Trend Analysis in Polls, Topics, Opinions and Answers

Gary Benattar, Philippe Trébuchet

Trend Analysis in Polls, Topics, Opinions and Answers

Gary Benattar* and Philippe Trebuchet
g.benattar@gmail.com, philippe.trebuchet@lip6.fr
Department of Computer Science
University of Paris 6 - Jussieu, Paris, France

June 20, 2011

Abstract

The goal of this work was to extract polls and opinions trends from a very large corpus. We developed a system that performs single words trend detection and also detects collocation candidates as well as collocation trends. We conducted experiments on a real-life repository, made available to us by Toluna.com, and counts around 23 millions documents. Our contributions were twofold: (1) we characterized the problem and chose the appropriate measures (eg z-score) to identify trends, (2) we designed a fully scalable system, leveraging Hadoop [1], Zookeeper [2] and Hbase [3]. This system includes a visual interface that displays a timeline highlighting trends in a 2-year window, as well as statistical reports by single words and n-grams.

*M.Sc. project in software engineering under the supervision of *Dr.* Philippe Trebuchet. Laboratory of Computer Science, Paris 6 (*LIP6*) *APR* (Algorithms, Programs and Resolutions) team

[1] http://hadoop.apache.org/

[2] http://zookeeper.apache.org/

[3] http://hbase.apache.org/

Contents

List of Algorithms

List of Figures

List of Tables

1 Introduction

1.1 What is a trend?

This work focuses on analyzing trends on a popular market research Web site, *toluna.com*. *Toluna*, is the world's largest online market research panel and online survey software provider to the global market research industry. Providing online market research sample and online research panel building expertise to over 1,500 of the world's leading market research agencies.

Toluna.com is a Web community site focusing on polls and opinions of people. Our mission is to allow anyone to know what the world thinks on any issue and foster a community of insight through the use of Web2.0 technology. Each poll created has value in itself; it is a snapshot of the world's opinions and as a result has tremendous value for community members interested in a particular topic.

According to Merriam-Webster, a trend can be defined as "the general movement over time of a statistically detectable change". In our specific case, we want to observe variations of occurrences of words or topics so as to identify interesting events. Trends are easily observed on the graphical representations of time series distribution as they appear as peaks.

As you can see figure 1 the frequency for the word 'bin' on the 5Th may 2011 rised sharply and on the 7th may 2011 this word came back to its normal frequency. This is the example of a perfect trends, short and volcanic.
An other example is shown in figure 2 for Prince William and Kate Middleton's wedding.

