

# Supervised Information Retrieval for Text and Images

by

Alexandros Kyriakides

Submitted to the Department of Electrical Engineering and Computer Science  
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2004

© Alexandros Kyriakides, MMIV. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly  
paper and electronic copies of this thesis and to grant others the right to do so.

Author .....  
Department of Electrical Engineering and Computer Science  
5 February, 2004

Certified by .....  
Tomaso Poggio  
Eugene McDermott Professor  
Thesis Supervisor

Accepted by .....  
Arthur C. Smith  
Chairman, Department Committee on Graduate Theses



# **Supervised Information Retrieval for Text and Images**

by

Alexandros Kyriakides

Submitted to the Department of Electrical Engineering and Computer Science  
on 5 February, 2004, in partial fulfillment of the  
requirements for the degree of  
Master of Engineering in Computer Science and Engineering

## **Abstract**

We present a novel approach to choosing an appropriate image for a news story. Our method uses the caption of the image to retrieve a suitable image. We have developed a word-extraction engine called WordEx. WordEx uses supervised learning to predict which words in the text of a news story are likely to be present in the caption of an appropriate image. The words extracted by WordEx are then used to retrieve the image from a collection of images. On average, the number of words extracted by WordEx is 10% of the original story text. Therefore, this word-extraction engine can also be applied to text documents for feature reduction.

Thesis Supervisor: Tomaso Poggio  
Title: Eugene McDermott Professor



## Acknowledgments

I would like to thank all the people who made the writing of this thesis possible. First of all, I would like to thank my thesis supervisor Tomaso Poggio for all his help and support for my research. Giorgos Zacharia has been by my side throughout my career at MIT, advising me at almost every step of the way, and helping me achieve my goals. For this I will be eternally grateful. I could not hope for a better academic advisor than Patrick Winston. His understanding and wisdom are greatly appreciated.

I would like to thank all my friends for being there for me when I needed them, and I hope I will always be able to do the same for them. Most importantly, I would like to thank my family. Their love and their constant, unconditional support have given me the strength to go on for all these years.



# Contents

<b>1</b>	<b>Introduction</b>	<b>15</b>
1.1	Motivation . . . . .	15
1.2	Vision . . . . .	16
1.3	Background . . . . .	16
<b>2</b>	<b>Overview</b>	<b>19</b>
2.1	The Problem . . . . .	19
2.2	Data . . . . .	21
2.2.1	Details . . . . .	21
<b>3</b>	<b>Approach</b>	<b>23</b>
3.1	Simple Classification . . . . .	23
3.2	Simple Document Retrieval . . . . .	24
3.3	A Novel Approach . . . . .	25
3.3.1	Classification . . . . .	25
3.3.2	Document Retrieval . . . . .	26
3.3.3	Feature selection . . . . .	26
<b>4</b>	<b>WordEx</b>	<b>27</b>
4.1	Creating WordEx . . . . .	27
4.2	Using WordEx . . . . .	28
4.3	The feature vector . . . . .	32

4.3.1	Term Frequency Inverse Document Frequency (TFIDF) . . . . .	33
4.3.2	Information Gain (IG) . . . . .	35
4.4	Theory and Tools . . . . .	36
4.4.1	Supervised Learning . . . . .	36
4.4.2	Decision Trees . . . . .	37
4.4.3	Boosting . . . . .	38
<b>5</b>	<b>Experiments</b>	<b>39</b>
5.1	Simple Classification Experiment . . . . .	39
5.2	Simple Document Retrieval . . . . .	40
5.2.1	Experiment 1 - small data set . . . . .	41
5.2.2	Experiment 2 - large data set . . . . .	41
5.3	Experiments using WordEx . . . . .	41
5.3.1	Evaluation . . . . .	43
5.3.2	Summarization . . . . .	44
<b>6</b>	<b>Results</b>	<b>47</b>
6.1	Simple Classification . . . . .	47
6.2	Simple Document Retrieval . . . . .	48
6.3	Using our Approach - WordEx . . . . .	49
6.4	Human Evaluation . . . . .	51
6.5	Summarization . . . . .	53
<b>7</b>	<b>Discussion</b>	<b>55</b>
7.1	Conclusions . . . . .	55
7.2	Further Improvements . . . . .	56
<b>8</b>	<b>Contributions</b>	<b>59</b>
<b>A</b>	<b>Detailed Results</b>	<b>61</b>







# List of Figures

2-1	An example news story clearly showing the story text, image, and caption text	20
4-1	The WordEx engine . . . . .	27
4-2	The WordEx web interface . . . . .	30
4-3	WordEx output . . . . .	31
4-4	A sample of words from the corpus used for the TFIDF calculation . . . . .	35
4-5	The thirty words with highest Information Gain . . . . .	37
5-1	An example story with an image for evaluation by a person . . . . .	46
A-1	Simple Naive Bayes . . . . .	62
A-2	Simple Naive Bayes with AdaBoost . . . . .	62
A-3	3 Nearest Neighbors . . . . .	63
A-4	3 Nearest Neighbors with AdaBoost . . . . .	63
A-5	5 Nearest Neighbors . . . . .	64
A-6	5 Nearest Neighbors with AdaBoost . . . . .	64
A-7	7 Nearest Neighbors . . . . .	65
A-8	7 Nearest Neighbors with AdaBoost . . . . .	65
A-9	C4.5 Decision Trees . . . . .	66
A-10	C4.5 Decision Trees with AdaBoost . . . . .	66
A-11	Linear SVM . . . . .	67
A-12	Linear SVM with AdaBoost . . . . .	67

B-1	News Story A . . . . .	70
B-2	Summary of News Story A . . . . .	70
B-3	News Story B . . . . .	71
B-4	Summary of News Story B . . . . .	71
B-5	News Story C . . . . .	72
B-6	Summary of News Story C . . . . .	72

# List of Tables

4.1	The training set used to train WordEx . . . . .	28
5.1	Simple Classification Experiment . . . . .	40
5.2	Simple Document Retrieval Experiment 1 . . . . .	41
5.3	Simple Document Retrieval Experiment 2 . . . . .	41
5.4	Experiments using WordEx . . . . .	42
6.1	Results of Simple Classification experiment . . . . .	47
6.2	Results of simple document retrieval using the full story text as query . . .	48
6.3	Results from different classifiers . . . . .	49
6.4	Number of correct captions retrieved using the full story, the predicted yes- words and the actual yes-words as the query . . . . .	50
6.5	Results of Human Evaluation . . . . .	51
6.6	Results of Z-test . . . . .	52
6.7	Interesting results obtained from ranking sentences based on yes-words . . .	53



# Chapter 1

## Introduction

### 1.1 Motivation

In recent years, with the continuing increase in available storage space, data repositories have grown to such sizes that it is becoming ever more difficult to keep track of this large amount of data. More importantly, it is becoming quite a task to retrieve the information that one needs. A good example of a huge data repository is the Internet itself. We are lucky to have search engines such as Google<sup>1</sup>, that aid us in our task of finding the information we need on the World Wide Web(WWW).

Finding a picture or photograph from a large database is also important. It is often desirable to accompany a piece of text with an appropriate picture or photograph. This usually proves quite time-consuming because it is a manual process. A person needs to browse through many pictures in order to find the one he wants due to the large number of pictures in the database. Organized databases can make this easier of course. For example, a database with pictures labeled by category, can significantly reduce the time someone needs to find the picture. The cost however is shifted to the categorization task. If the categorization is done manually, then it takes time to place the pictures in appropriate categories. It would therefore be desirable to have an automatic process, that can retrieve an

---

<sup>1</sup><http://www.google.com>

appropriate picture or photograph from a large database, with minimal human intervention.

An application where this could be very useful is for on-line news sites, like CNN<sup>2</sup> or Yahoo! News<sup>3</sup>. Once a journalist writes a news story, the next step is to find an appropriate photograph that will appear next to the news story on the web page. If this could be done automatically, it would save a great deal of time.

## 1.2 Vision

Our thesis is that we can devise machine learning algorithms that can automatically find an appropriate picture or photograph for a news story.

This can be modeled as an information retrieval problem. In the basic information retrieval problem, we have a data set and we wish to select a portion of this data set that is relevant to our needs. This is done by using a *query*. The retrieval engine outputs the results that satisfy the query. A good example is a search engine on the WWW, such as Google. The data set is the collection of web pages on the Internet. The search engine accepts a query as input. The query is made up of the search terms. The result is a list of web pages (URL's) that satisfy the query. The accuracy of the results can be evaluated in terms of precision and recall [17]. Precision is defined as the number of correct results returned, divided by the total number of results returned. Recall is defined as the number of correct results returned, divided by the total number of correct results present in the whole data set.

In our case, the query is the news story, and our results will be a list of pictures or photographs.

## 1.3 Background

As stated in [12], Image Retrieval has been a very active research area since the 1970's. Image Retrieval has been studied from two different angles: text-based and visual-based.

---

<sup>2</sup><http://www.cnn.com>

<sup>3</sup><http://news.yahoo.com>



The fields of text mining, boosting and combination of learning machines are of particular interest to us. Text mining describes the technique of extracting useful data from relatively unstructured collections of text. Boosting is a general method which attempts to "boost" the accuracy of any given learning algorithm. [15] . Schapire [14] came up with the first provable polynomial-time boosting algorithm in 1989. Algorithms have also been developed that make decisions by combining the predictions of several prediction strategies, called experts. [2] Some algorithms also use the added feature of allowing experts to abstain from making a prediction. Such experts are called specialists, because we think of them as making their prediction only when the instance to be predicted falls within their area of expertise [5]. Multi-label text classification methods have been studied and analyzed by McCallum [7] and by Imai et al. [6]. Huang et al. [5] state that not much work has been done on automatic image classification, and propose a method for hierarchical classification of images via supervised learning.



