# Statistical Analysis of Online News

by Vahab Pournaghshband

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

**Committee:**

Professor Laurent El Ghaoui
Research Advisor

20th December, 2007

* * * * * * *

Professor Babak Ayazifar
Second Reader

20th December, 2007

## Abstract

Statistical Analysis on Online News

by

Vahab Pournaghshband

Master of Science in Engineering Electrical Engineering and Computer Sciences

University of California, Berkeley

Social scientists examining questions such as media bias, or the tracking the image of people, groups, or concepts in the news, often must manually sort through the enormous amount of news data available online. StatNews aims to create a tool for processing large collections of text articles and headlines, statistically analyzing the retrieved data sets, and visualizing the results of the analysis. In particular, the idea is to come up with a new tool for extracting information from data sets with many variables. Such a tool, developed based on computational techniques, could be of interest not only to social scientists, but also to online news readers everywhere who are inundated with news items from many sources. We also believe that this will help spark a renewed interaction between applied mathematics and social sciences.

*"An abstract version"* of the StatNews has been developed (analysis, design, an implementation) here to smooth the road for the construction of the final, detailed version of the project. The main goal has been to figure out obstacles in advance, and well before the actual scaled project run into. Among specific questions in mind was to figure out

whether a penalized logistic regression is superior over simple frequency count in results or not.

## Acknowledgments

I am truly grateful to everyone who has directly and indirectly helped me during the past year. First and foremost, I would like to thank my advisor, Professor Laurent El Ghaoui, for giving me a chance to work with him, guiding me through the last year and teaching me how to be a good researcher. I, also, would like to thank the members of the StatNews group and the UC Berkeley EECS Department. Finally, I am deeply grateful to my parents, for their continuous supports and encouragements.

# Table of Contents

I. Preliminaries……………………………………………………………………7

1. Introduction………………………………………………………………7

1.1 Motivation…………………………………………………………7

1.2 Problem Statement and Challenges…………………………………...8

1.3 Outline of Report…………………………..............................8

2. Background……………………………..……………………….………..9

2.1 Document Representation……………………………….................9

2.1.1 Inverted Files………………………………………...................9

2.1.2 Suffix Arrays……………………………..……................11

2.2 Regression Models………………………….…….……................13

2.2.1 Linear Regression………………………….….……................13

2.2.2 Non-linear Regression…………………………………............14

2.2.2.1 Linearization………………………………..................14

2.2.2.2 Logistic Regression……………………….…….…......15

II. Architecture/Design……………………………................................................16

1. Overview…………………………………………………….....................16

2. Fetching……………………………………………………….……................17

3. Data Storage………………………….....................................................18

3.1 Term Selection……………………………………………................18

3.1.1 What is a Term? ……………………………………...............18

3.1.2 Stemming……………………………………….…................19

3.1.3 Stop Words……………………………….......................21

3.1.4 Infrequent words……………………………….............22

3.2 Vector space…………………………….................................23

3.2.2 Frequency Vector…………………….……….............23

3.2.3 Boolean Vector……………………………….................24

3.3 Storage Format……………………………..............................25

3.3.1 Frequency/Co-occurrence Matrix……………….………25

3.3.2 Market Matrix Exchange……………….………………25

4. Statistical Analysis…………………………….......................................27

4.1 Frequency Count………………………………….……................27

4.2 Penalized Logistic Regression…………………………….........28

5. Visualization……………………………................................................30

III. Implementation Details……………………………………...……................32

IV. Conclusions……………………………............................................................33

Bibliography…………………………….............................................................................34

# I. Preliminaries

# 1. Introduction

## 1.1 Motivation

Today, a vast array of data is publicly available on the Internet, leading to the impression that ”information is at our fingertips.” However, today’s Internet tools, such as search engines, usually only provide a very local view of the data, in response to a specific query. Equipped with microscopes, we find needles in the haystack, but we still know very little about the shape of the haystack. A case in point is news media. While we are inundated with news items from many sources refreshed constantly, it is hard to get a global view of information, especially across different sources and over entire spans of time. Certainly, RSS (Real Simple Syndication) feeds allow the user to assemble a ”sample” as desired. However this sample is a very limited instance of an aggregate view: it provides only a custom, microscopic focus, at a specific point in time.