Figure 1: $p('bin')$

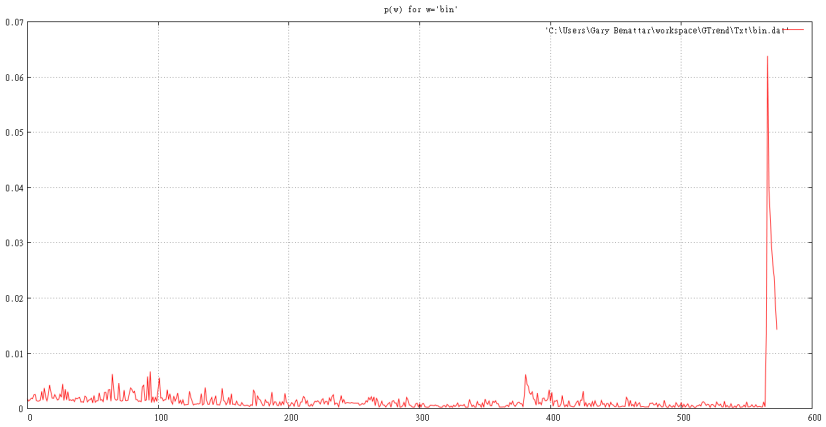


Figure 3: $p('favorit')$

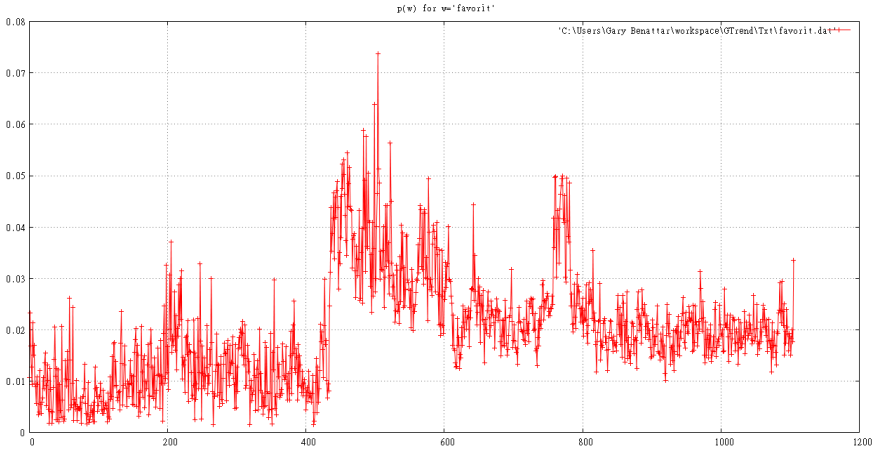
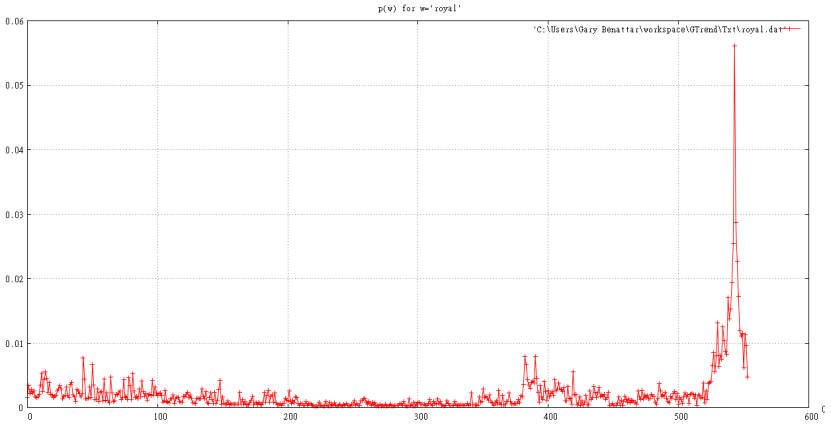


Figure 2: $p('royal')$



A good example (figure 3) is the general distribution of the word 'favorit'[4] which is, particularly on the *Toluna* website, a very common topic due to a large number of title which start with 'What is your favorite'. The average frequency for this word is very high, yet it is not a trend.

The biggest difficulty on trend detection is to know when exaclty the trend started and when it will stop, it can be less than one day or one hour depending on the website activity and the power of the trend.
Before introducing all the statistical methods used in this project and all the common problems for the implementation of a scalable architecture, we will introduce *Toluna* objects and data used in the system.

[4] As explained later, we use a stemmer in order to keep one canonical representation of morphological variations. As a result, 'favorite' as well as 'favorites' are mapped into the same stem "favorit", whose frequency over time we observe here.

2 Our Data: Polls and Topics from Toluna.com

The data we used for our work is based on *toluna.com* site activity from 2008 to May 2011. *Toluna.com* has two basic types of objects:

- **Topics:** A Topic in *toluna.com* is a kind of forum centered around a main question or topic asked by one user to the rest of the community. For example: *would you actually date someone you met on the internet?* is a topic currently running on toluna.com that has collected several hundreds responses/opinions. For example: "*No way there are to many sickos out there! Besides I am married so I am not looking to date!*"
  See here: `toluna.com/opinions/896033/would-actually-date-someone-internet.htm`.

- **Polls:** A poll in *toluna.com* is a simple question which can be answered in one or few clicks. For example: *Whats your favorite season? Summer, Fall, Winter, Spring, I love them all, I don't have a favorite season.*
  See here: `toluna.com/polls/1531080/Whats-your-favorite-season.htm`

A topic internal structure can be defined as follow:

```
{Title, Opinion*, Date, ..., ..., ...}
-> {'Do you like sushi?', { 'Yes I love it', 'No I hate
fish', 'sushi? what's this?' }, '2008-01-01',...,...,...}
```

A Topic can also contains other elements like videos or images, but they will be not used in this project.

An opinion internal structure can be defined as follow:

```
{TEXT, Opinion*, Date}
-> {'Yes I love sushi with wasabi', 'Really you like
wasabi?', 2008-02-02}
```

All the elements in the topic schema is all the elements used for the analysis. In fact most of the trend detection will be done on the topic title and not on the opinions. The opinions will be used to detect collocations for a specific topic in order to specify an elected trend.

