arrival. Another experiment will attempt to help mitigate the effect of concept drift by training three separate classifiers and testing them on the next block of data. For example, training on block 1 and testing on block 2. As a final experiment, the metric of attribute density will be used to help identify when a learned

concept becomes obsolete. The goal is to demonstrate that over time, attributes tend to occur less and less as a result of concept drift. The metric of attribute density considers the occurrence and absence of attributes. If for a given $n \times d$ attribute-document matrix, there are a total of nd possible values. The attribute density is defined as the total number of occurrences m divided by nd or

$$\text{attribute density} = \frac{m}{nd} \tag{4.1}$$

The results of these experiments are presented in the next subsection.

4.3.5 Results and Discussion

These results are presented in three parts. The first part provides the results for the experiment to see which attributes cause a learned classifier to decay faster over time. The second part shows the performance of the classifier when regular retraining is conducted. The third and final results section provides the attribute density measures for the first set of results.

One Classifier

Since the data was split into four blocks (labeled D_i, $i = 1, 2, 3, 4$). the first block is used for training. The subsequent blocks are used for testing. Accuracy is reported in Table 4.9 and Table 4.10 and those for the weighted ACC-COV measure are reported in Tables 4.11 and 4.12 for both types of attributes.

From these results, the average change in accuracy and weighted ACC-COV score can be computed for each test set. These values are reported in Tables 4.13 and 4.14. The values represent the average change with respect to that calculated on the training set

Dataset	NE				RT			
	D_1	D_2	D_3	D_4	D_1	D_2	D_3	D_4
Hockey	93.0	89.5	93.6	86.7	81.4	79.3	83.7	70.1
Movies	92.5	81.7	79.2	73.1	94.3	87.5	85.8	87.1
20NG-5	92.3	87.2	80.6	89.6	83.1	81.8	86.3	93.3
Rt-earn	74.3	69.7	64.2	68.6	88.8	86.8	82.7	87.7
Rt-acq	82.1	75.3	71.4	74.9	95.9	94.5	94.4	96.3
lokay-m	85.9	78.3	82.9	85.0	80.6	75.9	73.6	79.8
farmer-d	72.0	68.1	63.1	60.3	82.3	75.2	81.0	75.1

Table 4.9: NB classifier accuracy on sets of data ordered in time

Dataset	NE				RT			
	D_1	D_2	D_3	D_4	D_1	D_2	D_3	D_4
Hockey	93.5	91.6	94.8	87.2	93.2	93.4	93.4	85.7
Movies	81.4	72.3	65.3	60.5	93.2	87.9	87.5	81.4
20NG-5	94.2	90.1	84.2	93.0	96.8	91.8	90.2	96.2
Rt-earn	73.0	72.9	67.5	74.0	96.0	95.3	95.5	97.5
Rt-acq	84.8	80.3	78.4	82.0	94.4	93.2	93.3	95.8
lokay-m	84.1	80.1	76.2	79.1	85.4	81.5	80.1	84.1
farmer-d	74.2	70.4	67.7	62.5	77.7	72.5	79.3	75.2

Table 4.10: SVM classifier accuracy on sets of data ordered in time

Dataset	NE				RT			
	D_1	D_2	D_3	D_4	D_1	D_2	D_3	D_4
Hockey	98.3	95.3	98.4	65.7	32.0	21.9	39.9	3.3
Movies	93.6	68.2	58.3	46.5	99.1	79.2	80.1	81.6
Rt-earn	48.3	35.1	21.2	34.5	88.5	84.3	72.8	81.6
Rt-acq	79.9	46.9	41.0	45.7	100.0	99.7	99.8	100.0
20NG-5	99.6	34.3	35.4	94.9	83.8	72.1	78.2	99.2
lokay-m	91.8	83.5	80.1	71.3	37.7	21.9	10.4	12.4
farmer-d	62.5	49.6	48.7	20.2	38.8	46.5	48.1	29.2