# Chapter 2

## Overview

### 2.1 The Problem

The problem we will be addressing in this thesis is that of finding an appropriate picture to match a news story. On-line news sites usually accompany their news stories with an appropriate picture or photograph. To keep the terminology consistent, the name we will use for these pictures or photographs is *images*. On most news sites, these images also have a caption that describes the image.

Therefore, a **news story** that appears on an on-line news site consists of three separate parts:

1. **The story text** - the news story
2. **The image** - picture, photograph, etc.
3. **The caption text** - the caption describing the image

Figure 2-1 on page 20 shows an example news story with the three parts labeled.

More clearly stated therefore, the problem is to find an appropriate **image** to match the **story text**.


 <p><b>IMAGE</b></p>	<p><b>STORY TEXT</b></p> <p><b>Annan Sees All Sides Ready to Resume Cyprus Talks</b></p> <p>WASHINGTON (Reuters) - U.N. Secretary-General Kofi Annan said on Tuesday that all sides in the Cyprus dispute appeared ready to resume negotiations on a plan for reuniting the divided Mediterranean island.</p> <p>Annan, in Washington for talks with President Bush, said he had asked Turkish Prime Minister Tayyip Erdogan, Greek Cypriot President Tassos Papadopoulos and Greek Foreign Minister George Papandreou about a resumption of the U.N-backed talks to reunite Cyprus ahead of its accession to the European Union on May 1.</p> <p>However, Annan said he had not yet spoken to the Turkish Cypriot leader, Rauf Denktash, "but everyone seems ready to resume and I hope to be able to invite them to a meeting shortly," he told reporters at the White House.</p> <p>"President Bush supports my efforts and would like to see the talks resume, and supports the plan we've put on the table and urges the parties to press ahead and negotiate and find a settlement on the basis of that," Annan said.</p>
<p>United Nations Secretary-General Kofi Annan speaks to the press outside the West Wing of the White House, February 3, 2004 after meeting with President George W. Bush. U.N. Annan said that all sides in the Cyprus dispute appeared ready to resume negotiations on a plan for reuniting the divided Mediterranean island.</p> <p><b>CAPTION TEXT</b></p>	

Figure 2-1: An example news story clearly showing the story text, image, and caption text

In this thesis we will only use a text-based approach. We will not perform any image processing. The only data we will use to solve the stated problem, is the **caption text** and the **story text**. We will try to predict if an image is appropriate for a certain story, by using the caption of the image, and not the image itself.

## 2.2 Data

The data we used for our experiments was taken from the Yahoo! News<sup>1</sup> website. We found the data from this site appropriate because the captions of the images were significantly larger than those of other on-line news sites.

We collected news stories with their images and captions within the time period from November 2001 until April 2003. This is a period of 18 months. During this time we collected a total of 35766 news stories. For all of these, we stored the story text and the caption text. In order to conserve disk space, we only stored the images of 3695 of these stories, because the actual images were not needed for the actual construction of the machine learning algorithms. The images were only used for evaluation purposes.

### 2.2.1 Details

On average, each story text has about 565 words, with stories ranging from as low as 20 words, to as high as 2000 words. Each story has a title. On average, each title has about 7 words, with titles ranging from 1 to 16 words.

On average, each caption has about 50 words, with captions ranging from 10 to 140 words. Each news story usually appears on-line with one image. Sometimes, however there can be two images. Each image has its own caption. We say these captions are *associated* with the news story, because the images they describe appeared on-line as part of the news story.

---

<sup>1</sup><http://news.yahoo.com>



## Chapter 3

# Approach

There are a number of approaches that we can use to solve the problem we introduced.

In this chapter we will describe three approaches: Simple Classification, Simple Indexing, and a novel approach that we conceived for the purposes of this thesis.

### 3.1 Simple Classification

In a simple classification task, we have a set of classes, and a classification algorithm. Given a test instance, this algorithm tries to predict into which class it falls into. For example, say we have three classes of news stories: Business, Entertainment, and Technology. Given a news story, the classification algorithm will try to predict into which of these classes the news story falls into. In order to perform this task, the algorithm needs to be trained so that it can learn how to classify a news story into one of the three categories. The algorithm is trained using a *training set*. This training set consists of pre-classified news stories. For example, as a training set we can have: 100 Business news stories, 100 Entertainment news stories, and 100 Technology news stories. The algorithm uses the training set to build a model. The algorithm then uses the model to make future predictions.

For the purposes of this thesis, we can define each class to be a news story. We will therefore be performing multi-class classification. The *training set* will be the caption

text associated with each news story. For example, say our training set consisted of 1200 captions. And say these captions were obtained from 1000 different news stories. Then we will have 1000 different classes. (There are more captions than stories because some news stories have more than one caption associated with them.) The learning algorithm then builds a prediction model based on the captions of each news story. Given a story text, it then predicts into which class the story text falls into. Each of the classes is associated with one or more images, because each of the captions used in the training set describes an image. Consequently, the images that match a given story text are the images whose captions have the same class as the predicted class of the story text.

### 3.2 Simple Document Retrieval

In a simple document retrieval task, we have a set of documents in a database, and we wish to retrieve the correct document from this database, given a request. The request is called the *query* and it consists of a text string. For example, say our document database consists of 1000 news stories, and we want to retrieve one of these news stories. The first step is to create an index from these documents (similar to the index at the back of a book). Then, given a query, we retrieve all the documents that match the query. If the query is “United Nations”, for example, then all the documents having the words “united” and “nations” will be retrieved. Say, for instance, that out of the 1000 news stories in our database, 14 of these news stories contained the words “united” and “nations”. These 14 documents will be returned as the result of the query. Then a ranking scheme can be used to rank these 14 documents. For example, a document that has the words “united” and “nations” appearing 5 times might be more relevant than a document that has these words appearing once. We can also treat “united nations” as as a single *token* instead of breaking it up into two single-word *tokens*. This will have the result of returning the documents that have the words “united nations” appearing next to each other. This is probably desirable in the case of “United Nations”, but most likely it is not desirable in a case where the query is something like “countries nations”.



We can use a document retrieval method to try and solve the problem presented in this thesis. Given a database of images, we take their captions and create an index. We then use the story text as the *query*. We try a simple approach where each token is a single word. The results of the query will be a set of captions. We rank the returned results based on a document similarity measure. The top-ranking caption returned will be the one that is most similar to the query (the story text). To perform the indexing, retrieval, and document similarity ranking we used the program `arrow`[8].

### 3.3 A Novel Approach

Here we propose a way to solve the problem we presented by using both a classification part, and a document retrieval part. The first step involves classification, and the second step involves document retrieval.

#### 3.3.1 Classification

The classification we proposed in section 3.1 is a multi-class classification problem. In general, the running time of classification algorithms increases with the number of classes. It is desirable therefore to keep the number of classes small. With this new approach, we will perform binary classification. This means there will only be two classes, thus reducing the running time of the classification task.

We will try to predict which words in the story text are most likely to be present in the caption of an image that is appropriate for the news story.

Let’s take a generic example. A news story consists of the story text, an image, and the caption describing the image. The story text has 500 words. The caption text has 50 words. The 500 words of the story text can be placed into two classes. One class is “the words that *do not* appear in the caption”, and the other class is “the words that *do* appear in the caption”. It is quite obvious that if we take each of the words in the story text one-by-one, we will see that each word falls in exactly one of these two classes. For convenience, we will

name these two classes as **yes** and **no**. The words that fall in the **yes** class, will be called “yes-words” and the words that fall in the **no** class will be called “no-words”.

- Class **yes**: the words in the story text that also appear in the caption text (we call these the “yes-words”)
- Class **no**: the words in the story text that do not appear in the caption text (we call these the “no-words”)

### 3.3.2 Document Retrieval

This step is very similar to the method described in section 3.2. In 3.2 we used the complete story text as the query. Here we will use only the yes-words as the query.

In an ideal scenario, we will be able to correctly classify all the words in the story text into yes-words and no-words. The no-words are of no importance to us when trying to retrieve the caption that matches the story text. The no-words could actually decrease our chances of finding the correct caption, because they will retrieve the wrong captions. By using only the yes-words as the query, we can increase our chances of retrieving the correct caption.

### 3.3.3 Feature selection

We can say that our classification step “weeds out” the no-words from the news story so that it increases accuracy of the document retrieval step. Each word in the story text, is a *feature*. Some of these features (the no-words) need to be removed because in the document retrieval step, they have the ability to select the wrong captions. We need to extract only the important features from the story text (the yes-words), that will help us retrieve the correct caption. The process of removing the no-words and keeping the yes-words is called **feature selection**.

## Chapter 4

# WordEx

WordEx is a word-extraction engine. Given a news story text, WordEx extracts the yes-words from the story. We have defined “yes-words” in section 3.3.1.



Figure 4-1: The WordEx engine

WordEx is a classification algorithm: it classifies *instances* into *classes*. Each word that is given to WordEx as input, is an *instance*. Each instance is then classified into one of two classes: **yes** or **no**. See page 26 for a description of the classes.

### 4.1 Creating WordEx

WordEx uses a machine-learning model to perform the classification. Like most machine-learning models, it is created using a training set. Our training set consisted of 9394 instances. Each instance is described by a feature vector. The current version of WordEx

uses nine features. Each instance is a word, and therefore each of these features is a feature that describes the word. For example, “the number of times the word appears in the story text” is a feature. The feature vector is described in detail in section 4.3.

The 9394 training instances were obtained from a set of 938 news stories. These news stories were taken randomly from our complete data set mentioned in section 2.2. Therefore, the stories spanned the whole time period from November 2001 to April 2003. A restriction was placed on choosing these stories: each story text had 100 or more words.