Assume for example that one wishes to illustrate the image of a concept such as ”oil” in the Reuter’s data over a specific period of time. Our goal is to allow that person to understand the few words that are most statistically correlated with the concept at hand, and more generally, build a graphical model of the underlying news process.

We believe that many citizens should greatly benefit from an easy access such aggregate views. For the scientists, the analysis of publicly available Internet data, or web data analysis for short, is a ”new frontier” in the sense that it could lead to a renewed, very fruitful interaction between mathematics and social sciences.

In turn, many of the tools developed for online news analysis will be helpful to many other scientific areas. For example, many medical databases are now online, and they share many features with news databases: they are heterogeneous, with different structures and different types of content, from text to numerics to images; most are very large, and distributed; and they are dynamically changed[4].

## 1.2 Problem Statement and Challenges

While the StateNews seeks for more generalized answers to related questions (e.g. network of concepts, the network of news sources, analysis of voting patterns etc), we have concentrated on designing and developing a tool that would let us to see the image in the form of a graph for a given word or concept in the news. The approach is based on obtaining a local graphical model, where the key word at hand is shown, together with highly statistically related words. A key requirement here is that the graph should be sparse and be both statistically reliable and interpretable for the user. The result is expected to show the strength of the correlations between concepts inferred from the data set.

## 1.3 Outline of the report

After this section that we stated the problem and challenges, this report will continue with establishing necessary background that is required for later discussions. Following the background, the report then provides the details of the architecture and design behind this project. Finally, a brief discussion on implementation details and conclusion will end the report.

# 2. Background

## 2.1 Document Representation

In most massive data manipulation applications such as search engines, after retrieving the information from the web pages, an important question is how to store this information to be able to quickly search through it to find a particular term. Inverted File Index has been extensively used for such applications.

### 2.1.1 Inverted File Index

An index is a scheme for locating a given term in a text. An inverted file contains, for each term in the lexicon, an inverted list that stores a list of pointers to all occurrences of that term in the main text, where each pointer is, in effect, the number of a document in which that term appears. Also these words in the inverted file are lexicographically sorted to allow for a fast binary search for words in question. Note that frequency of the term occurrence in each document is not explicitly given here, but may easily be computed given the fixed size of the document. We now illustrate this idea in the following example:

Assume the following documents with corresponding contents reside in our corpus.

| Document | Text |
|---|---|
| 1 | UC Berkeley Computer Science Department |
| 2 | Computer science is a science of information process |
| 3 | I have a computer |

Then we build the corresponding index.

| Number | Term | Times; Documents |
|---|---|---|
| 1 | a | <1; 2> |
| 2 | Berkeley | <1; 1> |
| 3 | computer | <3; 1,2,3> |
| 4 | department | <1; 1> |
| 5 | have | <1; 3> |
| 6 | i | <1; 3> |
| 7 | information | <1; 2> |
| 8 | is | <1; 2> |
| 9 | of | <1; 2> |
| 10 | process | <1; 2> |
| 11 | science | <2; 1,2> |
| 12 | UC | <1; 1> |

Then if we want to find “computer science”, we can easily find document 1 and 2 contain these two words.

A word-level inverted file index can record where the word occurs in the document.

| Number | Term | Times; Documents Words |
|---|---|---|
| 1 | a | <1; (2;3)> |
| 2 | Berkeley | <1; (1;2)> |
| 3 | computer | <3; (1;3),(2;1),(3;4)> |
| 4 | department | <1; (1;4)> |
| 5 | have | <1; (3;2)> |
| 6 | I | <1; (3;1)> |
| 7 | information | <1; (2;7)> |
| 8 | is | <1; (2;3)> |
| 9 | of | <1; (2;6)> |
| 10 | process | <1; (2;8)> |
| 11 | science | <2; (1;3),(2;5)> |
| 12 | UC | <1; (1;1)> |

### 2.1.2 Suffix Arrays

Suffix array is a data structure designed for efficient searching of a large text that preserves consecutive appearances of terms. The data structure is simply an array containing all the pointers to the text suffixes sorted in lexicographical order. Each suffix is a string starting at a certain position in the text and ending at the end of the text. Searching a text can be performed by binary search using the suffix array. Now, we illustrate the idea with constructing a suffix array for "UCBerkeley".