Table 4.11: NB percent Weighted ACC-COV scores on sets of data ordered in time

Dataset	NE				RT			
	D_1	D_2	D_3	D_4	D_1	D_2	D_3	D_4
Hockey	97.5	92.5	99.4	88.2	98.2	97.8	95.7	67.6
Movies	79.8	48.4	54.9	37.1	97.5	83.2	85.0	79.7
20NG-5	99.1	86.9	72.2	94.2	100.0	94.1	84.8	100.0
Rt-earn	32.2	30.4	31.3	48.9	100.0	100.0	100.0	100.0
Rt-acq	65.4	38.1	33.5	67.2	99.8	97.0	97.3	100.0
lokay-m	73.5	58.9	44.0	63.6	80.8	32.7	27.7	57.7
farmer-d	40.3	37.0	38.9	29.1	58.1	20.1	37.8	18.1

Table 4.12: SVM percent Weighted ACC-COV scores on sets of data ordered in time

Measure	NE			RT		
	D_2	D_3	D_4	D_2	D_3	D_4
Accuracy	−6.04	−8.16	−7.70	−3.63	−2.70	−2.43
Weighted	−23.0	−27.3	−27.9	−7.8	−7.2	−10.4

Table 4.13: Average change in classifier scores over time for NB

D_1. From these results, the accuracy of classifiers trained on named entities are observed to decay faster over time than those trained on regular terms. As for the weighted ACC-COV values, those for Naive Bayes show similar decay, only the values are more pronounced as the distribution of classification moves toward lower uncertainty. The results for the SVM algorithm show not much difference between the values for named entities and regular terms. This outcome can be attributed to the fact that for SVM, the misclassifications as introduced by the effects of time, tend to be in the lower ranges of certainty as opposed to Naive Bayes where time effects those in the high range as well.

Measure	NE			RT		
	D_2	D_3	D_4	D_2	D_3	D_4
Accuracy	−3.93	−7.30	−6.70	−3.01	−2.49	−2.97
Weighted	−13.7	−16.2	−8.5	−15.6	−15.2	−15.9

Table 4.14: Average change in classifier scores over time for SVMs

Dataset	NE			RT		
	D_2	D_3	D_4	D_2	D_3	D_4
Hockey	89.5	95.1	86.6	79.3	85.1	66.8
Movies	81.7	78.5	79.2	87.5	87.1	84.7
20NG-5	87.2	88.8	92.1	81.5	87.1	84.8
Rt-earn	69.7	64.8	68.2	86.8	80.4	88.1
Rt-acq	75.3	73.7	72.8	94.5	95.1	97.4
lokay-m	78.3	78.3	82.2	75.9	77.4	81.3
farmer-d	68.1	67.8	76.3	75.2	76.2	77.4

Table 4.15: Accuracy for NB classifier if regularly retrained

Dataset	NE			RT		
	D_2	D_3	D_4	D_2	D_3	D_4
Hockey	91.6	96.0	90.4	93.4	93.3	83.0
Movies	72.3	70.2	75.3	87.9	89.2	86.8
20NG-5	90.1	90.1	92.2	91.8	92.0	96.7
Rt-earn	72.9	66.9	73.8	95.3	95.7	96.3
Rt-acq	80.3	78.6	83.2	93.2	93.6	95.2
lokay-m	80.1	74.7	83.4	81.5	79.8	84.0
farmer-d	70.4	72.6	70.3	72.5	79.6	75.1

Table 4.16: Accuracy for SVM classifier if regularly retrained

Retrained Classifier

Presented in this section are the results for the experiment where the effects of concept drift are expected to be less pronounced. The training data for each test set is the block of data that occurs immediately before it. In other words, for the test set D_3, the classifier is trained on the set D_2. The only results that change here are those for sets D_3 and D_4 since D_2 was already being tested with a classifier trained on the data in the previous block, D_1. The results for the retrained classifier on sets D_3 and D_4 are displayed in Tables 4.15 and 4.16 for accuracy and in Tables 4.17 and 4.18 for weighted ACC-COV. The difference between these scores and the corresponding results in Section 4.3.5 are then computed. This measure will evaluate the improvement for each score as a result of