From these 938 news stories, we only used the **story text** and the **caption text**. There were 938 story texts, and 1138 caption texts. There were more captions than story texts because some news stories had more than one image, and therefore more than one caption. Each word in a story text is an *instance*, that can be used for training. All the story texts have a total of 187882 words. Each word can be classified as a yes-word or no-word, depending on whether it also appears in one of the captions associated with the story text. From these 187882 words, 15224 of them were yes-words, and 172658 of them were no-words. We can conclude that on average 8% of the words in a story text are yes-words, and 92% of them are no-words.<sup>1</sup>

To train WordEx, we did not use all of the 187882 instances. We used a 5% balanced sample. As can be seen in Table 4.1, only 9394 instances were used, with a roughly equal number of yes and no words.

instances of class <b>yes</b>	4711
instances of class <b>no</b>	4683
Total number of training instances	9394

Table 4.1: The training set used to train WordEx

## 4.2 Using WordEx

Using WordEx is very simple. The input is simply a news story text. The input is separated into two distinct parts: the title of the news story, and the body of the news story. We

---

<sup>1</sup>This is after the removal of stop words.

developed both a command-line interface to WordEx, as well as an on-line web interface. The command-line version accepts one argument as input: the directory where to find the two files, `title.txt` and `story.txt`, that contain the title and body of the news story, respectively.

*Example:*

```
wordex.sh /data/articles/article2
```

The above example shows how to use Wordex on the UNIX command line. The directory `/data/articles/article2` should contain the files `title.txt` and `story.txt`.

At the time of this writing, the web interface was located at <http://spada.mit.edu/cgi-bin/WordEx.pl>. Figure 4-2 on page 30 shows the on-line interface with example text as input. Figure 4-3 on page 31 shows the output, after the “Extract Words” button was clicked.

# WordEx

This is a word extraction engine developed as part of my MEng thesis.  
It's purpose is to take as input a news article (consisting of a Title and Text).  
It then extracts the *important words* from the news article.  
An *important word* is defined as one which is predicted to be in  
the caption of a picture that matches the news story.  
Therefore, WordEx is used to find pictures that match news stories.

It's output can be used for other purposes as well, however.  
It uses a machine learning model to extract the words.

NOTE: When entering text in the form below, keep in mind that capitalization makes a difference  
to the way words are processed.  
For example, a capitalized word might be considered more important than a non-capitalized one.

For comments, questions, or suggestions, please send me email at: [alex1@mit.edu](mailto:alex1@mit.edu)

## Title:

Prime Minister Blair urged to admit Iraq mistake

## Text:

LONDON, Jan. 30 (Xinhuanet) -- Britain's Ex-Foreign Secretary Robin Cook has urged Prime Minister Tony Blair to admit that intelligence he presented to parliament on Iraq was is was "wildlywrong," the British Broadcasting Corporation (BBC) reported Friday.

"It's getting embarrassing to watch our government still tryingto deny reality," said Robin Cook, who resigned from the UK cabinet as leader of the House of Commons last year in protest against Blair's policy supporting a war against Iraq.

He said the prime minister should admit that although he believed information on supposed weapons of mass destruction "in all good faith," it was incorrect.

Click here to extract the important words:

Extract Words

Figure 4-2: The WordEx web interface

Please wait a few seconds...

**Title:**

Prime Minister Blair urged to admit Iraq mistake

**Text:**

LONDON, Jan. 30 (Xinhuanet) -- Britain's Ex-Foreign Secretary Robin Cook has urged Prime Minister Tony Blair to admit that intelligence he presented to parliament on Iraq was is was "wildlywrong," the British Broadcasting Corporation (BBC) reported Friday. "It's getting embarrassing to watch our government still tryingto deny reality," said Robin Cook, who resigned from the UK cabinet as leader of the House of Commons last year in protest against Blair's policy supporting a war against Iraq. He said the prime minister should admit that although he believed information on supposed weapons of mass destruction "in all good faith," it was incorrect.

**Extracted words:**

iraq minister cook prime blair robin foreign jan reported war supporting reality friday broadcasting british secretary tony house embarrassing admit resigned mistake london presented parliament commons corporation britain uk cabinet leader urged

[Back to WordEx](#)

Figure 4-3: WordEx output

### 4.3 The feature vector

The story text given to WordEx as input is split up into *tokens*. Each token is a single word. If a word appears more than once in the text, only a single token is created for that word. A list of 524 stop words is used to remove common words that provide little or no information.<sup>2</sup> For example, words like “and”, “if”, and “or” are in the list of stop words. No stemming is performed in this version of WordEx. After the stop words are removed, a **feature vector** is created for each of the remaining words. The following list describes the nine features present in the feature vector of each token.

1. The number of times the word appears in the whole story text.<sup>5</sup>
2. The number of times the word appears in the title of the story text.<sup>3</sup>
3. The number of times the word appears capitalized in the body of the story text.<sup>4</sup>
4. The number of times the word appears capitalized in the title of the story.<sup>3</sup>
5. The part of speech of the first occurrence of the word. (Noun, Verb, etc.)<sup>6</sup>
6. The position of the first appearance of the word in the whole story text.<sup>5</sup>
7. The length of the word in characters.
8. The TFIDF score of the word. (See Section 4.3.1).
9. The Information Gain score of the word (See Section 4.3.2)

---

<sup>2</sup>The list of stop words is the default list provided by the program `rainbow`[8]

<sup>3</sup>For the command-line interface, the title of the story text is the file `title.txt`. For the web interface, it is the top input box, marked “Title”.

<sup>4</sup>For the command-line interface, the body of the story text is the file `story.txt`. For the web interface, it is the lower input box, marked “Text”.

<sup>5</sup>The whole story text consists of the concatenation of the title and body of the story text, with the title placed first.

<sup>6</sup>An implementation of Eric Brill’s rule-based part-of-speech tagger[1] was used.



Here is a feature vector obtained from the example given on page 30. The features are separated by commas. This is the feature vector created for the word “minister”.

3,1,1,1,NNP,2,8,11.669298,0.17626

The first feature has the value of 3, and it indicates that the word “minister” appeared three times in the whole story text (once in the title, and twice in the body of the story). The second feature has the value of 1. This indicates that the word “minister” appears once in the title of the story. The third feature has the value of 1, because the word “minister” appears capitalized once in the body of the story text. The fourth feature also has the value of 1, because the word “minister” appears capitalized once in the title of the story.

The fifth feature, indicates the part of speech. The value *NNP* means that the part-of-speech tagger[1], recognized this word as a proper noun.

The sixth feature has the value of 2. This is the position where the word “minister” appeared for the first time in the whole story text (title included). The word “minister” has eight characters (letters). This is indicated by the value 8 in the vector.

The last two features are the TFIDF and Information Gain scores.

#### 4.3.1 Term Frequency Inverse Document Frequency (TFIDF)

A common scheme for calculating a numerical score for words in text documents is called term-frequency inverse-document-frequency (TFIDF)[13]. This score is calculated based on a corpus  $C$  of text documents, using this formula:

$$TFIDF(w, d) = TF(w, d) \times \ln \left( \frac{n}{DF(w)} \right)$$

$TFIDF(w, d)$  :The TFIDF score of word  $w$  in document  $d$ .

$TF(w, d)$  :The Term Frequency of word  $w$  in document  $d$ .

$n$  :The number of documents in the corpus  $C$ .

$DF(w)$  :The Document Frequency.

The Term Frequency of word  $w$  in document  $d$  is the number of times the word  $w$  appears in document  $d$ . The Document Frequency is the number of documents in corpus  $C$  that contain the word  $w$ .

It can be seen that the more documents a words appears in ( $DF(w)$  high), the lower the TFIDF score. In general, when using TFIDF scores, a low score indicates a word that is not important. The idea is that if a word appears in many documents, then it is not helpful in distinguishing one document from another. In the case of WordEx however, a low TFIDF score could indicate an important word as much as a high TFIDF score does. This is because WordEx uses decision trees in order to predict which words are important. See Section 4.4.2.

In the current version of WordEx the corpus  $C$ , contains 427429 text documents. These are news stories and image captions retrieved from various on-line news sites. The sole purpose of this corpus is so it can be used to calculate TFIDF scores for words. Therefore, we do not need to store all the documents in the corpus. We only need to know the total number of documents in the corpus, and the document frequency of each word that appears in the corpus. The document frequency of a word is the number of documents that contain the word. We have stored this information in a MySQL database. Figure 4-4 shows the 30 words in the corpus with the highest frequency. From this table, only the document frequency (**df**) field is used for the TFIDF calculation.