A poll internal structure can be defined as follow:

```
{Question, Answer*, Date}
 -> {'Do you like sushi?', {'Yes','No','Never try'},
2009-12-12}
```

A poll can be described as a short survey with only one question and multiple answers which can be 'single choice' answers ('Yes','No') or 'multiple choice' answers ('Love sushi','and also grilled chicken'). In this project poll question will be used for single word trend detection.

2.1 Events and Timeline

In our project we will consider the following events:

- Topic creation: In this case we will simply consider the topic title as a new document added at the specific creation time of the topic. Words in the topic will be analyzed and their frequencies will be updated.
- Opinion on a topic: In this case we will in fact ignore the text of the opinion but simply repeat the topic title.
- Poll creation: We will add the text of the poll and the answers, we will analyze it and update the frequencies of the words.
- Vote on a poll: Similar to what we do for opinions, we will simply consider the text of the poll one more time for each vote.

The sum of the events for a given interval is all the activity on the *Toluna.com* website. Everytime someone is doing something on the website will produce an event and will be added to the trend analysis.

2.2 *Toluna* Data Size

In this project we used the *Toluna.com* database from the beginning of 2008 up to May 2011. The data is anonymized and contains the following data:

- 1,133,635 Polls
- 16,694,422 Opinions
- 668,295 Topics
- 4,524,688 Answers

In the next section we describe how the data is modelized in this project.

3 Language Model

3.1 Definitions

In our language model we will use single words and bigrams as our basic units. A bigram is a pair of words that appear at least once on the same sentence within a distance of 5 other words.

For example, in the sentence above the following bigrams can be extracted:

```
'language-model' 'language-use' 'single-word'
'single-bigram' 'word-bigram' 'basic-bigram' 'basic-unit'
'bigram-pair' 'bigram-word' 'pair-word' 'word-appear'
etc...
```

Note that in our definition, the words in a bigram do not have to be adjacent but can be separated by other words and they can appear in any order. We use 5 as the maximum bigram distance, as "Most of the lexical relations involving a word w can be retrieved by examining the neighborhood of w, wherever it occurs whitin a span of five"[8] (-5 and +5 around w) . This will allow us to discover collocations, where a collocation is defined as a recurrent combinations of words that co-occur more often than expected by chance and that correspond to aribtrary word usage[8].

- Collocations

  In this work, collocations are used in order to specify a trend. Trend analysis is done on topics and polls. Once we detect a trend t_d we extract, count and sort all the collocations in the opinions and the answers which belongs to a topic or a poll containing t_d. Finally t_d will be equal to the top collocation (For more details see 4.4 Trend detection algorithm).

  We decided to show collocations only for trends, because the collocation always provides additional information because it occurs many times in the corpus. For example, in the collocation 'Mark Zuckerberg' it could be argued that 'Zuckerberg' is good enough and that 'Mark' is adding too litle information. However, we chose to use the collocation anyways because in the more general case it can help disambiguate.

- Stop words

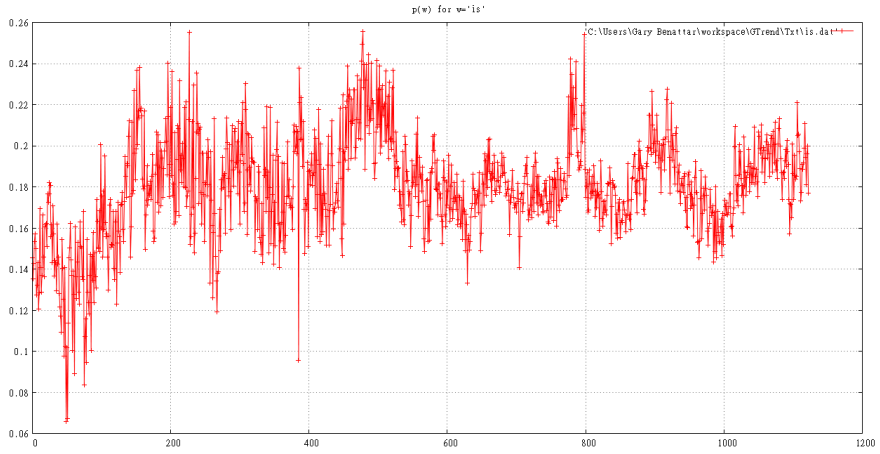
  A stop word is a common word which give us no information about the current topic or the current trend but that are mostly grammatical words such as articles, prepositions, etc. The most common stop words in English are: 'the', 'a', 'an', 'do','you','I','about',...