Dataset	NE			RT		
	D_2	D_3	D_4	D_2	D_3	D_4
Hockey	95.3	100.0	63.4	21.9	35.6	2.2
Movies	68.2	59.2	60.5	79.2	82.6	61.2
20NG-5	34.3	84.3	97.8	72.1	82.7	64.8
Rt-earn	35.1	22.5	42.6	84.3	61.1	83.5
Rt-acq	46.9	43.8	58.9	99.7	100.0	100.0
lokay-m	83.5	80.3	76.3	21.9	27.3	30.5
farmer-d	49.6	53.7	64.6	46.5	42.9	46.8

Table 4.17: Percent Weighted ACC-COV for NB classifier if regularly retrained

Dataset	NE			RT		
	D_2	D_3	D_4	D_2	D_3	D_4
Hockey	92.5	100.0	93.3	97.8	97.0	66.9
Movies	48.4	52.8	48.5	83.2	88.1	81.0
20NG-5	86.9	92.4	97.1	94.1	92.8	100.0
Rt-earn	30.4	31.0	59.2	100.0	100.0	100.0
Rt-acq	38.1	52.3	66.1	97.0	98.6	100.0
lokay-m	58.9	47.2	68.5	32.7	33.4	80.1
farmer-d	37.0	47.2	45.5	20.1	64.4	59.4

Table 4.18: Percent Weighted ACC-COV for SVM classifier if regularly retrained

Measure	NE		RT	
	D_3	D_4	D_3	D_4
Accuracy	1.7	2.7	0.1	−1.3
Weighted	8.7	12.2	0.4	−2.6

Table 4.19: Difference in NB scores between retrained and non-retrained classifiers

Measure	NE		RT	
	D_3	D_4	D_3	D_4
Accuracy	2.1	4.3	0.6	0.2
Weighted	6.9	7.1	6.6	9.2

Table 4.20: Difference in SVM scores between retrained and non-retrained classifiers

retraining for each slice of data. Reported in Tables 4.19 and 4.20 are the average results for each set, for accuracy and weighted ACC-COV respectively. These tables specify the values obtained for both Naive Bayes and SVMs. The values in these tables serve to quantify the effect of retraining the classifier.

From these results, named entities seem to benefit the most from the retraining since they were the ones to suffer the most from the aging of the classifier. In addition, the SVM classifier also made the most use from the more up to date attribute list. The properties of the datasets did not change however. A named entity dependent dataset did not make any special use from the updated regular terms and the converse is also true. The main result here is therefore that named entity trained classifiers can be negatively impacted by time, but the degree to which this is true depends on the dataset and the nature of the data. The weighted ACC-COV scores for the SVM algorithm show, as they did in Section 4.3.5, that retraining the classifier only serves to improve the classification of lower certainty examples. This observation is supported by the fact that for SVMs there is not much difference in the average difference in weighted ACC-COV scores.

Dataset	NE				RT			
	D_1	D_2	D_3	D_4	D_1	D_2	D_3	D_4
Hockey	1.32	1.26	1.35	1.27	5.28	5.10	6.37	6.36
Movies	1.01	0.81	0.72	0.72	3.30	3.16	3.20	3.03
20NG-5	0.66	0.62	0.60	0.53	2.20	1.96	2.04	2.40
Rt-earn	0.49	0.44	0.46	0.42	1.65	1.66	1.70	1.47
Rt-acq	0.49	0.44	0.46	0.42	1.65	1.66	1.70	1.47
lokay-m	1.63	1.32	1.15	1.04	2.60	2.19	2.36	2.26
farmer-d	1.66	1.95	1.62	1.35	2.36	2.53	2.19	2.44

Table 4.21: Attribute density over time

Attribute Density

The results for attribute density are displayed in Table 4.21. Similar to the other results for time analysis, the values obtained are presented in four sections to show how the attribute density changes over time. The density being reported is a percentage of attribute occurrences over the total possible number occurrences. The interesting details about the results in Table 4.21 are twofold. The first is how much more frequent regular terms are than named entities. The second is that these results support the earlier results that named entity occurrences decay more noticeably than for regular terms. This fact is observed even for named entity dependent datasets. The fact that the attributes tend to disappear after a while supports the theory that it is concept drift and novelty introduction which are to blame for the decrease in classification accuracy. As a note, *rt-earn* and *rt-acq* have the same attribute density values because they used the same attributes. Only the class labels changed.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis presents several findings regarding the behaviour and role of named entities in text classification tasks. These findings can be broken down into three different areas.

