Buzzspotting

Jonas Lindmark       Claes Mogren
tfy99jlk@cs          tfy99cmo@cs

9th June 2004

Abstract

What topics that are popular in the media varies quickly and it’s not easy to recognize trends or know what was the most reported topic at a certain date. When having a database containing news articles for every day it is interesting to be able to perform searches over time on the data. It would be possible to look for correlations between words or follow the number of times a so called “buzzword” have been mentioned and use that as a measure of popularity.

1 Introduction

The database in question is a collection of news articles from several different sites. Currently about 32000 articles spanning over the last 120 days. To visualize the information from the set of data we planned to generate a graph showing the number of articles that have been found and how many times the word have been mentioned in the articles over a period of time.

Initially our plan was to make it possible to detect correlations between pairs of buzzwords in the same article by using boolean search commands. The possibility to limit the search on a subset of the available news sources were another wanted feature. Furthermore, because all data was already present, a clickable graph image were planned that generated web links to the home pages to each article for the day clicked.

If the frequency of a certain word suddenly goes up quickly it could signify that some big event have occurred. The best way to calculate this would probably be to keep count of the frequencies of all words in all articles for each day, then compare the days.

2 Approach

The choice of using the MySQL database was based on two reasons; first that the current database was using MySQL and secondly the good indexing functions\(^1\) of that database. To get the full power of this indexing function we used the 4.1.1-alpha version of MySQL. Since the tables needed to extract the data were already there, no time was spent on designing the database.

\(^1\)MySQL FULLTEXT index
To build a simple user interface we used PHP in combination with HTML. The main reason for that was the simplicity of connecting PHP with a database and with other programs.

The graphical plots of the data was generated by calls to GnuPlot. The plot should be able to present the number of articles found each day, and how many times the sought word was found in those articles. GnuPlot can quickly generate images in many formats and is very suitable to plot graphs over time.

In case the search page would be integrated with the news page where the data is collected the designed was kept simple and using the same style sheets as the original page².

When a user wanted to limit the search to just a few sites, some check boxes below the search field had to be checked. When all of them were cleared, all sites were searched. In the same way, the user can expand the search to include the total amount of times the word has been mentioned in all articles in addition to just the number of articles.

Figure 1: The search page displaying a search on the word “bush” on all sites with both the number of articles and the number of words counted.

²http://senaste.info
3 Results

Despite the fact that the home page was set up on an old computer with 800MHz AMD processor, most queries were reasonable fast. The most complex search took about four minutes to do when going through about 32000 articles. The majority of one word searches took about 2-3 seconds. An example of a typical search is shown in figure(1). By limiting the query to just count the articles the search can be performed even faster. It was surprisingly easy to locate when major events occurred as long as you knew a word that is connected with the event. The figures below show the difference when between two searches that just differ in that one have + and the other - in front of “dominikanska”. The spike in the graph mark when a storm came in over the Caribbean region, resulting in many articles mentioning both countries.

![Graph showing search results](image1)

Figure 2: The graphs show the number of articles found for the searches “+haiti +dominikanska” and “+haiti -dominikanska”. Note that the scale of the y-axis is six times larger on the first search.

After some additional suggestions the search page was modified to incorporate the possibility to show in what percentage of articles the given word was found. Since the total amount of articles varies a lot from day to day, the percentage search gives a more unbiased display of the importance of the buzzword for each day. An example of this is is when something occur on weekends when the newsflow in general is lower.

![Graph showing search results](image2)

Figure 3: Comparing a search on the word “golf” with a calendar reveals that most articles containing the word are published on weekends.
The type of queries that takes very long to perform is the ones using the wild card character *. This can be used to find articles who include the search word and all the inflections of it. A search for “a*” took about four minutes and resulted in the following graph.

Figure 4: The first graph shows a search for “a*”. Assuming most articles contain some word starting with an “a”, it gives a fair estimate of the number of articles searched each day. The other graph displays the result search for “+Persson +häft*” and gives you a good hint on when his hip surgery was done.

4 Conclusions

The current implementation is lacking two major features. The first one is that the graph is not clickable, so there is no way to access the articles that generated the count for that particular day. Since the URLs for all articles are available after the search, all that’s needed is a way to map each point in the graph to the corresponding day in the search result. The idea to make hyper links of the graph results came a bit too late to get implemented.

Another limitation is that the “word of the week” is not calculated. It was a lot more work to calculate this than we first thought. It is needed to keep track of the frequency for all words written every day for as long as the database has been running. The table generated by this count will be very large and needs good database design and clever queries. We spent quite some time trying to implement this in an effective way, but unfortunately the result was not good enough to be useful.

A feature that we really felt was missing in MySQL was the ability to use “views” to reduce the amount of data each query had to go through. This will supposedly be available in version 5 of MySQL, and would both simplify and speed up many queries.

To make the searches as efficient as possible it is done almost completely in the database. PHP is only used for parsing queries and to call GnuPlot. Because of that the speed of a search depends completely on how fast the database can perform the query, and the memory requirements are quite low.

It would be possible to speed up the searches even further by creating additional tables that hold, for instance, the total amount of articles for each news source and day. That would speed up especially the percentage searches a lot.