The procedure for calculating the TFIDF score is rather simple. The story text is given to WordEx as input. WordEx tokenizes the text, each token being a word. Each word has a frequency (the number of times it appeared in the text). This frequency is  $TF(w, d)$  in the TFIDF formula above. The value of  $n$  is 427430. This is the number of documents in the corpus, plus one for the story text being given as input. The value for  $DF(w)$  is the value taken from the database, plus one. Using these three values, we can now calculate  $TFIDF(w, d)$ .

```
mysql> select word,frequency,df,last_update from wordtable order by frequency desc limit 30;
```

word	frequency	df	last_update
news	73762	24824	2003-11-13 05:20:42
sites	47798	19344	2003-11-13 05:20:42
web	47549	19419	2003-11-13 05:20:41
iraq	32282	10527	2003-11-13 05:20:42
war	30875	12368	2003-11-13 05:20:42
people	29128	13551	2003-11-13 05:20:41
year	26968	13792	2003-11-13 05:20:43
president	26580	14500	2003-11-13 05:20:43
united	26251	11656	2003-11-13 05:20:42
bush	25251	8767	2003-11-13 05:20:43
states	21770	10334	2003-11-13 05:20:41
israeli	21436	4738	2003-11-13 05:20:43
palestinian	20580	4247	2003-11-13 05:20:42
time	20357	12246	2003-11-13 05:20:42
military	20323	9253	2003-11-13 05:20:42
told	19639	11525	2003-11-13 05:20:42
officials	19413	10033	2003-11-13 05:20:41
israel	18684	4177	2003-11-13 05:20:42
city	17638	8898	2003-11-13 05:20:42
state	17506	9897	2003-11-13 05:20:41
government	17430	8526	2003-11-13 05:20:42
percent	17343	4963	2003-11-13 05:20:43
washington	17089	9928	2003-11-13 05:20:42
game	16742	5486	2003-11-13 05:20:43
world	16257	9571	2003-11-13 05:20:42
minister	16071	8740	2003-11-13 05:20:41
security	15928	7870	2003-11-13 05:20:42
friday	15773	9842	2003-11-13 05:20:41
tuesday	15767	10004	2003-11-13 05:20:43
reuters	15746	11146	2003-11-13 05:20:44

```
30 rows in set (0.05 sec)

mysql>
```

Figure 4-4: A sample of words from the corpus used for the TFIDF calculation

### 4.3.2 Information Gain (IG)

When doing class prediction, information gain[9] measures the number of bits of information obtained, given that a certain word is in a document or not. To train WordEx, information gain is calculated by using the 938 news stories described in Section 4.1. Each story is considered as being in a class of its own. Therefore, there are 938 classes. When using WordEx to extract the yes-words of a new story text, we add the new story text in another class of its own, and therefore there are 939 classes when calculating the information gain of each word in the story text. The formula for calculating information gain of a word  $w$  is based on probabilities.

$$\begin{aligned}
 IG(w) = & -\sum_{i=1}^m P(c_i) \log P(c_i) \\
 & +P(w) \sum_{i=1}^m P(c_i|w) \log P(c_i|w) \\
 & +P(\bar{w}) \sum_{i=1}^m P(c_i|\bar{w}) \log P(c_i|\bar{w})
 \end{aligned}$$

$IG(w)$  :The Information Gain of word  $w$ .  
 $m$  :The number of classes.  
 $P(w)$  :The probability of word  $w$ .  
 $P(\bar{w})$  :The probability of some other word than  $w$ .  
 $P(c_i)$  :The probability of class  $c_i$ .  
 $P(c_i|w)$  :The probability of class  $c_i$ , if word  $w$  is in the document (story text).  
 $P(c_i|\bar{w})$  :The probability of class  $c_i$ , if word  $w$  is not in the document (story text).

When training WordEx,  $m$  takes the value of 938. When using WordEx to predict the yes-words of a new story text,  $m$  has the value of 939. Figure 4-5 shows the thirty words with the highest information gain obtained from the example given on page 30. Note that some of the words in the list are not in the story text of the example. WordEx creates the list of information gain scores, and then picks out the scores for the words it needs.

## 4.4 Theory and Tools

WordEx uses supervised learning, decision trees and boosting. The algorithms we used are found in the Weka[16] machine-learning software. Weka is a project of the Department of Computer Science of The University of Waikato.

### 4.4.1 Supervised Learning

WordEx uses *supervised learning* to perform its function. In supervised learning, the classes are established beforehand. In our case, there are two classes: *yes* and *no*. Once the classes are established, the learning algorithm is given a sample of pre-classified training instances.

```
# bin/rainbow --print-word-infogain=30 -d models/data_test/ | nl
Loading data files...
Placed remaining 3016 documents in the train set:
Calculating info gain... words ::          56

1  0.30087 iraq
2  0.22633 war
3  0.21435 sites
4  0.21210 web
5  0.20657 bush
6  0.20130 president
7  0.19863 iraqi
8  0.19329 news
9  0.18886 york
10 0.18498 baghdad
11 0.18166 monday
12 0.18074 friday
13 0.17836 sunday
14 0.17626 minister
15 0.17224 world
16 0.17196 april
17 0.17039 tuesday
18 0.16791 thursday
19 0.16740 city
20 0.16675 wednesday
21 0.16532 military
22 0.16474 killed
23 0.16469 united
24 0.16451 state
25 0.16423 people
26 0.16315 forces
27 0.16315 washington
28 0.16149 saturday
29 0.15802 west
30 0.15759 states
```

Figure 4-5: The thirty words with highest Information Gain

For supervised learning, it is therefore required that we have example instances whose classes we know a-priori. We obtained these examples from news stories that appeared on on-line news sites.

#### 4.4.2 Decision Trees

The Weka machine-learning package includes an implementation of C4.5 decision tress as described by Quinlan[11]. To use a decision tree, one starts at the root of the tree, and moves through each node, until a leaf is reached. Each node is a *decision node* that specifies a test. The result of a test indicates which branch to follow. For example, the decision node could say: “If the feature *length of word* is greater than 8, take the branch on the left, otherwise take the branch on the right”. At each node therefore, a simple test is performed. Each leaf specifies a class, indicating the result of the classification.

Weka’s version of C4.5 is named J48. There are two important parameters that need to be specified for the J48 algorithm. The first parameter is *the minimum number of instances per leaf*. This is the minimum number of instances that must be present in the

training data for a new leaf to be created in the decision tree to handle those particular instances. In the current version of WordEx, we used the value 2. So when the decision tree was being created, if there was only one training instance which was not accounted for by the current tree, then a new leaf would not be created to handle that particular instance. The second parameter is *confidence value to be used when pruning the tree*. Pruning is performed to remove branches that are not significantly important for the tree to perform the classification. They are branches that provide little or no contribution to the accuracy of the model. A higher confidence value, decreases the amount of pruning that takes place. In the current version of WordEx we used a value of 25% for this parameter.

### 4.4.3 Boosting

Boosting[15] is used to improve the performance of a learning algorithm. In WordEx for example, a boosting algorithm called **AdaBoost**[4] is used to improve the performance of the C4.5 learning algorithm.<sup>3</sup> Boosting runs several iterations of the learning algorithm, and then combines the models into one single classifier. WordEx runs 10 iterations of AdaBoost.

---

<sup>3</sup>There are two versions of AdaBoost: **AdaBoost.M1** and **AdaBoost.M2**. For binary classification, as is the case in WordEx, both versions behave the same. See [4] for details.

## Chapter 5

# Experiments

In this chapter we present three types of experiments. They are based on the three approaches described in Chapter 3. The results of the experiments are collected together in Chapter 6 to make it easier for the reader to compare them.

### 5.1 Simple Classification Experiment

For this experiment we used a data set of 938 news stories. These news stories were taken randomly from our complete data set mentioned in section 2.2. Therefore, the stories spanned the whole time period from November 2001 to April 2003. Each story text in this set has more than 100 words. The data set contains a total of 2076 text files, each one under one of 938 directories. Each directory represents a class. Each news story is considered to be a class of its own, and therefore has its own directory. Each directory has a one `story.txt` file. This is the story text. Each directory also has one or two caption files. The first caption is always named `caption_1.txt`, and if a second caption exists, it is named `caption_2.txt`. In total, there are 1138 captions in this data set.

The files were saved in this manner, because of the requirements of the classification program (`rainbow`[8]) used for this experiment. It requires that documents belonging to the same class should be placed in the same directory.

The classification experiment is a two step process. The first step trains the classification model using a training set. The second step tests the model, using a test set. The captions were used as the training set, and the story texts as the test set. This is summarized in Table 5.1. We used three different types of classification algorithms: Naive Bayes, Nearest Neighbors, and Support Vector Machines.

Total number of news stories used in the experiment	938
Total number of classes	938
Total number of captions ( <i>training set</i> )	1138
Total number of story texts ( <i>test set</i> )	938

Table 5.1: Simple Classification Experiment

For each of the 938 test cases in the test set we recorded the top 5 classes returned as a result of the classification, to see which ones were correct. A correct result is one that agrees with the actual caption that appeared on the on-line news site with that news story.

## 5.2 Simple Document Retrieval

We performed two experiments. One experiment was with the same data set used for the Simple Classification experiment. See Section 5.1 for a description of the data set. This was so we could compare the indexing and retrieval methods. The second experiment used a larger data set. Simple Document Retrieval is much faster than our Simple Classification method, and that is why we could apply it to a larger data set. The experiment is done in two steps. First, the captions are used to create an index for the news stories. The second step is to query the index, to retrieve the closest match to the query. In these experiments the full story text was used as the query. The program `arrow`[8] was used to perform the indexing and retrieval<sup>1</sup>.

---

<sup>1</sup>`arrow` performs simple TFIDF-based retrieval



### 5.2.1 Experiment 1 - small data set

For this experiment we used 938 news stories. The 1138 captions in the data set were used in the indexing step, and then each of the 1138 story texts were used as the query. The top 5 results were recorded for each of the 938 queries performed, to see which ones were correct. The correct caption is the one that appeared on the news site with that news story. The following table summarizes the experiment.

Total number of news stories used in the experiment	938
Total number of captions ( <i>used to create the index</i> )	1138
Total number of story texts ( <i>used as the query</i> )	938

Table 5.2: Simple Document Retrieval Experiment 1

### 5.2.2 Experiment 2 - large data set

For this experiment we used 11019 news stories. The indexing was performed using the 11026 captions in the data set. Then the 11019 story texts were used as a query to perform document retrieval. Again, the top 5 results were recorded. The following table gives a summary of the experiment.

Total number of news stories used in the experiment	11019
Total number of captions ( <i>used to create the index</i> )	11026
Total number of story texts ( <i>used as the query</i> )	11019

Table 5.3: Simple Document Retrieval Experiment 2

## 5.3 Experiments using WordEx

The purpose of WordEx is to classify each word in the story text into one of two classes: *yes* or *no*. The yes-words (those belonging to the class *yes*), are the ones that are most likely to be present in the caption of an image that matches the story text. The following experiments test how well WordEx performs its function. We have tried a range of machine-learning algorithms: Naive Bayes, Nearest Neighbors, C4.5 decision trees[11], and Support

Vector Machines[3]. We also applied Boosting[15] to these algorithms.