First, we should assign index points to the sample text. Index points specify positions where search can be performed. In our example, index points are assigned character by character. Thus, we can search the sample text with the suffix array at any positions later.

| Text | U | C | B | e | r | k | e | l | e | y |
|---|---|---|---|---|---|---|---|---|---|---|
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Second, we should sort the index points according to their corresponding suffixes. The correspondence between the index points and the suffixes looks like:

| Suffix | | | | | | | | | | Index |
|---|---|---|---|---|---|---|---|---|---|---|
| U | C | B | e | r | k | e | l | e | y | 0 |
| | C | B | e | r | k | e | l | e | y | 1 |
| | | B | e | r | k | e | l | e | y | 2 |
| | | | e | r | k | e | l | e | y | 3 |
| | | | | r | k | e | l | e | y | 4 |
| | | | | | k | e | l | e | y | 5 |
| | | | | | | e | l | e | y | 6 |
| | | | | | | | l | e | y | 7 |
| | | | | | | | | e | y | 8 |
| | | | | | | | | | y | 9 |

After sorting it becomes:

| Suffix | | | | | | | | | | Index |
|---|---|---|---|---|---|---|---|---|---|---|
| | | B | e | r | k | e | l | e | y | 2 |
| | C | B | e | r | k | e | l | e | y | 1 |
| | | | | | | e | l | e | y | 6 |
| | | | e | r | k | e | l | e | y | 3 |
| | | | | | | | | e | y | 8 |
| | | | | | k | e | l | e | y | 5 |
| | | | | | | | l | e | y | 7 |
| | | | | r | k | e | l | e | y | 4 |
| U | C | B | e | r | k | e | l | e | y | 0 |
| | | | | | | | | | y | 9 |

Finally, the resulting index points become the suffix array for the sample text.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 6 | 3 | 8 | 5 | 7 | 4 | 0 | 9 |

## 2.2 Regression Models

Regression models are used to predict one variable from one or more other variables. Regression models provide the scientists with a powerful tool, allowing predictions about past, present, or future events to be made with information about past or present events.

### 2.2.1 Linear Regression

linear regression is a regression method that models the relationship between a dependent variable *Y*, independent variables $X_i$, $i = 1, ..., p$, and a random term $\varepsilon$ which is also known as noise of the model. The linear regression model can be written as

$$Y = \theta_0 + \sum_{i=1}^{p} \theta_i X_i + \varepsilon$$

where $\theta_0$ is the intercept, the $\theta_i$s are the respective parameters of independent variables, and *p* is the number of parameters to be estimated in the linear regression.

This method is called linear due to the fact that the relation of the response (the dependent variable *Y*) to the independent variables is assumed to be a linear function of the parameters.

The linear regression model can be written in vector-matrix notation as

$$Y = \vec{X}\vec{\theta} + \varepsilon$$

where, again, the term $\varepsilon$ is the model's error term (a misnomer but a standard usage) and represents the unpredicted or unexplained variation in the response variable; it is conventionally called the error whether it is really a measurement error or not, and is assumed to be independent of $X$.

### 2.2.2 Non-linear Regression

Nonlinear regression is the problem of inference for a model

$$y = f(x, \theta) + \varepsilon$$

based on multidimensional $x$,$y$ data, where $f$ is some nonlinear function with respect to unknown parameters $\theta$. At a minimum, we may like to obtain the parameter values associated with the best fitting curve (usually, least squares).

#### 2.2.2.1 Linearization

Some nonlinear regression problems can be linearized by a suitable transformation of the model formulation. For example, consider the nonlinear regression problem (ignoring the error):

$$y = ae^{bx}$$

If we take a natural logarithm of both sides, it becomes

$$\ln(y) = \ln(a) + bx$$

suggesting estimation of the unknown parameters by a linear regression of *ln(y)* on *x,* a computation that does not require iterative optimization. However, use of linearization requires caution. The influences of the data values will change, as will the error structure of the model and the interpretation of any inferential results. These may not be desired effects.

