ABSTRACT

Throughout the past decade machine learning research has attempted to address several challenges that the internet and its dynamic nature have presented. Particularly interesting are problems concerning classification. One way to study this area is by examining syndicate web sites that publish an .RSS file for their site. This file also known as a “feed” is essentially a listing of pages on their site with each containing a headline, URL, description, as well as other information. This information is updated on a regular basis by the web site and can be downloaded by users who have a program that can read the file. These programs, called “aggregators,” up to now only display the headlines and allow the user to browse to a particular story that they find interesting. This project will attempt to incorporate a machine learning algorithm (Naive Bayes Classifier) to find a way to classify news stories based upon their RSS information. Using RSS information in order to classify will be compared to using the full text of the article.
# TABLE OF CONTENTS

Abstract .................................................................................................................................................. ii

1. Background and Rationale ............................................................................................................. 1
   1.1 The classification problem ......................................................................................................... 1
       1.1.1 A learning scenario ............................................................................................................. 2
       1.1.2 Learning limitations ............................................................................................................ 3
   1.2 Target concept and Bayesian learning ....................................................................................... 3
   1.3 RSS (Really Simple Syndication) ............................................................................................. 5
       1.3.1 Aggregators ....................................................................................................................... 6

2. A Learning Aggregator ..................................................................................................................... 8
   2.1 Implementing an RSS aggregator ............................................................................................... 8
       2.1.1 Keyword filtering ............................................................................................................... 9
       2.1.2 Classification based on headlines ..................................................................................... 10
   2.2 Implementing a Naive Bayes Classifier ...................................................................................... 11
   2.3 Training .................................................................................................................................... 11

3. System Design .................................................................................................................................. 13
   3.1 User interface ............................................................................................................................. 14
       3.1.1 Adding an RSS feed to the Feed List ................................................................................ 14
       3.1.2 Deleting an RSS feed ......................................................................................................... 15
       3.1.3 Accessing the RSS Feeds ................................................................................................. 15
       3.1.4 Viewing the Raw XML ..................................................................................................... 16
       3.1.5 Viewing the keywords for the headline ............................................................................ 17
       3.1.6 Viewing the headline url .................................................................................................. 17
5.1.4 Further analysis..................................................................................................................39

5.2 Issues with the training and testing process ........................................................................40

5.2.1 Bias of the RSS Learner.................................................................................................40

5.2.2 Unlearning a learner........................................................................................................41

5.3 Project Evaluation ..............................................................................................................42

5.3.1 Limitations......................................................................................................................42

5.3.2 Storage requirements .....................................................................................................42

5.3.3 Further research..............................................................................................................42

Bibliography and References.....................................................................................................44

Appendix A RSS (2.0) Example ...............................................................................................45

Appendix B RSS Log file example ..........................................................................................47

Appendix C Sample Testing Sheet ..........................................................................................48
LIST OF FIGURES

2.1 Example XML file ................................................................. 8
3.1 User Interface ................................................................. 13
3.2 Viewing the Feed List ......................................................... 14
3.3 Adding a url ................................................................. 14
3.4 Deleting a feed ................................................................. 15
3.5 Viewing the Headlines and URLs ........................................ 15
3.6 Showing the xml .............................................................. 16
3.7 Display of the xml ............................................................ 16
3.8 Viewing the keywords ....................................................... 17
3.9 Keywords display window .................................................. 17
3.10 Viewing the headline url link ............................................ 18
3.11 Viewing the headline totals ............................................... 19
3.12 Viewing probability results ............................................... 19
3.13 Viewing the calculation .................................................... 20
3.14 Reset the database .......................................................... 22
3.15 Viewing the keywords in database .................................... 22
3.16 Viewing the keywords by category .................................... 23
3.17 Starting a training session .................................................. 24
3.18 The training panel ............................................................ 25
3.19 A Classification Example .................................................. 29
4.1 Beginning testing ............................................................. 31
4.2 Testing Panel ................................................................. 32
4.3 Testing in progress .................................................................33
4.4 The program’s classification guess ........................................34
4.5 Session information .................................................................34
5.1 Learning progress .................................................................37
5.2 Learning progress .................................................................38
5.3 Overall Computer performance .............................................39
LIST OF TABLES

2.1 Essential Elements of an RSS file.................................................................9

2.2 Headline classifications ..............................................................................10

4.1 Headline Database Statistics ....................................................................35
1. BACKGROUND AND RATIONALE

As the internet has grown over the past decade, new protocols, formats and standards have emerged (i.e., TCP/IP, HTML, XML). Each standard, as it has been accepted and implemented, has opened the floodgates of communication and information over the Internet. One such format, which has been used since its creation in 1999, is RSS (Really Simple Syndication or Rich Site Summary), however it is not so widely supported as of yet. Mostly used by syndicate web sites, it is a way by which a site can publish succinct information about their content whether it be the latest headlines or recently updated web pages. [Winer 2002]

1.1 The classification problem

The Internet has also posed new classification problem for machine learning research. Internet search engines index literally millions of web pages to allow queries based upon keywords. However, classification for such online content is a bit more of a challenge, particularly because of the dynamic nature of the Internet. A web page from 5 years ago that contained the word “Iraq” may have been classified as being about economics and oil, where today a page with the same word would most likely be about war and Iraqi freedom. Programs that process RSS files can be a good starting point for online classification research because of the growing number of sites which provide this format. These programs give access to the latest information and contain several types of information related to articles that can be useful for effective classification.
In machine learning, classification techniques are often implemented in many different ways from Artificial Neural Networks, to Decision Tree algorithms such as ID.4.5. However, Bayesian learning has proven to be a fairly good performer when it comes to text processing [Mitchell 1997].

1.1.1 A learning scenario

To better understand the machine learning process, imagine a scenario where there exists a teacher and a learner. The teacher provides examples of magazines articles for the learner to classify as being about one of 7 different categories: sports, U.S. News, Entertainment, World News, Business, Science/Technology, Health. The learner examines each article and makes notes about the keyword it contains. At first, the learning has a hard time classifying the magazines because (it is assumed) he starts out with no prior knowledge of what keywords a typical “business” or “sports” article contains. However, after being told what how each article should be classified, he is able to correspond certain keywords with certain classifications.