  Generally a list of stop words is used in text analysis which is not related to the actual context of the analysis and are language dependent. In processing of natural language data, stop words will be filtered out. The sentence above after filtering out the stop words will look like: *'Generally list stop words text analysis related actual context analysis language dependent'.*

  For example in the *snowball*[5] project or *lucene* [6] it is necessary to populate or create the stop word list for each additional language. In this work, we propose to not use a fixed stop words list but rather to use statistics to be able to ignore words that appear too frequently or with a frequency too flat. For example recurring topics on Toluna should not be considered as trends. (figures 4).

[5] http://snowball.tartarus.org/
[6] http://lucene.apache.org/

Figure 4: $p('is')$



Extracting these words will give us as well as common words something that we can call context common words. The following list contains a subsequence of context common words. You can see [what] [best] [favorite] which are clearly context common words.

you, the, do, is, what, to, of, which, your, who, a, in, should, for, like, think, will, how, would, be, best, this, are, on, or, have, and, most, with, it, i, if, prefer, about, win, at, new, whats, more, that, when, from, ever, favorite, better, did, many, use, we, get, ...

- Stemming

  > "[...] stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form - generally a written word form" Wikipedia

  Natural language processing systems often use stemmers in order to reduce the word forms of a single word. For example, 'fishing', and 'fish' belong to the same stem 'fish', so the output of the stemming process for these 2 words will be 'fish'. For example in the sentence 'I love fishing because I love fish', the single words 'fishing' and 'fish' will be count like the same word due to the stemming process [6]. Real words will be used only in the user interface in order to display understandable trends.

3.2 The model

Each event as defined in the previous section, will be represented by a tuple containing the following information: (event data, date). In order to classify our data, we will need to split the data in two part, on one hand, polls and topics and on the other hand opinions and answers.

- Topic and Opinion

  Upon receiving a topic creation event, the software will analyze the topic title, split the title into words and increment the word count of each word at the current date.

  As an opinion is linked to a topic, we propose a simple model in order to be not dependent to the topic creation date but to the opinion creation date, we simply re - publish the parent topic at the current date.

  For example:

```
{Event: Topic, Id: 1, Title:
        'What do you think about gladiator?',
Date: 2008-01-01 }
{Event: Topic, Id: 2, Title:
        'Should I buy the new epson camera?',
                        Date: 2008-01-01 }
{Event: Topic, Id: 3, Title:
        'I love fishing because I love fish',
                        Date: 2008-01-03 }

{Event: Opinion, Id: 8, Text:
        'I love this movie and I can re-watch
        again again and again,
         I want to become a gladiator',
                Parent: 1, Date: 2008-01-03 }
{Event: Opinion, Id: 9, Text: '
        Very good film!',
               Parent: 1, Date: 2008-01-03 }
{Event: Opinion, Id: 15, Text:
        'This is the longest movie
         ever seen in Hollywood',
                Parent: 1, Date: 2008-01-04 }
{Event: Opinion, Id: 100, Text:
         'No',
                Parent: 2, Date: 2009-01-04 }
```

  Classification by date (stop words were removed): (w,count)

```
2008-01-01 ('gladiator',1) ('movie',1)
        ('epson',1) ('camera',1)
2008-01-03 ('gladiator',2) ('movie',2)
        ('fish',2) ('love',2)
2008-01-04 ('gladiator',1) ('movie',1)
2009-01-04 ('epson',1) ('camera',1)
```

  As we can see, everytime a new opinion appears on a topic, the topic is re-published at the current date, because we assume that if a user let an opinion on a topic, it means that the topic itself as a trend is reinforced. This allows us to in fact, as a first draft, completely ignore the text of the

opinion and simply strengthen the topic. The text of the opinion will be used in a second stage as shown below.

- Poll and Answer

  For polls and answers the process is very similar to the one we just described in the previous section. However we wanted to introduce a difference that would reflect that it is much easier to vote on a poll (single click) than to give an opinion of more than 20 words (over 100 clicks). So we decided to republish the text of the poll for each number of votes. Some testing on our data allowed us to set the number to 10.