Two data sets were used for these experiments. One data set was used to train WordEx, and the other was used to test WordEx. Both the training set and test set were obtained from a corpus of 938 news stories. This corpus is described in detail in Section 4.1. The training set consists of 968 instances, with the two classes almost evenly distributed. There are 480 instances of the class *yes* and 488 instances of the class *no*. The test set consists of 19370 instances. The two classes are distributed as they would be for an average news story. There are 1599 instances of the class *yes* and 17771 instances of the class *no*. The yes-words make up about 9% of the test set. Table 5.4 summarizes the training set and test set.

training set	instances of class <b>yes</b>	480
	instances of class <b>no</b>	488
	total number of training instances	968
test set	instances of class <b>yes</b>	1599
	instances of class <b>no</b>	17771
	total number of test instances	19370

Table 5.4: Experiments using WordEx

Twelve experiments were carried out in total, each experiment with a different classifier. In the cases where boosting was applied, AdaBoostM1 was ran with 10 boost iterations. These experiments helped us decide which classifier was best suited for WordEx. The following is the list of classifiers tested:

- Simple Naive Bayes
- Simple Naive Bayes with AdaBoostM1
- 3 Nearest Neighbors
- 3 Nearest Neighbors with AdaBoostM1
- 5 Nearest Neighbors
- 5 Nearest Neighbors with AdaBoostM1

- 7 Nearest Neighbors
- 7 Nearest Neighbors with AdaBoostM1
- C4.5 Decision Trees
- C4.5 Decision Trees with AdaBoostM1
- Support Vector Machines with a linear kernel
- Support Vector Machines with a linear kernel and AdaBoostM1

### 5.3.1 Evaluation

As described in Section 3.3.2, the purpose of the yes-words predicted by WordEx is so they can be used to find a suitable image for the story text, by retrieving the correct image caption. To evaluate how well the yes-words fulfill their purpose, we carried out the following experiments.

#### Caption Retrieval

This experiment is the same as the one described in Section 5.2.2. Again, 11026 captions from 11019 different news stories were used to create an index. This time however, the full story text was *not* used as the query to perform the retrieval. Instead, the yes-words predicted by WordEx were used as the query. On average, the number of yes-words returned by WordEx for each story text was 62.<sup>1</sup>

WordEx of course, does not achieve 100% accuracy when predicting yes-words. We wanted to see however, how well the caption retrieval would be, if WordEx was 100% accurate. Therefore, we also performed caption retrieval with the *actual* yes-words of each news story. On average, the number of actual yes-words in each story text is 15.<sup>1</sup>

---

<sup>1</sup>For the data set used in this experiment, each story text contains 566 words on average, and each caption text has 49 words on average.

## Psychophysics Experiment

The purpose of this experiment is to see how humans rate the images chosen for news stories. We created an on-line web survey<sup>2</sup> where people were presented with a news story together with an image, and they were asked to rate how well the image matches the news story. Figure 5-1 on page 46 shows an example news story, asking the user to evaluate it.

There were 200 news stories in the sample to be evaluated. They were all taken from a corpus of 3695 news stories. Half of these news stories were exactly as they appeared on-line at the news site. We call this the **actual** set. The other half were a story text, together with an image that was predicted by our algorithm (using WordEx and then caption retrieval, as described above). We call this the **predicted** set. The evaluator had four choices for each news story in order to indicate how well the image matches the story text. The choices were poorly, adequately, very well, or cannot decide. The story presented to the evaluator was chosen at random every time, from the sample of the 200 news stories. The evaluator is not aware if a news story is one from the **actual** set or from the **predicted** set. The experiment therefore compares how good our predictions are compared to actual news stories that appear on-line. Each person was asked to evaluate about 20 stories.

### 5.3.2 Summarization

Using the yes-words predicted by WordEx, we also performed a summarization experiment. The method used is the following:

1. Take the original text to be summarized and give it as input to WordEx
2. The output is a list of yes-words.
3. For each sentence in the original text, rank each sentence depending on the number of yes-words contained in the word. A sentence with a high content of yes-words would therefore rank higher.

---

<sup>2</sup><http://spada.mit.edu/cgi-bin/evaluation.pl>

4. Sort each sentence in the original text by rank, and return the  $n$  sentences with the highest rank. Where for example  $n$  is 5.
5. These  $n$  sentences are the summary of the original text.

We performed summarization on 100 news story texts.



ABOARD USS KITTY HAWK (AFP) - Two US pilots escaped with minor injuries when they ejected moments before their tanker aircraft slid off a carrier, the navy's Fifth Fleet confirmed. The pilots of the S-3B Viking airborne refuelling plane received minor injuries when it slid off the deck of the USS Constellation about 0210 GMT, said a news release received aboard the Kitty Hawk, another carrier engaged in bombing missions over Iraq ( news - web sites ). "Shortly after touching down on the deck, the S-3B malfunctioned while taxiing on the carrier's flight deck and slid to the port side of the deck," the release said Tuesday. "The plane went over the side, hit flight deck safety netting and the two pilots onboard ejected into the water. The plane then went into the water," it said. A Seahawk helicopter, on standby in case of a flight accident, dropped a rescue swimmer into the water to retrieve the pilots. The frigate USS Thach, also nearby in case of an emergency, sent a boat to the scene. "Cause of the S-3B malfunction is under investigation," the release said. The S-3Bs, also performing a refuelling role aboard the Kitty Hawk, are to be phased out and their tanking job taken over by the new F/A-18 Super Hornet attack fighters, a pilot has said.

**Please rate how well the picture matches the story :**

- ☐ poorly
- ☐ adequately
- ☐ very well
- ☐ cannot decide

Next

Figure 5-1: An example story with an image for evaluation by a person

## Chapter 6

# Results

### 6.1 Simple Classification

The captions from 938 news stories were used as the training set, and the story texts of those news stories were used as the test set. Six classifiers were used: Naive Bayes, 3-NN (Nearest Neighbors), 7-NN, 15-NN, 30-NN, and linear SVM. Each classifier returns a ranked list of the resulting classification. The results are ranked based on confidence. Table 6.1 shows the number of correct captions retrieved.

Rank	Correctly classified instances					
	Naive Bayes	3 NN	7 NN	15 NN	30 NN	SVM linear
1st	453	416	399	390	375	552
2nd	104	95	115	113	112	54
3rd	34	55	56	58	55	18
4th	29	0	32	39	34	17
5th	24	1	16	19	31	3
total	644	576	618	619	607	644
percentage	69%	60%	66%	66%	65%	69%

Table 6.1: Results of Simple Classification experiment

A correct caption is one which appeared on-line with the news story text that was given as a test instance. For example, the linear SVM returned the correct caption as the top-ranking result, for 552 story texts. For 54 story texts, the correct caption was returned as

the second-ranking result, and so on. We decided to record only the top 5 ranking results, for comparison purposes.

The linear SVM performs the best, and then the Naive Bayes classifier. We see that when using a small set of news stories (938 in this case), we can achieve a relatively good retrieval accuracy.

## 6.2 Simple Document Retrieval

The same 938 news stories that were used in the Simple Classification experiment were used, as well as a larger data set of 11019 news stories. Table 6.2 shows the number of *correct* captions retrieved. The full story text was given as input (the query). The output was a list of captions matching the query. The results (a list of captions) are ranked based on the similarity of the caption to the query. If the caption retrieved is the caption that appeared on-line with that news story text, then it is considered to be *correct*.

Rank	Number of correct captions	
	small data set (938 news stories)	large data set (11019 news stories)
<b>1st</b>	599	3024
<b>2nd</b>	81	1263
<b>3rd</b>	36	748
<b>4th</b>	18	504
<b>5th</b>	15	358
<b>total</b>	749	5897
<b>percentage</b>	80%	54%

Table 6.2: Results of simple document retrieval using the full story text as query

We can compare the Simple Document Retrieval to Simple Classification, using the small data set. We see that Document Retrieval performs substantially better. We also observe that performance falls significantly for the larger data set.



### 6.3 Using our Approach - WordEx

We used different classifiers to see which one performs the best at predicting yes-words. The detailed results returned by Weka[16] can be found in Appendix A. It is important to achieve a high recall rate for the yes-words and the no-words. Recall rate is how many instances were correctly classified, out of all the instances of that class. So the recall rate for the yes-words is how many yes-words were correctly classified as *yes* out of all the yes-words in the test set. Similarly, for the no-words, the recall rate for the no-words is how many no-words were correctly classified as *no* out of all the no-words in the test set. The success rate is the number of correctly classified instances out of all the test instances. The success rate in this experiment is not indicative of how well a classifier performs. The reason for this is that 92% of the test instances are no-words. So a classifier that classifies all instances as *no* would achieve a success rate of 92%. In such a case however, the recall rate for the yes-words would be 0%. It is important therefore to compare the recall rates as well.

Classifier	Recall		Success Rate
	no	yes	
Simple Naive Bayes	92.9%	37.3%	88.3%
Simple Naive Bayes & AdaBoostM1	92.9%	37.3%	88.3%
3 Nearest Neighbors	69.0%	71.4%	69.2%
3 Nearest Neighbors & AdaBoostM1	64.3%	68.4%	64.6%
5 Nearest Neighbors	69.0%	72.1%	69.3%
5 Nearest Neighbors & AdaBoostM1	69.0%	72.1%	69.3%
7 Nearest Neighbors	69.6%	72.1%	69.8%
7 Nearest Neighbors & AdaBoostM1	69.6%	72.1%	69.8%
C4.5 Decision Trees	75.9%	72.0%	75.6%
C4.5 Decision Trees & AdaBoostM1	75.2%	73.7%	75.1%
Linear SVM	70.0%	74.9%	70.4%
Linear SVM & AdaBoostM1	70.1%	74.7%	70.4%

Table 6.3: Results from different classifiers

The classifier we decided to use is C4.5 Decision Trees with AdaBoost. This is because it gives the best balance of recall for no-words and yes-words.