For example, if the learner reads an article containing the following keywords: Spurs, basketball, David Robinson, San Antonio, NBA and playoffs, he will learn that these words are contained in a sports headline. Other series of keywords such as economy, stock market, Dow Jones, profit margin, would correspond to a business article. However, what about the keywords that may appear in several different classifications? Would the word “basketball” ever appear in a business article? Of course it would, in which case the learner knows that the correct classification is not based upon one keyword “basketball” which may be found more often in Sports articles. Rather the classification
must be calculated based on the complete set of keywords. This learning process is what Bayesian learning attempts to implement.

### 1.1.2 Learning limitations

After the training session, the learner may become quite proficient at classifying the training data. However, once the learner is turned out to the real world, and set to work on data from different distributions, he will most likely not be 100% successful. The teacher cannot possibly provide enough examples to cover every keyword and keyword combination for every classification and the learner may not be intelligent enough to classify all articles based upon keywords alone. [Mitchell 1997]

### 1.2 Target concept and Bayesian learning

In machine learning, researchers deal with hypotheses and “target concepts“. A hypothesis is a possible classification for an article, and the target concept is the “correct” classification the program should give when the it sees a particular set of keywords. A program must be designed in such a way that it can be given training examples and “learn” to classify them correctly. Then, its accuracy can be tested against another set of data.

The target concept is often represented as a mathematical function expressed in terms of probability, weights, or rewards. Bayesian learning is an approach to machine learning that is based upon Bayes theorem:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$  \hspace{1cm} (1.1)
Bayes theorem (Eq. 1.1) essentially states that given data \( D \), the probability for hypothesis \( h \) (or the classification) can be determined. For many machine learning situations, the theorem can be reduced to:

\[
h_{ML} \equiv \arg\max \ P(D|h) \tag{1.2}
\]

where \( h_{ML} \) represents the hypothesis which has the maximum likelihood (ML). The equation can be reduced to \( h_{ML} \) (Eq. 1.2) since \( P(D) \) is constant and independent of \( h \), and if every hypothesis has an equal chance to be correct, \( P(h) \) can be removed from the equation. It is desirable to have the hypothesis with the greatest probability given the data, or keywords in this case. [Mitchell 1997]

According to Mitchell, the probability must be calculated for the attribute values, which in this case consists of a set of keywords. This gives the following equation,

\[
V_{MAP} = \arg\max \ P(a_1, a_2...a_n | v_j) \ P(v_j) \tag{1.3}
\]

where \( V_{MAP} \) represents the target function (or target concept), \( v_j \) represents the instance, \( a_1, a_2...a_n \) represent the various attributes. Since a very large number of training examples are required in order to have a good estimate of \( P(a_1, a_2...a_n | v_j) \) for all the possible instances, this project (which resembles closely a Naive Bayes classification implementation) will use a classifier with a modified Bayes algorithm for the sake of performance [Mitchell 1997].

A Bayes learner (such as Naive Bayes) essentially keeps totals of each keyword for each classification. Each attribute is assumed to be conditionally independent. This assumption of course is not true, since for example, the two words “Dow” and “Jones”
often come together in business articles. However, this assumption allows us to derive the following equation,

\[ V_{NB} = \arg\max_{j} P(v_j) \prod_{i} P(a_i|v_j) \]  

(1.4)

and to estimate the \( P \) values of attributes based upon their frequencies. Here, the best classification is found by multiplying probability of the hypothesis \( v_j \) (or classification in this case) times the product of the probabilities of each attribute \( a_i \) given the hypothesis \( v_j \). The hypothesis with the highest total probability becomes the \( V_{NB} \). [Mitchell 1997]

1.3 RSS (Really Simple Syndication)

In 1999, an extension to the XML file format called RSS was adopted by the World Wide Web Consortium as a way for syndicate sites to publish information across the Internet. The first version of RSS stood for “RDS Site Summary”, where RDS stands for “Resource Description Framework”. Improvements were made on this initial standard and version 1.0 and was dubbed “Rich Site Summary.” By 2002, version 2.0 (renamed again to “Really Simple Syndication) had come out and is thought to probably be the final version. [Fallside 2001]

It is becoming increasingly popular for websites to provide an “RSS feed” with descriptive information concerning their site’s content. The idea is especially relevant for news sites where their content is changing daily. Visitors to these sites want to know what has changed, but without having to browse through the entire site or even wait for the homepage to load. Many syndication websites such as BBC, NBC, numerous entertainment and sports sites from around the world are providing their own RSS feed.
The RSS file is a relatively small file which contains such information as the site data, article, title, and summary data. Particularly for a person wanting to keep updated with several news sites, downloading the RSS files provides a way to scan a site’s information and save the user a lot of valuable time.

1.3.1 Aggregators

An application which downloads RSS files and provides a way for a user to browse to the links is called an “aggregator.” There are several examples of such applications such as Aggie (http://bitworking.org/Aggie.html) and UserLand’s products (http://www.userland.com/). However, none have been found which also incorporate machine learning algorithms specifically for classifying news stories.

An aggregator will generally download the RSS files to check for update on a regular basis. The time interval for downloading can be set to every minute to once a week, depending upon how often the user wants to check for new information. Aggregators can do all the checking in the background and only display the information when it is new. So, a user does not have to interrupt his work to check a site for new content. Rather, he can continue working and the aggregator will monitor the RSS feed and pop up the latest headlines as soon as they become available. A site called NewsIsFree (http://www.newsisfree.com) seems to have some “Advanced Aggregator” capability of searching based upon RSS file information and some type of classification, however, it is unclear how they implement this.

The remainder of this paper will describe the prosed program design of an aggregator and how a Naive Bayes learner will be incorporated into it. Chapter 2 will describe more details as to how this project will create an aggregator, analyze RSS files, use the
database for storage, and how the training and testing process will work. Chapter 3 will cover more of the implementation and coding details. Chapter 4 will cover the testing and evaluation process, and Chapter 5 will discuss the expected results of this project.
This project involves designing a program that can learn to classify online news stories using RSS files. There are essentially three phases of the project, all of which are typical for machine learning: design, training, and testing. Details of these phases will be covered in the following chapters.