### 2.2.2.2 Logistic Regression

In statistics, logistic regression is a model used for prediction of the probability of occurrence of an event. It makes use of several predictor variables that may be either numerical or categories. For example, the probability that the word oil appear in a document given that Iran and OPEC appeared in the same document. Logistic regression analyzes binomially distributed data of the form:

$$Y_i \sim B(n_i, p_i),\ for\ i = 1,\ldots,m$$

where the numbers of Bernoulli trials $n_i$ are known and the probabilities of success $p_i$ are unknown. The logits of the unknown binomial probabilities (*i.e.*, the logarithms of the odds) are modeled as a linear function of the $X_i$.

$$\log it(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \theta_0 + \sum_{j=1}^{p} \theta_j x_{i,j}$$

The interpretation of the $\theta_j$ parameter estimates is as the additive effect on the log odds ratio for a unit change in the *j*-th explanatory variable.

# II. Architecture/Design

In this section we give a detailed design of the project. Please note that the Term Selection and Vector Space discussed here are similar to SONIA project at Stanford described by Mehran Sahami [2].

## 1. Overview

The StatNews project consists of four components: Fetching, Data Storage, Statistical Analysis, and Visualization as illustrated in figure 1.

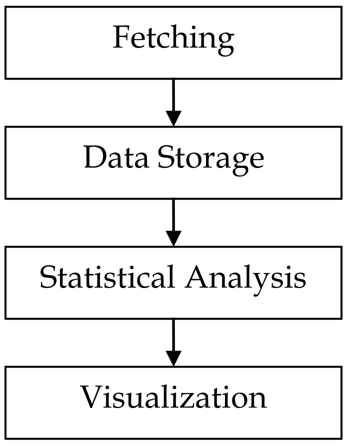

Figure 1. Components of the StatNews Project

*Fetching* attempts to gather useful information from online sources (a form of information retrieval), *data storage* stores the information in a structure that preserves important information needed later to find correlation between terms. *Statistical analysis* performs statistical analysis to find such correlations, and finally *visualization* aims to visualize the results (e.g. correlation between terms) in a comprehensible manner.

## 2. Data Fetching

The role of the data fetching is to retrieve useful information from online corpora needed for further correlation calculation. While the desired fetching tool is intended to do this automatically (web crawler[1]), this was done manually here by writing an HTML/XML parser to collect useful information such as headlines from the offline access to all Reuter's articles published in 1997 and 1998. The headlines are then stored in a CSV (Comma Separated Value) file in the format described in figure 2.

```
DOC_ID_1, [headline of document 1]
DOC_ID_2, [headline of document 2]
DOC_ID_3, [headline of document 3]
...
```

Figure 2. Output Format of the Fetching Process

Note that the document ID (DOC_ID) contains information specific to the article as well as the timestamp. Also, the headlines are chronologically ordered. This is mainly useful when we decide to examine how an image of a concept has evolved over time.

# 3. Data Storage

The data storage component refines the crude data gathered by the fetching component and converts it to a structured format for further analysis. As mentioned earlier, it is crucial for the designed structure to preserve information that is later used to derive statistical conclusions.

## 3.1 Term Selection

### 3.1.1 What is a Term?

Terms are the atomic units of our project that attempts to find interesting correlations between terms. However, before any further manipulations we need to clearly define the concept of term.

The most common definition of a term in English is that a term is a sequence of alpha-numeric characters which it delimited by white space (spaces, tables, new-line characters etc) or punctuation marks (such as a comma or exclamation mark). In addition, all uppercase letters in a document are converted to lowercase, so effectively capitalization is ignored. That is why as the first step in the implementation I implemented filters that remove numbers, punctuations, and convert the text to lowercase with the goal of extracting terms from the text.

Input to the Term Extraction:

```
I want to know God's thoughts... the rest are details. Albert Einstein 1879-1955
```

After Terms Extraction:

```
I want to know God s thoughts the rest are details Albert Einstein
```

### 3.1.2 Word Stemming

In most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of information retrieval applications. The word stemmers attempt to reduce a word to its stem or root form. Thus, the key terms of a document are represented by stems rather than by the original words. This not only means that different variants of a term can be conflated to a single representative form (for instance, “writing, write, writer, and wrote” all are treated as only one word “writ”), it also reduces our dictionary size, that is, the number of distinct terms needed for representing a set of documents.

Numerous stemmers have been developed over the past years, but the one we used specifically was Porter stemmer[11] which is a conflation stemmer developed by Martin Porter in 1980. The main reason Porter stemmer was selected is for its linear time complexity. However, the Porter stemmer is known to overstem the words. Overstemming is an error where two separate inflected words are stemmed to the same root, but should not have been. For this, lemmatization seems a more promising approach.