  For example:

```
{Event: Poll, Id: 7, Title:
        'Do you like french fries', Date: 2008-01-01}
{Event: Poll, Id: 8, Title:
        'Do you like french potatoes',
        Date: 2008-01-01}

{Event: Answer, Id: 12,
        Text: 'yes', Parent : 7,Date: 2008-03-03}
{Event: Answer, Id: 18,
        Text: no, Parent : 7,Date: 2008-03-03}
{Event: Answer, Id: 19,
        Text: 'yes', Parent : 7,Date: 2008-03-03}
{Event: Answer, Id: 22,
        Text: 'yes', Parent : 7,Date: 2008-03-03}
{Event: Answer, Id: 29,
        Text: 'yes', Parent : 7,Date: 2008-03-03}
{Event: Answer, Id: 32,
        Text: 'yes', Parent : 7,Date: 2008-03-03}
{Event: Answer, Id: 33,
        Text: 'yes', Parent : 7,Date: 2008-03-03}
{Event: Answer, Id: 34,
        Text: 'yes', Parent : 7,Date: 2008-03-03}
{Event: Answer, Id: 37,
        Text: 'yes', Parent : 7,Date: 2008-03-03}
{Event: Answer, Id: 40,
        Text: 'yes', Parent : 7,Date: 2008-03-03}
```

  Classification by date (stop words were removed): (w,count)

```
2008-01-01 ('french',2) ('fries',1)
        ('potatoes',1)
2008-03-03 ('french',1) ('fries',1)
```

  In the following section we will propose a statistical method in order to detect trend and collocation trend.

4 Trend detection

4.1 Notations

- Word frequency

  The frequency for a word $f(w)_d$ is equal to the number of occurrence of w in d.

- Word proportion

  The proportion for a word w in d is equals to:

  $f(w)_d$: the frequency for w in d

  t_d: number of words in d

  $$p(w)_d = \frac{f(w)_d}{t_d}$$

  Example: in the list, we list the proportion of the word *sushi* (value=frequency)

  date:2009-11-06, value=5.72E-4
  date:2009-11-11, value=0
  date:2009-11-12, value=3.97E-4
  date:2009-11-21, value=6.48E-4
  date:2009-12-07, value=0
  date:2009-12-08, value=5.62E-4
  date:2009-12-31, value=5.46E-4
  date:2010-01-19, value=8.10E-4
  date:2010-02-02, value=3.34E-4
  date:2010-02-07, value=3.75E-4

- Mean

  The mean for a word is the average proportion for the word w on the overall period.

  N: Total number of days

  $p(w)_d$: proportion at the date d

  $$\bar{x} = \frac{1}{N} \sum p(w)_d$$

- Standard deviation

  The standard deviation show how much variation there is from the average. A low standard deviation indicates that the data is very close to the mean.

  $$\sigma = \sqrt{\frac{1}{N} \sum (x_d - \bar{x})^2}$$

4.2 Zscore

- What is a Zscore

  A zscore indicates how many standard deviation an observation is above or below the mean. That's what we usually called standardizing data.

$$Z(w)_d = \frac{x_d - \bar{x}}{\sigma}$$

  The following subset contains zscore at each date for w = 'sushi' (zscore = value)

```
date:2009-11-06, value=0.89
date:2009-11-11, value=2.61
date:2009-11-12, value=0.54
date:2009-11-21, value=1.04
date:2009-12-07, value=1.90
date:2009-12-08, value=0.87
date:2009-12-31, value=0.86
date:2010-01-19, value=1.42
date:2010-02-02, value=0.45
date:2010-02-07, value=0.54
date:2010-02-13, value=0.55
```

  The z-score is a measure commonly used in finance and defined as:

> "The Z-Score is a multivariate formula for a measurement of the financial health of a company and a powerful diagnostic tool that forecasts the probability of a company entering bankruptcy within a 2 year period"[1].

  As a z-score detects large variations from the normal, in our context it will be used to detect large and uncommon variations of a word proportion at a given date. In the following section, we explain how we use the z-score to identify a trend.

- Bracketing

  In order to interpret this value we put our raw data into brackets and set a threshold for each bracket.

  The proportion can be expressed as:

$$Po(w) = \frac{nbD(w)}{c_d}$$

  where $nbD(w)$ stands for the number of days during which w appears, c_d stands for the number of days covering the entire period.