### 2.1 Implementing an RSS aggregator

Designing an aggregator requires familiarity with HTTP protocol and XML format. XML is known to be a powerful markup language which is used in many types of Internet applications where complex data structures must be represented. [W3C 1999] An example of a partial XML file is shown below:

```xml
<?xml version="1.0" ?>
- <purchaseOrder orderDate="1999-10-20">
  - <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  - <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
  </billTo>
</purchaseOrder>
```

Figure 2.1
Example XML file (taken from World Wide Web Consortium) [Fallside 2001].
Some typical web browsers can display XML file contents (such as Internet Explorer 5.5), however, most browsers are not able to parse an RSS file, because the extension is generally .rss rather than .xml. So, another program must be designed that can recognize RSS files as XML files and parse the information contained in them.

There is only a small set of required tags that makes up a well-formed RSS file (An example is included in Appendix A), as shown in Table 2.1.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>The title of the RSS feed.</td>
</tr>
<tr>
<td>summary</td>
<td>The description of the RSS feed.</td>
</tr>
<tr>
<td>item</td>
<td>An item associated with the feed.</td>
</tr>
<tr>
<td>link</td>
<td>The absolute URI of the item associated with the feed.</td>
</tr>
<tr>
<td>title</td>
<td>The title of the item associated with the feed.</td>
</tr>
<tr>
<td>description</td>
<td>The description of the item associated with the feed.</td>
</tr>
<tr>
<td>date</td>
<td>The date of the item associated with the feed.</td>
</tr>
</tbody>
</table>

Since some sites do not publish well-formed RSS files, which means that one or more of the required elements may be missing, the program should be able to extract all the information it can. The Sindic8 project (found at http://www.syndic8.com/) has been an effort to get more sites to publish RSS feeds and so its site was a valuable resource in finding feeds for training data.

2.1.1 Keyword filtering

Text processing has been an area of research since the 1950’s and it has become even more a critical area for online processing that has grown out of the Internet. [Bae 2002]. Determining what the keywords are plays an important role in the effectiveness of
the learning algorithm. Keywords are extracted from the RSS file from only the headlines tag. It was originally thought that the use more fields such as description could help in this analysis, but this idea proved to be additional work and unrelated to the major focus of this project. So, headlines alone are used for the training and testing process.

2.1.2 Classification based on headlines

Headlines could be considered the shortest possible summary of the article. When considering headlines as opposed to the full text of a web page for classification, the amount of data that must be processed is drastically lower. If web pages can be classified with relatively good accuracy based upon their headlines, this classification can be done much faster.

This project will use 7 classifications for headlines. These 7 classifications were chosen because they have found on a similar news classification site already in existence (news.google.com).

Table 2.2 Headline classifications

<table>
<thead>
<tr>
<th>Classification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>World News</td>
<td>News involving other countries.</td>
</tr>
<tr>
<td>U.S. News</td>
<td>Local and national news, occurring in the U.S. and not involving another country.</td>
</tr>
<tr>
<td>Business</td>
<td>Any economic or money related news story. Could include some political stories.</td>
</tr>
<tr>
<td>Science/Technology</td>
<td>Any science story that is not related to health. Internet, computers, etc. related stories.</td>
</tr>
<tr>
<td>Sports</td>
<td>Any story related to sports.</td>
</tr>
<tr>
<td>Entertainment</td>
<td>Any story involving a musician, television/movie star, band, etc.</td>
</tr>
<tr>
<td>Health</td>
<td>Any story related to health or health-related technology.</td>
</tr>
</tbody>
</table>
Some research has already been done in the area of news classification using headlines and summary information along with article text for classification [Ahmad 2001]. However, the methods were not incorporated into a news aggregator program. The results of previous experiments have been helpful in working through this project.

2.2 Implementing a Naive Bayes Classifier

As pointed out in the previous chapter, a Naive Bayes classifier is based upon Bayes Theorem of probability and assumes conditional independence of each attribute. Thus, the RSS keywords are considered to be the attributes for each news article and the following equation is used to figure the probability of each classification:

\[
V_{NB} = \arg \max_j P(v_j) \prod_i P(a_i|v_j)
\] (2.1)

To discover the classification with the highest probability, each classification is examined and its probability multiplied by the total probability of the keywords. The classification with the highest probability becomes the desired hypothesis according to Naive Bayes (\(V_{NB}\)).

Keywords information is extracted from the RSS file and then the probabilities are computed. The more keywords, then the more accurate the prediction should be. The implementation of the Bayesian formula is discussed in more detail in Chapter 3.

2.3 Training

The training phase is a relatively simple endeavor. It begins by examining the RSS file extracting the information from the headlines and description information. This set of
keywords then becomes a vector of attributes \(<a_1, a_2, ..., a_n>\). Once the correct classification is known, (either by the user or another more automated means), the aggregator will examine the table for the classification to see if it is already present. If so, then it will increment the count for that keyword. There will be one table in the database for each classification with a column for the word and another for the frequency (or number of times it has occurred). If it is not present then it will add a new entry for the keyword and initialize the count to the number of times that keyword appeared in the article. It should not be too uncommon for one keyword to appear only in one table and not in all the tables.
3. SYSTEM DESIGN

The aggregator was designed in C++ using Microsoft’s Visual C++ IDE (Integrated Development Environment). The program is designed using the Document/View architecture for several reasons, namely to facilitate the exploration of the xml files that make up the RSS feeds, to allow the analysis of the contents of the database, and to display the computations and results that the program undergoes in order to classify the various headlines. Figure 3.1 shows a general view of the program user interface.

![User Interface](image-url)

Figure 3.1
User Interface
3.1 User interface

3.1.1 Adding an RSS feed to the Feed List

The upper left-hand region of the user interface shows the urls of the RSS Files. To add a feed to the list, the user clicks on **RSS > Add feed** (Figure 3.2).