1. Attributes for classification
2. Attribute representation
3. Behaviour of attributes in classification tasks

Each of these conclusions are presented in turn.

The first conclusion is that named entities are in fact useful in classification tasks. In such tasks, a general approach to the problem may not offer an appropriate solution based on the choice of classification algorithm and attribute representation. The datasets found to be named entity dependent in this thesis were the ones where a context was well-defined as part of the classification task. This restriction is needed because it establishes

the semantic background for the named entities. In other words, the tasks likely to lend themselves well to named entity classification are those found at the bottom nodes of a hierarchical classification tree. Another case where they are useful is within an organization where a clear context is present. To determine if named entities would be useful for classification, a characterization technique can be used to evaluate the ability of the classification algorithm to rank examples adequately. The best ranking technique among Naive Bayes, SVM and Decision Trees was found to be Naive Bayes since it correctly scores the hard-to-classify examples with low certainty and reflects the entire attribute list specified. After correctly characterizing a classification task, it may be appropriate to ignore a class of attributes to reduce the number of attributes. In a named entity dependent task, for example, it is beneficial on two levels to throw out regular terms: accuracy is improved and training and classification time is reduced as a result of the feature reduction. This result is in contrast to the findings by Cooley in [2] who reports named entities to not be useful, but does not acknowledge that his task was perhaps not dependent on them and that these attributes could be useful in other tasks.

The second conclusion relates to the representation of named entities. The work presented in this thesis shows that if named entities are useful, the method used to represent them can impact on classification performance. The degree of this impact depends on the nature of the named entities given the context of the classification task. If the entities are unique or unambiguous then t heir representation has little effect. If named entities overlap, however, then using multi-word attributes can help reduce this confusion.

The third and final conclusion of this thesis is that named entities, more so than regular terms, are dependent on the effects of time in classification. In other words, if

a classification task is dependent on named entities, then time must be considered more closely and it may be necessary to retrain the document classifier more frequently than for a regular term dependent task. Classifiers trained on named entities cannot be thought of as overfitting the data since there is no generalization possible for such terms. They are affected more due to the fact that they tend to come in and out of prominence over time. Hence, the attributes used for learning in the training set may not appear in the testing set. If new information appears in the testing set and it degrades performance significantly, then the classifier must be retrained to ensure a quality classification.

5.2 Future Work

The future work for this thesis involves using the conclusions presented above to create new techniques to classify texts where named entities are important.

The first would be to reduce the cost of dataset characterization. Named entity tagging is a costly endeavour, requiring several steps including parsing and part-of-speech tagging. It may be possible to develop a heuristic to extract likely named entities, or at least the most prominent ones, and use these to characterize the dataset. With such a heuristic, datasets could be tested for dependence on named entities at low computational cost.

The second possible future work would be to use the experimentation from Section 4.1 to develop a wrapper which would consider all three attribute types (named entities, regular terms, and a combined list) and return the one which produces the best results with Naive Bayes. The attribute list could then be used to conduct further classifier training and testing. Such an approach would suffer from being more computationally expensive than other wrapper approaches and would only be considered if the dataset is

hypothesized to be named entity dependent. If the first future work is successful than the knowledge of named entity dependence would be a lot cheaper to evaluate.

The third element of future work would be to study the possibility of using the characterization method to classify examples within a dataset as being named entity dependent or regular term dependent. The method presented in this thesis provides a characterization of the dataset as a whole, given the attributes specified. The real challenge with attempting to characterize individual examples is that any thresholds (assuming they would be needed) trained on earlier data would have to hold when applied on future data. The general problem here is in finding a method that will establish a relationship between past events and future events. With named entities in particular it is difficult to use the past to predict the future, since there is nothing governing when they will come into prominence.