Once the yes-words are predicted, we use them as the query to the Information Retrieval

algorithm, which in turn returns a list of captions. Table 6.4 shows the results for two data sets: a small data set consisting of 938 news stories, and a large data set consisting of 11019 news stories. Three different queries were given as input to the retrieval algorithm:

1. the full story text (on average, each query contains 560 words)
2. the predicted yes-words, which were the output of WordEx (on average, each query contains 62 words)
3. the actual yes-words (on average, each query contains 15 words)

Rank	Number of correct captions					
	small data set (938 news stories)			large data set (11019 news stories)		
	full story	predicted yes- words	actual yes- words	full story	predicted yes- words	actual yes- words
<b>1st</b>	599	599	852	3024	2852	6169
<b>2nd</b>	81	83	47	113	1193	2157
<b>3rd</b>	36	38	11	58	719	1013
<b>4th</b>	18	11	5	39	530	512
<b>5th</b>	15	15	3	19	368	304
<b>total</b>	749	746	918	5897	5662	10155
<b>percentage</b>	80%	80%	98%	53%	51%	92%

Table 6.4: Number of correct captions retrieved using the full story, the predicted yes-words and the actual yes-words as the query

## 6.4 Human Evaluation

Table 6.5 shows how human evaluators rated the images that our algorithm selected. We compared this to how they evaluated articles that appeared on on-line news sites. An **on-line article** is a story text, together with the actual image that was used on the on-line news site. It is a news story just like it appeared on-line. A **predicted article** is a news story text, together with an image that was predicted by our algorithm. The algorithm uses WordEx to predict a suitable image for the story text. See page 46 for an example article that was presented for evaluation.

	<b>on-line articles</b>		<b>predicted articles</b>	
<b>Rating</b>	number of ratings	percentage	number of ratings	percentage
very well	339	45%	311	40%
adequately	239	32%	228	30%
poorly	176	23%	226	30%
cannot decide	46 ( <i>ignored</i> )	( <i>ignored</i> )	71 ( <i>ignored</i> )	( <i>ignored</i> )
total	754	100 %	765	100%

Table 6.5: Results of Human Evaluation

About 65 people took part in the evaluation.<sup>1</sup> In total, 1519 articles were ranked. We do not consider the “cannot decide” response as an answer. We can see that although the rankings for the predicted articles are slightly worse than those of the on-line articles, the two results are significantly similar.

---

<sup>1</sup>The evaluation was done on-line, on the WWW. The number 65 is the number of distinct IP addresses that were logged by our web server. The actual number of people who took part in the evaluation should therefore be about 65.

We performed a Z-test to see if the responses to the on-line articles differ significantly from the responses to the predicted articles.

For each of the ratings (very well, adequately, poorly) we calculated the  $Z_0$  test statistic, using the formula below.

$$\hat{p} = \frac{n_1\hat{p}_1 + n_2\hat{p}_2}{n_1 + n_2}$$

$$Z_0 = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\hat{p}(1-\hat{p})(\frac{1}{n_1} + \frac{1}{n_2})}}$$

$$H_0 : \hat{p}_1 = \hat{p}_2$$

$$H_1 : \hat{p}_1 \neq \hat{p}_2$$

The value of  $n_1$  is 754, and  $n_2$  is 765. The value of  $\hat{p}_1$  and  $\hat{p}_2$ , are the percentages of on-line and predicted articles, respectively. For the case of “very well” for example,  $\hat{p}_1$  is  $339/754 = 0.4496$  and  $\hat{p}_2$  is  $311/765 = 0.4065$ . We accept the hypothesis  $H_0$  if  $|Z_0| > 1.96$  for a 5% confidence test. Otherwise, we reject it for  $H_1$ . Table 6.6 shows the results of the test. We see that the “very well” and “adequately” ratings are significantly similar, whereas the “poorly” rating is significantly different between the on-line and predicted articles.

Rating	$Z_0$	comment
very well	1.696	significantly similar
adequately	0.800	significantly similar
poorly	-2.739	significantly different

Table 6.6: Results of Z-test

## 6.5 Summarization

Some examples of summarization can be found in Appendix B.

From the 100 news story texts that we summarized, we noticed some interesting results. The first two sentences in the story text, ranked significantly higher than the rest. This shows that the first two sentences are “rich” in yes-words. We would like to compare our method with the “Lead-Based” summarization method. Lead-Based summarization is a simple method: it chooses the first  $n$  sentences from a piece of text in order to produce a summary. Although it is simple, it has been shown by [10] that it ranks favorably as a summarization method. Our summarization method, gives very similar results to the Lead-Based method, due to the fact that the sentences closer to the start of a story text have a higher content of yes-words. Table 6.7 shows how the first 10 sentences of a story text rank, based on our yes-word ranking scheme. For example, the first sentences of all 100 news stories, had an average rank of 3.84. A rank of 1 indicates that the sentence ranked the highest, a rank of 2 indicates that it ranked second highest, and so on. In 51 of the 100 story texts that were summarized, the first sentence had a rank of 1.

Sentence	Average rank
1st	3.84
2nd	8.59
3rd	10.20
4th	10.01
5th	10.45
6th	11.15
7th	10.05
8th	11.40
9th	10.88
10th	11.63

Table 6.7: Interesting results obtained from ranking sentences based on yes-words



## Chapter 7

# Discussion

### 7.1 Conclusions

We have found that by using the Simple Classification method we described, we cannot obtain satisfactory results. Also, the fact that there are many classes (one for each news story), greatly slows down the algorithm. The correct approach is to use a Document Retrieval scheme: we have the story text, and we need to retrieve the caption of the image that is suitable for the story text. The problem is that the story text is too big to used as the query for the retrieval. We therefore used a feature reduction scheme, to reduce the number of words in the story text. The extracted features are called the “yes-words”. The word extraction engine is called WordEx. WordEx takes the story text, and outputs a smaller set of words, which can then be used as the query to the retrieval engine.

Although Table 6.4 shows that we obtain slightly better results by using the full story text as the query, it is worth noting that the full story text has many more words than the “predicted yes-words”. The story text has 560 words on average, whereas the the “predicted yes-words” are about 60. With a word reduction of about 90%, we can retrieve captions almost as well as by using the full story text. In the same table (Table 6.4), we can see that if WordEx reached 100% accuracy when predicting the yes-words, then we could achieve excellent results when performing the retrieval step.

The evaluation method described by Table 6.4 shows how many of the captions retrieved were the ones that appeared on-line with the news story. Our goal however, is not to find the actual image that appeared on-line with a certain news story. Our goal is to find an *appropriate* image for a news story. The *appropriateness* of an image should be judged by a human reader. That is why we performed the Human Evaluation Experiment, which showed that our predictions were satisfactory. The results of the experiment, shown in Table 6.5, indicate that the appropriateness of the our predicted images was almost as good as that of the images that appeared on the on-line news site.

## 7.2 Further Improvements

The most important improvement would be to increase the prediction accuracy of WordEx. This could be done in several ways.

- Experimenting with a wider range of classifiers in order to find a classifier that performs better than the one currently used.
- Adding more features to the feature vector of each word.
- Use weighted yes-words . In the current version of WordEx, each predicted yes-word has equal weight. The classifier used by WordEx however, assigns a confidence value to each yes word. If we can give a higher weight to yes words with higher confidence, this could improve the retrieval step.
- Use stemming.

It is worth noting that even if WordEx achieved an accuracy of 100%, it could still not predict all the words in the caption text, by only using the story text. The reason is that some of the words in the caption text are not present in the story text. Even if WordEx could predict *all* the yes-words correctly, it would only be able to predict about 52% of the words in the caption text.<sup>1</sup> The rest of the words in the caption text, are not present in the

---

<sup>1</sup>This is after the stop words are removed.



story text. One way to improve our retrieval step would be to try and predict these words by using the story text. This would could be called *feature expansion*, because we would be adding words that are not present in the story text.



## Chapter 8

# Contributions

- The formulation of a method that uses the *caption* of an image to select an appropriate image for a news story.
- Demonstrated the significance of “yes-words”.
- A word extraction engine called **WordEx** that can predict “yes-words”.
- Performed feature reduction of story text, with satisfactory results.