An input box then shows allowing the user to add the url (Figure 3.3).
3.1.2 Deleting an RSS feed

To delete a feed from the list the user can right-click on the feed and click Delete (Figure 3.4).

3.1.3 Accessing the RSS Feeds

By clicking on a feed in the list, the headlines and corresponding url for that headline are displayed in the window to the right.

![Figure 3.4 Deleting a feed](image1)

![Figure 3.5 Viewing the Headlines and URLs](image2)
3.1.4 Viewing the Raw XML

To view the actual XML retrieved from the RSS URL, the user can right-click and then click on **RSS Feed > Show XML** for entire feed (Figure 3.6).

A window then pops up displaying the actual XML (Figure 3.7).
3.1.5 Viewing the keywords for the headline

The keywords for the headline can be observed by clicking on a headline, right-clicking and choosing **Headline > Show Headline Keywords** (Figure 3.8).

![Figure 3.8](image1)

Viewing the keywords

The keywords are displayed in a popup window (Figure 3.9).

![Figure 3.9](image2)

Viewing the keywords

3.1.6 Viewing the headline url

Particularly when training the RSS Learner, it is necessary to not only view the headline, but view the web page of the headline. To view the url link for the headline, the user can double-click on the headline and the web page is displayed in the browser window at the bottom right-hand corner of the program interface (Figure 3.10).

```markdown
| Alcohol Related Problems Linked to Certain Protein | http://abclocal.go.com/kabc/health/112403_hs
| Skin Cancer, More Serious Cancers Linked | Headline
| ADHD Suffers May have Different Brain St | Show Headline Keywords
| Can You Really Have Your Cake And Eat It | RSS Feed
| A Face Mask that can Help in the Battle of the Bulge | http://abclocal.go.com/kabc/health/111903_hs
```
3.1.7 Viewing the Training and Testing statistics

During training and testing it is necessary to view various statistics such as the number of headlines processed for each category, the number processed during the session, the percentage correct (for testing only) as well as the numbers of headlines processed overall. These totals can be viewed in the left-hand pane (Figure 3.11).
The totals can be toggled between the global counts from the database and the counts for only the current session. In the case of a testing, the percentage correct is also shown.

### 3.1.8 Viewing the Training Session Probability Results

During training, it is helpful to view the calculations the program makes in order to classify a headline.

![Figure 3.11 Viewing the headline totals](image)

The total number of headlines the program has learned.

The headlines learned for each category thus far.

Show the stats for the current training of testing session.

---

![Figure 3.12 Viewing probability results](image)

The program guesses the Health classification for this headline.

The probability calculated for each classification is displayed as well.

---

By clicking on the Calculation details button, a window is shown that contains the keyword counts in the database and the method of calculation by the program (Figure 3.13).
3.2 Database

The database used is a MySql database. Originally, Oracle was thought to be the best choice, but the availability of MySql and the fact that it no special features of Oracle are used, a standard relational database is all that was required.

The database holds the frequencies of the keywords for each classification. There is one table for each classification (i.e., one for Sports, another for Entertainment, etc.). The database schema is relatively simple. Each classification table has only two columns, keyword and frequency.

![Figure 3.13](image)

Viewing the calculation
An additional table was added to contain all the headlines learned. This table was necessary for two reasons:

1. Headlines should not be re-learned. If a headline is learned more than once this would place additional counts of keywords into the headline classification and give that classification slightly more weight than it would normally have.

2. The program should not be tested on the same headlines it was trained on. The point of this project is to determine how well a program can learn to classify new headlines, not how well it can recalculate the classification of a previously seen headline. Using headlines already learned would cause the program to perform better than it actually performs.

If a headline has been used for training, it must be skipped during the testing phase.

Another table stores all the keywords learned. This table is the master keyword storage and represents the “vocabulary” of the learner. Compared to the English language the table is relatively small (less than 5,000 words). The English language contains approximately 50,000 words meaning that the vocabulary of the learner represents less than 10% of possible words. [Mitchell 1997] It is worth noting that the maximum size of the vocabulary is the number of words in the English language, theoretically speaking. (Actually for this project the number would be slightly higher because different tenses of verbs are stored as distinct keywords. I.e., win, won, winning,...) The totals from this table are used during the calculation of the probability which is described later in this chapter.

3.2.9 Resetting the database

The database can be reset by clicking on Database > Reset Database (Figure 3.4). Doing this clears out all the tables. This was particularly useful during the initial training sessions where keywords needed to be enumerated and added to the correct tables.
3.2.10 Viewing the keywords in the database

By clicking on Database > Show keywords in database (Figure 3.5) the user can view the keywords and the frequencies for each category as well as the overall keywords in the vocabulary table.

A window shows with buttons for each category. When the user clicks on a category, the results from the table are displayed.
3.3 Log file

An important part of this project was to capture the results of each training and testing session. A log file outputs all actions during a session. A sample output for a testing session can be found in Appendix B. The format for this file is such that it can be imported as a tab delimited file into Microsoft Excel. The results of this Excel file could easily be imported into Microsoft Word to produce a testing sheet that could be printed and given to a human tester. A sample of this printed sheet can be found in Appendix C.
3.4 Training phase

The RSS Learner requires a user to view the headlines and “tell” the program the categories of the headlines it evaluates. Training the RSS Learner is completed through the following steps:

1. Select **RSS > Start** training from the menu.

![Starting a training session](image)

2. The program displays a window with buttons for each category (Figure 3.18). The program will begin at the first rss feed and display the headlines in the headline viewer window. The web page for the first headline will appear in the web page viewer window. The user can view the headline, view the web page and then click the button for the appropriate category.

3. The program automatically enumerates the keywords in the headline and when the user clicks the category the keywords are added to (or the counts incremented in) the appropriate category table.

4. If a rss feed is invalid (url is unreachable) the user can hit “Next Feed”.
5. If a headline is invalid (url is unreachable or has already been learned) the user can click the “Skip Headline”.