*Lemmatization:*

A more complex approach to the problem of determining a stem of a word is lemmatization. This process involves first determining the part of speech of a word, and applying different normalization rules for each part of speech. The basic idea is that, if we are able to grasp more information about the word to be stemmed, then, we are able to more accurately apply normalization rules (which are, more or less, suffix stripping rules) and prevent errors such as overstemming. However, this is too complicated for our short-run purposes and worse, very computationally expensive, and since this abstract-level project is desired to be simple, lemmatization was not used here.

Input to the Porter Stemmer:

```
Great works are performed, not by strength, but by perseverance.
```

After Porter Stemmer:

```
Great work ar perform not by strength but by persever
```

### 3.1.3 Stop Words

These are words such as prepositions, conjunctions, and pronouns that are used to provide structure in language rather than content. Such words are commonly used in documents regardless of topic, and thus have no topical specificity. As a result, we can eliminate such words from the documents, as they will hold almost no correlation with the desired term. Examples of such words are listed in figure [3]. We used the stop words list from an online source [14] in addition to our stop words. Our list contains a total of 614 stop words.

```
all
allow
allows
almost
alone
along
alongside
already
also
although
always
an
and
another
any
anybody
anyhow
```

Figure 3. A Sample List of Stop Words

Input to the Stop Words Remover:

```
Whatever the mind can conceive and believe, the mind can achieve.
```

After Eliminating the Stop Words:

```
mind conceive believe mind achieve
```

### 3.1.4 Infrequent Words

There are many words that appear in the corpus very infrequently. Since our goal is to identify correlation between terms that appeared often in the documents, then words which only appear, say, one or twice (or generally infrequently) in the collection do not describe significant relationship with frequent terms and can be ignored. Also, since the term frequency plot of the entire corpus is smooth (as an example of such plot, refer to figure 4) setting a dynamic threshold is not easy. For this reason, a static threshold of 10 was set, and as a result, the size of our dictionary was reduced from 4355 to 903 terms.

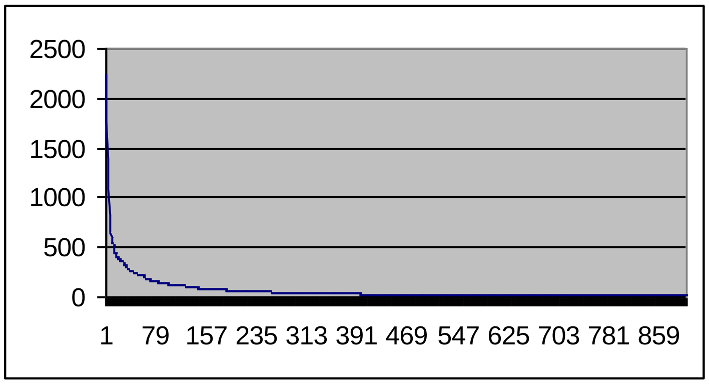

Figure 4: Plot of All Terms Co-occurred with the Term “oil” in Reuters’s 97-98 (terms vs. frequency)

## 3.2 Vector space

At this stage anything survived the preprocessing filters (discussed in section 3.1) is considered an important term and will be added to the dictionary and will become a dimension of the document vectors. We use the vector space representation for document commonly employed many information access problems[7].

In our vector space representation, each document is characterized by the Boolean or numerical vectors. These vectors are embedded in a space in which each dimension corresponds to a distinct term in the corpus of documents being characterized. As we will show later, these vectors will then be used to build interesting matrices.

### 3.2.1 Frequency Vector

A frequency vector defines each document. The first element (*0*-th) contains information about the document (e.g. document ID, timestamp etc) and for the remaining elements, the *i*-th element represents the function $f(\zeta(t_i,d))$, where $\zeta(t_i,d)$ is the number of times term $t_i$ appeared in the document *d*. The function *f* could have many definitions. The simplest form is the identity function that records the frequency of each term in the dictionary appeared in the document *d*. However, more interesting functions are as follows[3,8,9]:

$$f(a) = \log(a+1)$$
$$f(a) = \sqrt{a}$$
$$f(a) = \frac{a}{a+\beta}$$
$$f(a) = TFIDF(a) = a \cdot IDF(\frac{N}{n_t})$$

where in the last formula, *N* is the total number of documents and $n_t$ is the number of documents in which term *t* appears at least once.