## Appendix A

# Detailed Results

In this appendix we include the detailed results obtained from the experiments described in Section 5.3. Binary classification is performed. The two classes are **yes** and **no**. The training set consisted of 968 instances and the test set consisted of 19370 instances. (See Table 5.3)

```

=== Error on test data ===

Correctly Classified Instances      17113      88.348  %
Incorrectly Classified Instances    2257      11.652  %
Kappa statistic                     0.2824
Mean absolute error                 0.1351
Root mean squared error             0.3166
Relative absolute error              27.0247 %
Root relative squared error         63.3169 %
Total Number of Instances          19370

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision    Recall    F-Measure    Class
0.929      0.627      0.943      0.929      0.936      no
0.373      0.071      0.322      0.373      0.346      yes

=== Confusion Matrix ===

      a      b  <-- classified as
16516  1255 |      a = no
 1002   597 |      b = yes

```

Figure A-1: Simple Naive Bayes

```

=== Error on test data ===

Correctly Classified Instances      17113      88.348  %
Incorrectly Classified Instances    2257      11.652  %
Kappa statistic                     0.2824
Mean absolute error                 0.3064
Root mean squared error             0.3461
Relative absolute error              61.2715 %
Root relative squared error         69.2273 %
Total Number of Instances          19370

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision    Recall    F-Measure    Class
0.929      0.627      0.943      0.929      0.936      no
0.373      0.071      0.322      0.373      0.346      yes

=== Confusion Matrix ===

      a      b  <-- classified as
16516  1255 |      a = no
 1002   597 |      b = yes

```

Figure A-2: Simple Naive Bayes with AdaBoost

```

=== Error on test data ===

Correctly Classified Instances      13402      69.1895 %
Incorrectly Classified Instances    5968      30.8105 %
Kappa statistic                    0.1655
Mean absolute error                 0.3429
Root mean squared error             0.4813
Relative absolute error             68.589 %
Root relative squared error        96.2579 %
Total Number of Instances          19370

```

```

=== Detailed Accuracy By Class ===

```

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.69	0.286	0.964	0.69	0.804	no
0.714	0.31	0.172	0.714	0.277	yes

```

=== Confusion Matrix ===

```

a	b	<-- classified as
12261	5510	a = no
458	1141	b = yes

Figure A-3: 3 Nearest Neighbors

```

=== Error on test data ===

Correctly Classified Instances      12514      64.6051 %
Incorrectly Classified Instances    6856      35.3949 %
Kappa statistic                    0.1225
Mean absolute error                 0.3417
Root mean squared error             0.522
Relative absolute error             68.3483 %
Root relative squared error        104.4091 %
Total Number of Instances          19370

```

```

=== Detailed Accuracy By Class ===

```

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.643	0.316	0.958	0.643	0.769	no
0.684	0.357	0.147	0.684	0.242	yes

```

=== Confusion Matrix ===

```

a	b	<-- classified as
11421	6350	a = no
506	1093	b = yes

Figure A-4: 3 Nearest Neighbors with AdaBoost

```

=== Error on test data ===

Correctly Classified Instances      13416      69.2617 %
Incorrectly Classified Instances    5954      30.7383 %
Kappa statistic                     0.1685
Mean absolute error                 0.3542
Root mean squared error             0.4626
Relative absolute error             70.8443 %
Root relative squared error         92.5168 %
Total Number of Instances          19370

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision    Recall    F-Measure    Class
0.69       0.279      0.965       0.69      0.805       no
0.721      0.31       0.173       0.721     0.279       yes

=== Confusion Matrix ===

      a      b  <-- classified as
12263  5508 |      a = no
  446   1153 |      b = yes

```

Figure A-5: 5 Nearest Neighbors

```

=== Error on test data ===

Correctly Classified Instances      13416      69.2617 %
Incorrectly Classified Instances    5954      30.7383 %
Kappa statistic                     0.1685
Mean absolute error                 0.3589
Root mean squared error             0.4895
Relative absolute error             71.7714 %
Root relative squared error         97.8961 %
Total Number of Instances          19370

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision    Recall    F-Measure    Class
0.69       0.279      0.965       0.69      0.805       no
0.721      0.31       0.173       0.721     0.279       yes

=== Confusion Matrix ===

      a      b  <-- classified as
12263  5508 |      a = no
  446   1153 |      b = yes

```

Figure A-6: 5 Nearest Neighbors with AdaBoost



```

=== Error on test data ===

Correctly Classified Instances      13517      69.7832 %
Incorrectly Classified Instances    5853      30.2168 %
Kappa statistic                     0.1728
Mean absolute error                 0.36
Root mean squared error             0.4537
Relative absolute error              71.9928 %
Root relative squared error         90.7366 %
Total Number of Instances          19370

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision    Recall    F-Measure    Class
0.696      0.279      0.965      0.696      0.809      no
0.721      0.304      0.176      0.721      0.283      yes

=== Confusion Matrix ===

      a      b  <-- classified as
12364  5407 |      a = no
  446   1153 |      b = yes

```

Figure A-7: 7 Nearest Neighbors

```

=== Error on test data ===

Correctly Classified Instances      13517      69.7832 %
Incorrectly Classified Instances    5853      30.2168 %
Kappa statistic                     0.1728
Mean absolute error                 0.3659
Root mean squared error             0.4741
Relative absolute error              73.176 %
Root relative squared error         94.821 %
Total Number of Instances          19370

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision    Recall    F-Measure    Class
0.696      0.279      0.965      0.696      0.809      no
0.721      0.304      0.176      0.721      0.283      yes

=== Confusion Matrix ===

      a      b  <-- classified as
12364  5407 |      a = no
  446   1153 |      b = yes

```

Figure A-8: 7 Nearest Neighbors with AdaBoost

```

=== Error on test data ===

Correctly Classified Instances      14648      75.6221 %
Incorrectly Classified Instances    4722      24.3779 %
Kappa statistic                     0.2297
Mean absolute error                 0.342
Root mean squared error            0.4236
Relative absolute error             68.3916 %
Root relative squared error        84.7203 %
Total Number of Instances          19370

```

```

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall   F-Measure   Class
0.759     0.28      0.968      0.759    0.851      no
0.72      0.241     0.212      0.72     0.328      yes

```

```

=== Confusion Matrix ===

      a      b  <-- classified as
13496  4275 |      a = no
  447   1152 |      b = yes

```

Figure A-9: C4.5 Decision Trees

```

=== Error on test data ===

Correctly Classified Instances      14544      75.0852 %
Incorrectly Classified Instances    4826      24.9148 %
Kappa statistic                     0.2293
Mean absolute error                 0.2841
Root mean squared error            0.4275
Relative absolute error             56.824 %
Root relative squared error        85.4922 %
Total Number of Instances          19370

```

```

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall   F-Measure   Class
0.752     0.263     0.97      0.752    0.847      no
0.737     0.248     0.211     0.737    0.328      yes

```

```

=== Confusion Matrix ===

      a      b  <-- classified as
13365  4406 |      a = no
  420   1179 |      b = yes

```

Figure A-10: C4.5 Decision Trees with AdaBoost

```

=== Error on test data ===

Correctly Classified Instances      13643      70.4337 %
Incorrectly Classified Instances    5727      29.5663 %
Kappa statistic                     0.187
Mean absolute error                 0.2957
Root mean squared error            0.5437
Relative absolute error             59.1327 %
Root relative squared error        108.7499 %
Total Number of Instances          19370

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision    Recall    F-Measure    Class
0.7         0.251         0.969        0.7         0.813        no
0.749       0.3           0.184        0.749       0.295        yes

=== Confusion Matrix ===

      a      b  <-- classified as
12446  5325 |      a = no
  402   1197 |      b = yes

```

Figure A-11: Linear SVM

```

=== Error on test data ===

Correctly Classified Instances      13646      70.4491 %
Incorrectly Classified Instances    5724      29.5509 %
Kappa statistic                     0.1866
Mean absolute error                 0.356
Root mean squared error            0.4405
Relative absolute error             71.1911 %
Root relative squared error        88.1055 %
Total Number of Instances          19370

=== Detailed Accuracy By Class ===

TP Rate    FP Rate    Precision    Recall    F-Measure    Class
0.701       0.253         0.968        0.701       0.813        no
0.747       0.299         0.183        0.747       0.294        yes

=== Confusion Matrix ===

      a      b  <-- classified as
12452  5319 |      a = no
  405   1194 |      b = yes

```

Figure A-12: Linear SVM with AdaBoost



## Appendix B

# Generated Summaries

In this appendix we include some examples of summaries that were generated by using the method described in Section 5.3.2.

NEW YORK - Pakistan's leader said in an interview published on Saturday that he needs visible U.S. gestures, such as the release of American F-16 fighters sold to his country over a decade ago, to help blunt domestic criticism of his decision to support the U.S. bombing in Afghanistan. President Pervez Musharraf, in an interview with the New York Times, said he would ask President Bush for such concrete steps when they meet on Saturday. The two leaders are in New York for a meeting of the U.N. General Assembly. The Times said Musharraf put a particular emphasis on the release of the F-16s because their arrival would be a strong signal that the United States was restoring Pakistan to the stature of a genuine ally. Pakistan bought the 28 F-16s in the 1980s, but the U.S. Congress cut off all aid and military sales to Islamabad in 1990 due to Pakistan's secret nuclear weapons development program. Delivery of the airplanes was blocked, even though they already had been paid for. Musharraf told the Times that Pakistan even had received a bill for the storage of the aircraft while they are being held in Arizona. Pakistan, which shares a long border with Afghanistan, is a key ally in the United States' war against the al Qaeda network of Islamic militants and its Saudi-born leader, Osama bin Laden, thought to be behind the Sept. 11 attacks on the United States. Afghanistan's Taliban regime has housed bin Laden and the group and refused to hand him over to the United States. People in Pakistan, a U.S. Cold War ally, feel the United States abandoned the country after 1989, when the Soviet Union pulled out of Afghanistan, he told the paper. The United States then imposed economic sanctions when Pakistan developed nuclear weapons in response to India's nuclear program and the two countries conducted tit-for-tat nuclear tests in 1998. Major debt relief, military assistance and understanding on the subject of Pakistan's nuclear capabilities would go a long way toward alleviating those wrongs, the paper quoted Musharraf as saying. Pakistan's president also said he was worried that there was no broad-based coalition waiting in the wings in Afghanistan to replace the Taliban, the Times reported. The Northern Alliance, the leading opposition force now in Afghanistan, is made up largely of non-Pashtuns. Pashtuns are the dominant ethnic group in Afghanistan.