![Training Panel Diagram]

**Figure 3.18**
The training panel

### 3.4.11 Learning process

1. The RSS Learner extracts all the words from the headlines.

2. Any stop words are skipped. Stop words are words such as “the”, “and”, “at”, etc.

3. When the user clicks the category, the program checks the table for each keyword in that category. If the keyword is already present, the count is incremented. If the keyword is not present, it is added and the count is set to 1. At the same time, the count is incremented for the keyword in the vocabulary table.

### 3.4.12 Handling of phrases

Originally it was thought that an additional table was needed to store potential phrases. This table would store combinations of keywords and if a phrase appeared more than once in headlines, it would be added to the appropriate table. This was an attempt to
capture such keywords such as proper names (i.e., “Michael Jordan”) or other phrases (like “stock market”) that have significance as groups of words.

However, this idea was done away with due to questions of how to count the keywords if a phrase was found. For example, if the headline “stock market on the rise” was found in a business headline, then without dealing with phrases, the system would count three keywords (“on” and “the” are not keywords and are filtered out), and the count would be incremented in the business table for the keywords “stock”, “market”, and “rise”. If the system were to also keep track of phrases, then it would check the phrase table for a previous occurrence of “stock market”. If the phrase had appeared in an earlier headline, then it would be a legitimate phrase, otherwise the system would simply add it to the table in hopes of finding an occurrence of it in a later headline.

In calculating the probability of each category, the number of occurrences is totalled and divide by the total number of keywords in the table. If the system counts “stock market” as one keyword, then this would not result in a total higher than another category that had a count for each individual keyword. One option would be to count three keywords “stock market”, “stock”, and “market” for the business category. However, doing this would count a single keyword more than once. The focus of the project is the implementation of the Bayesian learning algorithm, so it was decided to keep the project simplified and not explore all the different keyword enumeration techniques. Furthermore, if the project were continued, the tables for each category could grow enormously. Limiting to single keywords and not allowing phrases means that the most each table could grow to would be the number of words in the English language (approximately 50,000 [Mitchell 1997]).
3.4.13 Handling of special html characters

As the program was developed, it was realized that special html characters such as “#amp;” or “&#039;”, that represent various characters such as apersands, semicolons, quotes, etc., did not resemble normal words in the English language. It was thought that these special characters should be converted to their ASCII equivalent values first before being stored in the database (for a keyword that by chance had a special symbol in it). However, since the program does not necessarily care whether a character is an ASCII apostrophe or the HTML symbol, any keywords that happened to have special symbols were left as is. Furthermore, these are so many special symbols to convert, that the effort needed to convert all such symbols was outside the scope of this project.

3.5 Implementation of Bayesian learning

During the testing phase is when most of the computation will be performed for the Bayesian learner. The probabilities will be calculated based on the frequency of keywords contained in the database. For calculating the probabilities, the process that will be used is as follows:
For each classification \( v_j \) in \( V \) and each attribute in \( <a_1, a_2, \ldots, a_n> \),

Calculate

\[
V_{NB} = \arg\max_{v_j} P(v_j) \prod_{i} P(a_i | v_j)
\]

For this project it may be helpful to think of a variation of the above Naive Bayes calculation:

\[
V_{NB} = \arg\max_{v_j} P(v_j) P(\sum a_i | v_j)
\]

The reason for this is that there is only one \( P(a_i | v_j) \) and this probability is found by taking the sum of the keyword instances for each classification.

\[
P(v_j) = \frac{\# \text{ of headlines processed for classification } v_j}{\text{total } \# \text{ of headlines processed}}
\]

\[
P(a_i | v_j) = \frac{\# \text{ of keywords in headline } + \# \text{ of keywords counts found for classification } v_j}{\text{total } \# \text{ keywords in learners vocabulary } + \text{total } \# \text{ keywords in table for } v_j}
\]

Probability result for \( v_j = P(vj) \times P(a_i | vj) \)

In calculating the probability for \( P(a_i | v_j) \), it is interesting to note that the number of keywords in headline is added to the numerator and total number of keywords in the learner’s vocabulary is added to the denominator. The justification for this comes by the fact that every classification has at least some probability of being correct even if there are no keywords present in the table. The absolute minimum probability for each classification is the number of keywords in the headline divided by the total number of keywords in the vocabulary. For example, if the headline has 5 keywords and the learner’s vocabulary has 2,000 words, then each classification has at least a 5/2000 chance of being correct. For every count that is found however, this number is added to the numerator and will
consequently increase the probability of the classification. Using this approach eliminates the problem of dealing with zero probability in the case of no keywords being found for a particular classification.

Figure 3.19 shows a simplified classification example.

---

**Figure 3.19**
A Classification Example

---

Headline with three classification possibilities and three keywords:

**HEADLINE:**

“Spurs look for **profits** in **advertising**”

Query the database and check how many classifications contain the keywords and how many instances of the word has appeared for each classification.

Database totals: (total headlines processed, 400; total keywords learned, 1350):
Sports (150 headline, 458 keywords)
Business (160 headline, 510 keywords)
Entertainment (90 headline, 382 keywords)

**Keyword:**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Sports</th>
<th>Business</th>
<th>Entertainment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spurs</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>profits</td>
<td>0</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>advertising</td>
<td>1</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td>6</td>
<td>13</td>
<td>7</td>
</tr>
</tbody>
</table>

P(ai|vj)  

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Sports</th>
<th>Business</th>
<th>Entertainment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spurs</td>
<td>.0050 (9/1808)</td>
<td>.0086(16/1860)</td>
<td>.0058 (10/1732)</td>
</tr>
<tr>
<td>profits</td>
<td>.375 (150/400)</td>
<td>.4 (160/400)</td>
<td>.225 (90/400)</td>
</tr>
<tr>
<td>advertising</td>
<td>.00187</td>
<td>.00344</td>
<td>.00130</td>
</tr>
</tbody>
</table>

The news article is therefore classified as a Business headline.
4. TESTING AND EVALUATION

Testing is an important part of any learning process. The RSS Learner provides an integrated way by which the program can be tested to evaluate its classification ability for headlines. During testing, the user gives the program a list of headlines which it classifies based on the keyword data it finds in the database. The user then tells the program if it classified correctly or not. During the testing session RSS Learner keeps track of its percent correct and writes the results to its log file.