### 3.2.2 Boolean Vectors

In this case, we are only interested to see whether the term $t_i$ appeared in the document *d*, regardless of how many times it actually appeared. Therefore, we use the following function $f(\zeta(t_i,d))$

$$f(a) = \begin{cases} 1 & a \geq 1 \\ 0 & otherwise \end{cases}$$

Where, as for the frequency vector, $a = f(\zeta(t_i,d))$.

## 3.3 Storage Format

### 3.3.1 Frequency/Co-occurrence Matrix

When storing the data, a common way to ensure preserving the important information is to store them into a matrix.

These matrices are basically a collection of the vectors discussed in the section 3.2, where each row is a vector where represents a document. For frequency matrix, the value stored in $a_{ij}$ (assume $i,j > 1$) is number of times the $j$-th term appeared in the $i$-th document.

This is similar for the co-occurrence matrix, with one difference that the value stored in $a_{ij}$ is Boolean. That is, $a_{ij}$ is 1 if the $j$-th term appeared at least once in the $i$-th article and 0 otherwise. Figure 5 illustrates how such matrix is stored in a file.

```
file_name   , [term_1]        , … , [term_n]
[document_1], [term_1 exists?], … , [term_n exists?]
[document_2], [term_1 exists?], … , [term_n exists?]
…
```

Figure 5. How Co-occurrence Matrices Stored in a File

### 3.3.2 Matrix Market Exchange

The frequency and the co-occurrence matrices can potentially become gigantic as the number of documents and the size of the dictionary increases. This overly large size can create subtlety by consuming a large chunk of memory as well as slowing the process of computation due to high I/O cost.

On the other hand, we expect that these matrices to be considerably sparse and contain many zeros. This is intuitive since not all the terms in the dictionary would appear in the same document. This motivates us to use Matrix Market Exchange Format which has been primarily designed to store sparse matrices. The Matrix Market (MM) exchange formats provide a simple mechanism to facilitate the exchange of matrix data. MM is a file format suitable for representing general sparse matrices since only nonzero entries are provided, and the coordinates of each nonzero entry is given explicitly. This is illustrated in the following example of a real 3×3 general sparse matrix.

| | | |
|---|---|---|
| 0 | 0.25 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 2 |

(a)

```
0,1,0.25
1,0,1
0,2,2
```

Figure 6. (a) Sparse Frequency Matrix (b) The Corresponding MM Stored in a File

# 4. Statistical Analysis

Our goal at this stage is to draw statistical conclusions. We have used two tools to achieve this: (1) by counting the number of times a term in the dictionary appeared, and (2) by solving for the parameters of a penalized logistic regression. In both cases, we will think of words in articles or headlines as binary random variables with a value of 1 if the word is present and 0 if the word is absent.

## 4.1 Frequency Count

Frequency count is done by just counting of the number of times a term A appeared along with a specific term B in mind in the entire corpus. By taking frequency counts, we would hope to find answer to questions like: are words A and B correlated? Are they anti-correlated? Figure 7 show the top 10 most frequent terms appeared with the term “oil” in the Reuters’s data set. One could draw conclusions based on this observation.

| Term | Frequency |
|---|---|
| Price | 2229 |
| Gas | 1781 |
| Product | 1373 |
| NWE (North West Europe) | 1274 |
| Fuel | 1098 |
| Palm | 756 |
| Crude | 646 |
| Steady | 541 |
| Iraq | 521 |
| Export | 498 |

Figure 7. The Top 10 Most Frequent Terms Appeared with the Term "oil" in the Reuters's Data Set.

## 4.2 Penalized Logistic Regression

Our goal is to find estimation for parameters of a penalized logistic regression that best fits our data. The problem formulation for this optimization problem is as follows (this is presented in Wainwright et al [13] which was initially developed by Boyd et al:

Let $G(V,E)$ denote a graph with vertex set $V$ of size $|V|=p$ and edge set $E$. $p$ is considered to be the number of terms in the dictionary. Also, denote the set of neighbors of a vertex $v$□$V$ as $N(s)$: $N(s) = \{(s,v)$ □ $E\}$. Given $n$ samples $x^{(i)}$ □ $\{0,1\}^p$ drawn from an unknown distribution, we consider the problem of estimating neighborhoods $N_n(s)$□$V_n$ so that Pr[ $\check{N}(s) = N(s)$ , *for all* $s$ □ $V_n\}$] → 1. Our goal is to use ℓ1-reguralized logistic regression to estimate these neighborhoods as well as the actual values of the parameters $\theta_{i,j}$ as a secondary concern which can be achieved by the following collection of optimization problems:

$$\hat{\theta}^{s,\lambda} = \arg\min_{\theta\in\Re^p}\left\{\frac{1}{n}\sum_{i=1}^{n}[\log(1+\exp(\theta^T z^{(i,s)})) - x_s^{(i)}\theta^T z^{(i,s)}] + \lambda_n \parallel \theta_{\backslash s} \parallel_1\right\}$$

Where $s$□$V$, $\theta_{\backslash s}$ is the vector of all coefficients of $\theta$ except the one in position $s$. and the quantity $\theta_t^{s,\lambda}$ can be thought of as a penalized conditional likelihood estimate of $\theta_{t,s}$. The estimate of the neighborhood $N(s)$ is the given by

$$\hat{N}_n(s) = \{t \in V, t \neq s : \hat{\theta}_t^{s,\lambda} \neq 0\}$$

This optimization problem has desirable statistical properties. That is, when we use some data (say 14000 headlines) to estimate the undirected graphical model, what we get is only an estimate. It means our estimation of the parameters might not be correct because of randomness in the data. This corresponds to random events happening in the world and random word choices by reporters at newspapers. But suppose we could collect an infinite number of articles or headlines. Then, by solving these logistic regression problems, we would recover the correct undirected graphical model exactly. This is called asymptotic consistency. Statisticians consider this to be a desirable property in an estimator. Wainwright et al. [13] showed that using logistic regression to estimate the undirected graphical model is asymptotically consistent. Moreover, Wainwright et al. showed that this is true even if we have an infinite number of words in our dictionary.

Basically we are primarily interested in the neighbors of $s$, the query term, which are correlated terms with $s$ (i.e. their estimated $\theta$'s are non-zero). $\theta$'s also be thought as the correlation strength between of the terms t and s. As shown in the optimization problem statement, the penalty term forces to minimize the number of non-zero $\theta$'s, leaving only the important terms as the neighbor of $s$.

## 5. Visualization

The last step is to visualize our result in a form of undirected graphs once the correlations are calculated. This is achieved by considering all $p$ terms in the dictionary as nodes of such graph. An edge, or even more preferably weighted edges, represents correlations between the terms in the dictionary. The big picture, the undirected graphical model (also known as independence graph[6]), allows the user to answer questions about all interrelationships among the $p$ words in the dictionary.

To visualize the result of the penalized logistic regression, we draw an edge between terms $i$ and $j$ if $j$ is in the set of neighbors found for terms $i$ AND $i$ is in the set of neighbors found for word $j$; in this case, we would conclude terms $i$ and $j$ are correlated (the concept of neighbors is defined in section 4.2). Alternatively, instead of "and" in the step above, "or" could be used.

For simple frequency count, however, the edge mean whether the terms $i$ and $j$ have appeared frequently together. The weight (length) of each edge illustrated the relative correlation between the terms (the closer they are, the stronger their relationship is). Figure 8 depicts the graph for the term "oil" in the Reuter's data set.