Figure B-1: News Story A

NEW YORK - Pakistan's leader said in an interview published on Saturday that he needs visible U.S. gestures, such as the release of American F-16 fighters sold to his country over a decade ago, to help blunt domestic criticism of his decision to support the U.S. bombing in Afghanistan. The United States then imposed economic sanctions when Pakistan developed nuclear weapons in response to India's nuclear program and the two countries conducted tit-for-tat nuclear tests in 1998. People in Pakistan, a U.S. Cold War ally, feel the United States abandoned the country after 1989, when the Soviet Union pulled out of Afghanistan, he told the paper. Pakistan, which shares a long border with Afghanistan, is a key ally in the United States' war against the al Qaeda network of Islamic militants and its Saudi-born leader, Osama bin Laden, thought to be behind the Sept. 11 attacks on the United States.

Figure B-2: Summary of News Story A

By Jonathan Wright UNITED NATIONS - Israeli and Palestinian leaders went their separate ways at the United Nations on Sunday, and the United States said again that President Bush had no plans to meet Palestinian President Yasser Arafat. Israeli Foreign Minister Shimon Peres had talks with Secretary of State Colin Powell, while Arafat met with U.N. Secretary-General Kofi Annan, officials said. Peres and Arafat would not be able to meet, an Israeli official added. Arafat is expected to see Powell before they leave New York, but no time has been set. Even a brief encounter between Arafat and Bush appeared to be out of the question. "We have no plans to meet with Chairman Arafat," Bush's national security adviser, Condoleezza Rice, told ABC. "The president has always said that he will use his meetings to advance causes, not just for the sake of meetings," she said on the "This Week" program. Bush's refusal to meet Arafat in New York has shocked Arab commentators, who say the Israeli-Palestinian conflict urgently needs a heavy dose of U.S. mediation, especially when the United States is trying to win Arab support for its campaign against the Taliban and al Qaeda in Afghanistan. Rice repeated the U.S. position that Bush would not see Arafat until Washington was satisfied Arafat had done enough to crack down on Palestinians trying to attack Israelis. "We need to see action from Chairman Arafat and the Palestinian Authority," she told ABC. "Until there is a real effort at the cessation of terrorism, it's going to be very hard to get the peace process going." "Secretary of State Powell is seeking an opportunity and will likely meet with Chairman Arafat sometime while he's here in New York, and so this is moving along," she added. In his speech to the U.N. General Assembly Saturday, Bush made only a brief reference to the Middle East conflict and repeated the U.S. position of the past few weeks. MUBARAK WELCOMES BUSH SPEECH Bush said: "The American government also stands by its commitment to a just peace in the Middle East. We are working toward a day when two states -- Israel and Palestine -- live peacefully together, within secure and recognized borders. "We will do all in our power to bring both parties back into negotiations. But peace will only come when all have sworn off -- forever -- incitement, violence, and terror." Egyptian President Hosni Mubarak Sunday welcomed Bush's support for the creation of a Palestinian state, but officials said Egypt wanted to see practical follow-up steps. An Israeli official said Powell did not present Peres with a comprehensive plan for peace and that Washington was stepping up the pressure on Arafat. "Now the world demands from Arafat to act according to his statements against terrorism and now he must prove it," Peres told his aides after the meeting, the official said. "While the United States still calls for Israel to continue withdrawing, as Israel plans to, from Area A, the United States now puts extra pressure on Arafat to fight terrorism," added the official, who asked not to be identified. When Israel reoccupied West Bank towns last month after the assassination of an Israeli Cabinet minister, the United States at first called for an immediate withdrawal. It has since toned down the urgency of its demand. At the meeting between Arafat and Annan, the U.N. secretary-general briefed Arafat on his talks on the Middle East with Bush and French President Jacques Chirac. "The Secretary-General assured President Arafat that he would continue his efforts, working closely with other key players including the United States, the European Union and the Russian Federation," a U.N. statement said.

Figure B-3: News Story B

By Jonathan Wright UNITED NATIONS - Israeli and Palestinian leaders went their separate ways at the United Nations on Sunday, and the United States said again that President Bush had no plans to meet Palestinian President Yasser Arafat. At the meeting between Arafat and Annan, the U.N. secretary-general briefed Arafat on his talks on the Middle East with Bush and French President Jacques Chirac. "The Secretary-General assured President Arafat that he would continue his efforts, working closely with other key players including the United States, the European Union and the Russian Federation," a U.N. statement said. Rice repeated the U.S. position that Bush would not see Arafat until Washington was satisfied Arafat had done enough to crack down on Palestinians trying to attack Israelis. "We need to see action from Chairman Arafat and the Palestinian Authority," she told ABC. "Until there is a real effort at the cessation of terrorism, it's going to be very hard to get the peace process going." "Secretary of State Powell is seeking an opportunity and will likely meet with Chairman Arafat sometime while he's here in New York, and so this is moving along," she added. Bush's refusal to meet Arafat in New York has shocked Arab commentators, who say the Israeli-Palestinian conflict urgently needs a heavy dose of U.S. mediation, especially when the United States is trying to win Arab support for its campaign against the Taliban and al Qaeda in Afghanistan.

Figure B-4: Summary of News Story B

SOUTH ORANGE, N.J. - Religion and spiritual teachings can help people transcend political differences and should never be used to rationalize terrorism, Iranian President Mohammad Khatami said. The reformist president and Muslim cleric addressed the United Nations on Friday, then spoke at Seton Hall University, a Roman Catholic school that has participated in a U.N. peace program Khatami started called "Dialogue Among Civilizations." "Unjustifiable human error has often turned religion into an instrument aimed at justifying inhuman behavior and restricting the scope of human interaction," Khatami said at the university. "But this surely contradicts the purpose of God and divine messenger." Roman Catholic Cardinal Theodore McCarrick, the archbishop of Washington, echoed the Iranian leader's call for more interfaith dialogue. He quoted Pope John Paul II and the Quran to make the point. "The bishops have striven to help our Catholic people realize that we are all brothers and sisters in God's one human family," McCarrick said. Khatami, whose predominantly Shiite Muslim nation was once seen as the chief exporter of Islamic radicalism, condemned terrorists who use the religion to promote their cause. He did not directly refer to the Taliban, but Iran believes the Taliban have warped Islam, and Iran has helped arm the opposition northern alliance in Afghanistan. "We see how an obscurantist misrepresentation of Islam terrorizes the world and whoever does not share in its fanatical illusions, subjecting innocent women, men and children to blind wrath misnamed a holy war or jihad," he said. Khatami made similar remarks in his speech to the U.N. General Assembly. He declared the perpetrators of the Sept. 11 terror attacks "a cult of fanatics" disconnected from the larger world and said they had carried out "an appalling crime." However, he also urged caution in the response. "A misplaced sense of might could lead to failure to hear the calls of people of goodwill or the cries of children, women and the elderly in Afghanistan," Khatami said. He said he had urged U.N. Secretary-General Kofi Annan to create an agenda to combat terrorism and unify international political will. John Negroponte, the U.S. envoy to the United Nations, also urged dialogue. "The greater danger confronting us in the world today is not that we speak in different languages, but that we don't always listen to any language," Negroponte said. At Seton Hall, Khatami also referred to the plight of Palestinians in the Israeli territories, but didn't mention the Jewish state by name. He said the teachings of Moses were incompatible with the "violent, racist misinterpretation that has driven a nation out of its homeland."

Figure B-5: News Story C

The reformist president and Muslim cleric addressed the United Nations on Friday, then spoke at Seton Hall University, a Roman Catholic school that has participated in a U.N. peace program Khatami started called "Dialogue Among Civilizations." "Unjustifiable human error has often turned religion into an instrument aimed at justifying inhuman behavior and restricting the scope of human interaction," Khatami said at the university. "But this surely contradicts the purpose of God and divine messenger." Roman Catholic Cardinal Theodore McCarrick, the archbishop of Washington, echoed the Iranian leader's call for more interfaith dialogue.

SOUTH ORANGE, N.J. - Religion and spiritual teachings can help people transcend political differences and should never be used to rationalize terrorism, Iranian President Mohammad Khatami said.

He did not directly refer to the Taliban, but Iran believes the Taliban have warped Islam, and Iran has helped arm the opposition northern alliance in Afghanistan. "We see how an obscurantist misrepresentation of Islam terrorizes the world and whoever does not share in its fanatical illusions, subjecting innocent women, men and children to blind wrath misnamed a holy war or jihad," he said.

At Seton Hall, Khatami also referred to the plight of Palestinians in the Israeli territories, but didn't mention the Jewish state by name.

Figure B-6: Summary of News Story C



# Bibliography

- [1] Eric Brill. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, pages 152–155, Trento, IT, 1992.
- [2] Nicol Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, 1997.
- [3] R. Cooley. Classification of news stories using support vector machines, 1999.
- [4] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
- [5] Jing Huang, S. Ravi Kumar, and Ramin Zabih. An automatic hierarchical image classification scheme. In *ACM Multimedia*, pages 219–228, 1998.
- [6] Toru Imai, Richard Schwartz, Francis Kubala, and Long Nguyen. Improved topic discrimination of broadcast news using a model of multiple simultaneous topics. In *Proc. ICASSP97*, pages 727–730, 1997.
- [7] A. McCallum. Multi-label text classification with a mixture model trained by em, 1999.
- [8] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/mccallum/bow>, 1996.
- [9] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1996.

- [10] Luis Perez-Breva and Osamu Yoshimi. Model selection in summary evaluation.
- [11] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.
- [12] Yong Rui, Thomas S. Huang, and Shih-Fu Chang. Image retrieval: Past, present, and future. In *International Symposium on Multimedia Information Processing*, 1997.
- [13] G. Salton and M. J. McGill. *The SMART and SIRE experimental retrieval systems*. McGraw-Hill, New York, 1983.
- [14] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
- [15] Robert E. Schapire. A brief introduction to boosting. In *IJCAI*, pages 1401–1406, 1999.
- [16] Ian H. Witten, Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations.
- [17] Yiming Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, 1(1-2):69–90, 1999.