4.1 Method

It was originally thought that the testing process could be a less interactive procedure, where the computer could monitor a list of RSS feeds and the user could periodically view the log file and evaluate classifications of the program. This proved to be more work since a additional program was required to evaluate the log files. The fact that a user still had to specify the correct classification at some point meant that the testing process was much easier if it was done interactively. In other words, the user would monitor the RSS Learner during the testing session and give the correct classifications as program needed them.

It is important to note that there is no learning taking place during testing for this project. The training and testing phases are kept separate. However, another implementation may combine the two to allow the program to add keywords to the database as it goes. For this project, the best method of doing both testing and training was for the user to first
test the program on a set of new headlines, then after getting the testing results, go back
and train the computer on the same list of headlines.

4.1.1 Beginning a testing session

To begin a testing session the user clicks first on RSS > Testing > Start testing
(Figure 4.1).

![Figure 4.1](image1)

Beginning testing

The Testing Panel appears with buttons for each classification (Figure 4.2). The panel is
similar to the panel used during training (Figure 3.18).
User clicks the correct classification for the headline.

If the link is dead, or if it has been used for training already, the user can skip it.

If the feed is a dead link, or if the entire feed has been used for training, the user can go to the next feed in the list.

The RSS Learner will go to the first RSS feed in the feed list and the first headline of that list (Figure 4.3).
4.1.2 Viewing the RSS Learners classification

The classification that the program gives to the headline can be viewed in the Classification Probability Result window in the left hand pane of the main window (Figure 4.4).
Additional information about the testing session such as number of headlines processed and the percent correct can be found in the Testing Session Data window in the left hand pane (Figure 4.5).

Program incorrectly guessed this is a Health headline.

The user can click on Calculation details to view the computation that computer performed to guess the classification.

Figure 4.4
The program’s classification guess

Figure 4.5
Session information
4.2 Evaluation

Originally, the goal was for the RSS Learner to perform with at least 80% classification accuracy. However, this accuracy seemed a bit arbitrary. Pure random guessing would yield 14% accuracy rate (this represents $1/n$ correct, with $n$ being the number of different classifications possible, $1/7$ in this case). So, it was decided that the Learner must be able to do at least better than random guessing and hopefully as good as a human can do.

4.2.1 Database content used for testing

After training for 3 weeks, the RSS Learner had processed a total of 1,072 headlines and 3,096 distinct keywords (Table 4.1).

<table>
<thead>
<tr>
<th>Classification</th>
<th>Number of headlines processed</th>
<th>Number of keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>World</td>
<td>138</td>
<td>502</td>
</tr>
<tr>
<td>U.S.</td>
<td>161</td>
<td>573</td>
</tr>
<tr>
<td>Sports</td>
<td>138</td>
<td>510</td>
</tr>
<tr>
<td>Business</td>
<td>105</td>
<td>467</td>
</tr>
<tr>
<td>Science/Technology</td>
<td>142</td>
<td>461</td>
</tr>
<tr>
<td>Entertainment</td>
<td>197</td>
<td>743</td>
</tr>
<tr>
<td>Health</td>
<td>191</td>
<td>735</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>1,072 headlines</strong></td>
<td><strong>3,991 keywords</strong></td>
</tr>
</tbody>
</table>

a. Some keywords are repeated in several classifications. There were 3,096 distinct keywords in the RSS Learner’s vocabulary.

4.2.2 Program learning

To compare the RSS Learner’s ability to that of humans, a group of 6 adults were given three sets of headlines from actual RSS feeds. Each set had at least 27 headlines (one contained 34) and contained a sampling of all seven categories (World, U.S., Sports,
Business, Sci/Tech, Entertainment, and Health). Each adult was given the same headlines (a sample of one testing sheet used is found in Appendix C). The RSS Learner was also given the same headlines. Chapter 5 presents the results of the testing as well as an evaluation of the project.
5. RESULTS AND CONCLUSION

Since classification research for the World Wide Web is not a new area by any means. Hopefully this project will lend some data to support the effectiveness of using RSS feeds for information retrieval.

5.1 Testing Results

5.1.1 Showing that the computer is learning

To verify that the program was in fact learning and its classification accuracy was improving a simple test was performed. A new database instance was created and used to train the program using only 57 headlines (compared to the 1,072 in the real instance). Testing on 40 headlines gave the computer only a 10% accuracy rate, which is less than random guessing. However, using the normal database instance the accuracy rate for the same 40 headlines jumped to 53% accuracy (Figure 5.1).

![Figure 5.1](image-url)
5.1.2 Human vs. Computer Results

The computer did not perform as well as expected with the three headline lists as compared to the humans. On one particular list it only scored an 11%. The other two were higher (33% and 50%). Figure 5.2. shows the computer’s average compared to each person’s average.

![Overall Performance on all three lists](image)

**Figure 5.2**
Learning progress

5.1.3 Computer performance

The computer did do roughly 2 times better than random guessing, although it did only half as good as the human average (Figure 5.3).
5.1.4 Further analysis

Speculation over why the Learner did perform as well as anticipated leads to several consideration:

More sampling would yield more accurate results

During other miscellaneous test sessions the RSS Learner often performed at over 50% accuracy rate and occasionally over 70%. These numbers are well above its performance in the “Human vs. Computer” testing. If there was more time for testing, more headline lists could have been tested and this could possibly have increased the Learner’s average accuracy rate.

Small Vocabulary of the RSS Learner

Even though the RSS Learner had 3,096 keywords in its vocabulary, this number represents only a small percentage (less than 10%) of the possible words it could have. It is interesting to think of this learner as being relatively “young”, perhaps having the same
number of words in its vocabulary as a child might. Based upon this idea it is easy to see why more training is necessary to more accurately represent the vocabulary of a human adult. When the RSS Learner did not match any keywords for a headline during a testing session, the probability of the headline defaulted to the classification with the most headlines learned.