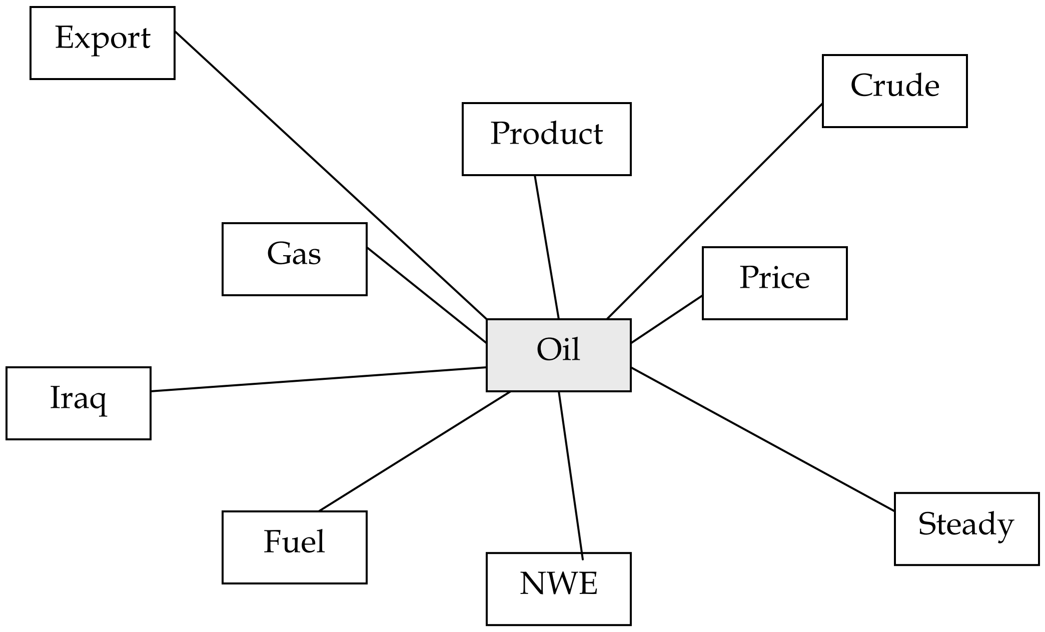

Figure 8. The Visual Result “oil” in the Reuter’s Data Set

# III. Implementation Details

*Programming Language*

Python has been used as an implementation tool for this project. It was primarily due to the fact that Python is a script language and can be used to implement both standalone, for offline users, and web-based applications for online users. Furthermore, Python's core syntax and semantics are minimalist the standard library and comprehensive. Also, Python is a multi-paradigm programming language (primarily functional, object-oriented, and imperative) which has a fully dynamic type system and uses automatic memory management.

*Compatibility*

The Python modules have been implemented in a way that they are compatible with both Windows and Macintosh Operating Systems.

# IV. Conclusions

StatNews has aimed to create a tool for processing large collections of data sets and visualizing the results of the analysis. The idea, in particular, has to come up with a new tool for extracting information from data sets with many variables.

The details and results of the abstract version of the StatNews which has been developed (analysis, design, an implementation) in the past few months was provided in this report. To reemphasize, the main goal of this abstract version has been to figure out obstacles in advance, and well before the actual scaled project run into. Among specific questions in mind was to figure out whether a penalized logistic regression is superior over simple frequency count in results or not. Finding answers for many questions of this type throughout the design and implementation of this project have made our effort successful.

# Bibliography

[1] Manning, C., Raghavan, P., and Schütze, H., *Introduction to Information Retrieval*, Cambridge University Press. 2008.

[2] Mehran Sahami, PhD Dissertation, *Using Machine Learning To Improve Information. Access*, Stanford University, 1998.

[3] Van Rijsbersgen, C. J. *Information Retrieval*, Butterworths, 1979

[4] El Ghaoui, L., Statistical Analysis of Online News (Miller Professional Proposal)

[5] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Francisco, CA, 1988

[6] Minka T., *Notes on Independence Diagrams*

[7] Salton, G., Wong, A., and Yang, C. S. A vector space model for automatic indexing, *Communications of the ACM 18* (1975), 613-620.

[8] Roberstson, S. E., and Sparch-Jones, K. Relevance weighting of search terms. *Journal of the American Society of Information Science 27* (1976), 129-146

[9] Salton, G., and Buckley, C. Term weighting approaches in automatic text retrieval. *Information Processing and Management 24,5* (1988), 513-523.

[10] Cutting, D. R., Karger, D. R., Pederson, J. O., and Tukey, J. W. Scatter/Gather: a cluster-based approach to browsing large document collections. In *Proceedings of ACM/SIGIR (1992)*, pp. 318-329.

[11] Frakes, W. B. *Stemming algorithms in Information Retreival: Data Structures and Algorithms*, W. B. Frakes and R. Baeza-Yates, Eds. Prentice Hall, 1992, pp. 131-160.

[12] Peter J. Bickel and Elizaveta Levina. Regularized estimation of large covariance matrices. *To appear in Annals of Statistics*, 2006.

[13] Wainwright, M., Ravikumar, P., and Lafferty, J., High-dimensional graphical model selection using L1-regularized logistic regression. *Proceedings of Advances in Neural Information Processing Systems*, 2006.

[14] *http://www.webconfs.com/stop-words.php*