The fourth possible future work is to develop a specialized named entity classification algorithm to be used in cases where named entities are important. Such an algorithm would take into consideration the properties of named entities as reported in this thesis. Namely, named entities tend to occur in bunches, coming in and out of prominence at unpredictable time intervals. The difficulty with the Naive Bayes algorithm when such a behaviour is observed is that it puts too much importance on when the attribute does not appear (especially if it was prominent in the training set). An improvement to this algorithm, when considering named entities, would be to ignore the conditional probability when an attribute is absent.

A final possible future work would be to investigate the characterization of a dataset not only on named entities but on the named entities subtypes. In other words, train separate classifiers using the names of people, organizations and locations. For this thesis,

this was not possible due to the fact that for named entity recognition this is still not done with high enough accuracy. The attribute types were therefore lumped together, and some semantic information was lost. If separated, a dataset could be said to be dependent on people names or location names for example.

Bibliography

[1] Yiming Yang and Jan O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Douglas H. Fisher, Ed., Nashville, US, 1997, pp. 412–420, Morgan Kaufmann Publishers, San Francisco, US.

[2] R. Cooley, "Classification of news stories using support vector machines," in *IJCAI '99 Workshop on Text Mining*, August 1999.

[3] Sam Scott and Stan Matwin, "Text classification using wordnet hypernyms," in *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, 1998.

[4] Satoshi Sekine, "Named entity: History and future," http://cs.nyu.edu/~sekine/papers/NEsurvey200402.pdf, 2004.

[5] Andrei Mikheev, Marc Moens, and Claire Grover, "Named entity recognition without gazetteers," in *EACL*, 1999, pp. 1–8.

[6] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: A framework and graphical development environment for robust NLP tools and applications," in

Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics, 2002.

[7] Ian H Witten and Eibe Frank, *Data mining: Practical machine learning tools and techniques with Java implementations*, Morgan Kaufmann, San Francisco, CA, 1999.

[8] Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, and Yoad Winter, "On feature distributional clustering for text categorization," in *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval.* 2001, pp. 146–153, ACM Press.

[9] David D. Lewis, "An evaluation of phrasal and clustered representations on a text categorization task," in *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval.* 1992, pp. 37–50, ACM Press.

[10] Ellen Riloff and Jeffrey Lorenzen, "Extraction-based text categorization: Generating domain-specific role relationships," in *Natural language information retrieval*, Tomek Strzalkowski, Ed., pp. 167–196. Kluwer Academic Publishers, Dordrecht, NL, 1999.

[11] W. Bruce Croft, Howard R. Turtle, and David D. Lewis, "The use of phrases and structured queries in information retrieval," in *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 1991, pp. 32–45, ACM Press.

[12] Chris Clifton and Robert Cooley, "Topcat: Data mining for topic identification in a text corpus," in *Principles of Data Mining and Knowledge Discovery*, 1999, pp. 174–183.

[13] Inxight Corp., "Inxight smartdiscovery: Discover the true value of information," http://www.inxight.com/pdfs/SmartDiscovery_white_paper.pdf, 2003.

[14] Ron Bekkerman, Andrew McCallum, and Gary Huang, "Automatic categorization of email into folders: Benchmark experiments on Enron and SRI corpora," Technical Report IR-418, University of Massachusetts, 2004.

[15] Jin Huang and Charles X. Ling, "Using AUC and Accuracy in Evaluating Learning Algorithms.," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 3, pp. 299–310, 2005.

[16] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz, "A multiple resampling method for learning from imbalances data sets," *Computational Intelligence*, vol. 20, no. 1, pp. 18–36, February 2004.

[17] Franca Debole and Fabrizio Sebastiani, "An analysis of the relative hardness of Reuters-21578 subsets," *Journal of the American Society for Information Science and Technology*, 2005, Forthcoming.

[18] Nathalie Japkowicz, "The class imbalance problem: Significance and strategies," in *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000)*, 2000, vol. 1, pp. 111–117.

[19] Bianca Zadrozny and Charles Elkan, "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers," in *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning.* 2001, pp. 609–616, Morgan Kaufmann Publishers Inc.

[20] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: a library for support vector machines," http://www.csie.ntu.edu.tw/~cjlin/libsvm/, 2004.

[21] P. Domingos and F. Provost, "Well-trained PETs: Improving probability estimation trees," CDER Working Paper #00-04-IS, New York University, 2000.