*Only one RSS Learner was used*

There were 6 adults for the test and only one machine. The nature of the training process tends to be somewhat subjective, so the fact that one person trained the RSS Learner may have limited the computer’s ability when compared to a group of humans. One alternative would have each adult also train an RSS Learner. This would have given an average computer result to match the average human result. With the time required for doing this, it simply was not possible.

**5.2 Issues with the training and testing process**

**5.2.1 Bias of the RSS Learner**

Every machine learning scenario has some sort of bias. For this project there was a heavy bias towards the classification ability of the individual who trained it. In a situation where a general classification of news stores is desired for widespread use, there are problems. Some news stories may actually have several different possible classifications and choosing which one is determined by the individual.

During the “Human vs. Computer” testing session, a list and description of the categories was given to each of the 6 adults. The “rules” for classification were kept fairly simple, but this left room for the adults to choose perhaps different classifications in some
cases. For instance, for one person a headline about “Napster” would be considered Science/Technology while for another it may considered Entertainment. The same problem for the keyword Napster may be true for a keyword about “video games” as well. There also tended to be issues when there were headlines concerning the U.S. and Iraq that were sometimes classified as World headlines and sometime as U.S. headlines.

5.2.2 Unlearning a learner

One realization about the Bayes learning as it is implemented in the RSS Learner is that once the machine has learned to classify a certain way, it is hard for it to unlearn that way. When the RSS Learner for some reason or another was taught the wrong classification for the headline (putting keyword counts in the wrong table), it took two instances of each of those keywords in the right classification to effectively “unlearn” the mistake. When there is a small number of keywords in the database, it is easily corrected, although it may be more difficult with a large number of keywords in the database or with many keywords classified incorrectly.

During testing it was noticed that the adults sometimes changed how they classified certain keywords. They may have started classifying a keyword one way, but by the end of the testing session they were classifying it another way. The RSS Learner showed to be consistent. This predictability is good especially for testing, however it also raises a few issues:

1. The bias of the RSS Learner may be fine if only the person relying on correct classifications is the individual who trained it. For widespread use, one RSS Learner trained by one person is not sufficient.
2. Classifications change for some keywords. For example, the word “hurricane” may be associated with World news one month and U.S. news the next. Tracking the changes in the target concept involves more training and the ability to change quickly.

5.3 Project Evaluation

5.3.1 Limitations

During testing, there was a slight performance delay in the classification process. Once the program retrieved the RSS file, it took up to 30 seconds to classify all the headlines in that feed. Originally it the project was to incorporate more intelligence to assist in the area of keyword extraction such as using an thesaurus, morphing handling (e.g., to handle different tenses of the same verb). However, it proved to be enough work to implement the basic Bayes algorithm and create the user interface. Some of the classification time and the the storage requirement could have probably been reduced by adding these “extras”. [Salton 1989]

5.3.2 Storage requirements

The storage requirements for this project must consider that fact that the more training done, the more storage required. Additionally, if this project were continued for several more months, the keywords for the tables would only grow. However, the tables will not grow indefinitely, since there is a finite number of keywords in the English language.

5.3.3 Further research

Future research along these lines should no doubt incorporate several different learning algorithms into one system. Every algorithm often has its strength for a different scenario (or possible a different class of RSS files in this case). A different classifier may
converge on the target function (or learn the classification) faster in some cases. So, a sys-
tem that implemented several classifiers and some sort of weighted majority to determine
classifications would almost surely be more effective.

The only thing to keep in mind is that any system designed for online learning must 
be able to continue to learn. The Internet is evolving and changing constantly. The advan-
tage for Naive Bayes (and instance based learning such as K-Nearest Neighbor), is that 
they are more lazy than eager learners and training involves only adding instances to the 
database. For other methods, such as Neural Networks and decision trees, it may be a little harder to simply “add” training instances.

Online learning is an open field for the future and a paradigm that is in many ways very challenging. Hopefully, RSS files and other similar technologies (such as Seman-
ticWeb, [Berners-Lee 1998]) will help make such learning as effective as possible.
Books:


Journal or Conference Articles:


Web Sites:


APPENDIX A -- RSS (2.0) EXAMPLE

<?xml version="1.0" ?>
- <!-
  RSS generated by Radio UserLand v8.0.5 on 9/30/2002; 4:00:00 AM Pacific
-->
- <rss version="2.0"
  xmlns:blogChannel="http://backend.userland.com/blogChannelModule">
  <channel>
    <title>Scripting News</title>
    <link>http://www.scripting.com/</link>
    <description>A weblog about scripting and stuff like that.</description>
    <language>en-us</language>
    <blogChannel:blogRoll>http://radio.weblogs.com/0001015/userland/scriptingNewsLeftLinks.opml</blogChannel:blogRoll>
    <blogChannel:mySubscriptions>http://radio.weblogs.com/0001015/gems/mySubscriptions.opml</blogChannel:mySubscriptions>
    <blogChannel:blink>http://diveintomark.org/</blogChannel:blink>
    <copyright>Copyright 1997-2002 Dave Winer</copyright>
    <lastBuildDate>Mon, 30 Sep 2002 11:00:00 GMT</lastBuildDate>
    <docs>http://backend.userland.com/rss</docs>
    <generator>Radio UserLand v8.0.5</generator>
    <category domain="Syndic8">1765</category>
    <managingEditor>dave@userland.com</managingEditor>
    <webMaster>dave@userland.com</webMaster>
    <ttl>40</ttl>
  - <item>
    <description>"rssflowsalignright"With any luck we should have one or two more days of namespaces stuff here on Scripting News. It feels like it's winding down. Later in the week I'm going to a <a href="http://harvardbusinessonline.hbsp.harvard.edu/bu/b02/en/conferences/conf_detail.jhtml?id=5775tg&pid=144XCF">conference</a> put on by the Harvard Business School. So [...]</description>
    <pubDate>Mon, 30 Sep 2002 01:56:02 GMT</pubDate>
  </item>
</channel>
</rss>
<item>
  <pubDate>Sun, 29 Sep 2002 19:59:01 GMT</pubDate>
</item>
</channel>
</rss>
## APPENDIX B -- RSS LOG FILE EXAMPLE

<table>
<thead>
<tr>
<th>RSS Log File For November 21, 2003 14:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing begun November 21, 2003 14:50</td>
</tr>
<tr>
<td>'Feel Free to Jack Into My iPod'</td>
</tr>
<tr>
<td>'Ebay Sellers Generous with Junk'</td>
</tr>
<tr>
<td>'Witness to History: Inside Trauma Room One'</td>
</tr>
<tr>
<td>'Arrests made in wake of Istanbul terror blasts'</td>
</tr>
<tr>
<td>'Jackson already in debt'</td>
</tr>
<tr>
<td>'Rockets hit hotels, oil ministry in Baghdad'</td>
</tr>
<tr>
<td>'Rapper found guilty of murder, receives 21 years'</td>
</tr>
<tr>
<td>'Thanksgiving going to the dogs'</td>
</tr>
<tr>
<td>'2 wsu volleyball players earn all-league honors'</td>
</tr>
<tr>
<td>'Food intolerance and food allergy: A fishy business'</td>
</tr>
<tr>
<td>'Group Says Atkins Diet May Be Dangerous'</td>
</tr>
<tr>
<td>'Iraq insurgents displaying ingenuity (AP)'</td>
</tr>
<tr>
<td>'Note Suggests Sniper Juries Are Divided (AP)'</td>
</tr>
<tr>
<td>'Work on N. Korea nuke reactors suspended (AP)'</td>
</tr>
<tr>
<td>'Opponents block energy bill in Senate (AP)'</td>
</tr>
<tr>
<td>'Doctors Look at &amp;039;Chronic Daily Headaches&amp;039; (AP)'</td>
</tr>
<tr>
<td>'Jackson&amp;039;s Attorney Already in Spotlight (AP)'</td>
</tr>
<tr>
<td>'Dow industrials fall 6; Nasdaq climbs 5 (AP)'</td>
</tr>
<tr>
<td>'Cardinals trade Tino Martinez to D-Blue (AP)'</td>
</tr>
<tr>
<td>'Bush says Turkey New Front in &amp;039;war on terror&amp;039; (Reuters) &amp;039;'</td>
</tr>
<tr>
<td>'US detonates &amp;039;mother of all bombs&amp;039; in Florida test (Reuters)'</td>
</tr>
<tr>
<td>'Bush, Blair seal alliance with pub lunch (Reuters)'</td>
</tr>
<tr>
<td>'AARP accused of &amp;039;conflict of interest&amp;039; (usatoday.com)'</td>
</tr>
<tr>
<td>'Energy bill backers push ethanol plan (usatoday.com)'</td>
</tr>
<tr>
<td>'Bombers hit British targets in istanbul (washingtonpost.com)'</td>
</tr>
</tbody>
</table>

Testing ended November 21, 2003 14:57
number of unique headlines learned to date: 562
total headlines tested: 27
number correct: 9
percentage correct: 33
### APPENDIX C -- SAMPLE TESTING SHEET

**Tester:**

|----------|----------|------------|-----------------|---------|---------|-----------------------|

**Headline:** (Total headlines tested: 27)

<table>
<thead>
<tr>
<th>Headline</th>
<th>Enter the category:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feel Free to Jack Into My iPod</td>
<td></td>
</tr>
<tr>
<td>EBay Sellers: Generous With Junk</td>
<td></td>
</tr>
<tr>
<td>Witness to history: Inside Trauma Room One</td>
<td></td>
</tr>
<tr>
<td>Arrests made in wake of Istanbul terror blasts</td>
<td></td>
</tr>
<tr>
<td>Jackson already in debt</td>
<td></td>
</tr>
<tr>
<td>Rockets hit hotels, oil ministry in Baghdad</td>
<td></td>
</tr>
<tr>
<td>Rapper found guilty of murder, receives 25 years</td>
<td></td>
</tr>
<tr>
<td>Thanksgiving going to the dogs</td>
<td></td>
</tr>
<tr>
<td>2 WSU volleyball players earn All-League honors</td>
<td></td>
</tr>
<tr>
<td>Food intolerance and food allergy: A fishy business</td>
<td></td>
</tr>
<tr>
<td>Group Says Atkins Diet May Be Dangerous</td>
<td></td>
</tr>
<tr>
<td>Iraqi Insurgents Displaying Ingenuity (AP)</td>
<td></td>
</tr>
<tr>
<td>Note Suggests Sniper Jurors Are Divided (AP)</td>
<td></td>
</tr>
<tr>
<td>Work on N. Korea Nuke Reactors Suspended (AP)</td>
<td></td>
</tr>
<tr>
<td>Opponents Block Energy Bill in Senate (AP)</td>
<td></td>
</tr>
<tr>
<td>Doctors Look at 'Chronic Daily Headaches' (AP)</td>
<td></td>
</tr>
<tr>
<td>Theories Surround JFK Assassination (AP)</td>
<td></td>
</tr>
<tr>
<td>Jackson's Attorney Already in Spotlight (AP)</td>
<td></td>
</tr>
<tr>
<td>Dow Industrials Fall 6; Nasdaq Climbs 5 (AP)</td>
<td></td>
</tr>
<tr>
<td>Cardinals Trade Tino Martinez to D-Rays (AP)</td>
<td></td>
</tr>
<tr>
<td>Bush Says Turkey New Front in War on Terror (Reuters)</td>
<td></td>
</tr>
<tr>
<td>Guerrillas Fire Rockets at Fortified Baghdad Sites (Reuters)</td>
<td></td>
</tr>
<tr>
<td>US Detonates 'Mother of All Bombs' in Florida Test (Reuters)</td>
<td></td>
</tr>
<tr>
<td>Bush, Blair Seal Alliance with Pub Lunch (Reuters)</td>
<td></td>
</tr>
<tr>
<td>AARP accused of conflict of interest (USATODAY.com)</td>
<td></td>
</tr>
<tr>
<td>Energy bill backers push ethanol plan (USATODAY.com)</td>
<td></td>
</tr>
<tr>
<td>Bombers Hit British Targets in Istanbul (washingtonpost.com)</td>
<td></td>
</tr>
</tbody>
</table>

#correct

Percent correct