  The bracketing is done as follows:

```
[0-5[, [5-10[, [10-20[, [20-30[, [30-40[,[40-50[,
[50,60[, [60,70[, [70,80[,[80-90[, [90,100],
```

We can see that the interval for the two first brackets are different. This is because these brackets will contain very uncommon words which will need a special analyse due to the zero frequency problem (see section 5).

As explained, each bracket contains a z-score threshold. These 11 thresholds are the results of various training sessions among different set of data. First of all, these thresholds were tuned on a training data, and then were connected to another set of data. Trend extraction was a success in both case. More $Po(w)$ is low, more the result of a small frequency variation will be high for the z-score.

The closer we are to 100, more the threshold should be low. Keep in mind that each day we have 0 occurrences for a word, the proportion will be equal to zero.

Table 1 shows a subset of the words within the first, second and the last brackets. Many of these words are misspelled or are not proper words per se (they can be names or parts of names). Yet even these are interesting because they come from new events that are suddenly occurring, examples include:'DSK','bin','laden','osama','swayze'.

Table 1: Bracketing content

[0-5[∪ [5-10[	[40-50[	[90-100[
aaarghhhhh	absolut	age
aand	appreci	agre
aandolan	besid	allow
aang	cheer	anim
aani	common	anyon
aap	confus	anth
aapent	coupon	babi
aapko	delet	bad
aar	dirti	ban
aaramerican	disappear	befor
aarcheri	disast	believ
aardvark	employe	book
aargghhhhh	especi	brand
aargh	exact	buy
aaron	expert	call
badtast	flavor	car

Table 2: Final z-score threshold

[0-5[	[5-10[	[10-20[	[20-30[	[30-40[
20	25	15	12	10

[40-50[	[50-60[	[60-70[	[70-80[	[80-90[	[90-100]
10	9	8	6	5	5

4.3 The zero frequency problem

What will happen if a word which belongs to the first bracket suddenly happens to be used inside a topic title or poll title? In the best case, 95 percents of frequencies will be equal to zero. If this word is used suddenly, even if the variation is not significant, the z-score value will jump and be greater than the current threshold.

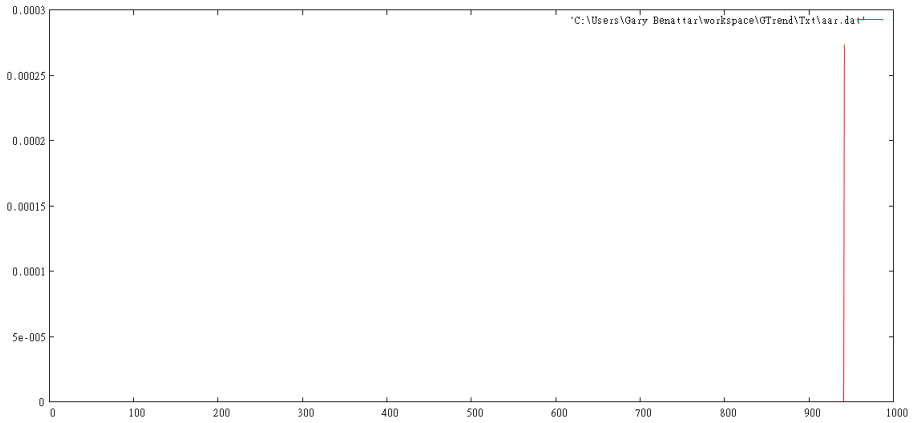
The example below shows the current distribution for w='aar', as we see a single occurrence, for all the other dates, the proportion is equal to zero.

$$\mathrm{date}{:}2010{-}08{-}02,\ \ \mathrm{value}{=}2.73\mathrm{E}{-}4$$

Table 3: The zero frequency problem where w = 'aar'

d	$p(w)_i$
2008-01-01	0
2008-01-02	0
...	0
2010-08-02	2.73E-4
...	0
...	0

Figure 5: $p('aar')$



Algorithm 1 Zero frequency problem

$l \leftarrow w \in [0-5[\cup[5-10[$
for $e \in l$ **do**
  $averageList \leftarrow mean(p(e))$
end for
return $mean(averageList)$

Table 4: Zero frequency threshold

Value
8.91E-4

You can see that on 2010-08-02 this word can be elected as a trend.

The common way to solve this problem is to eliminate zero values and to consider that this word appeared like others (Laplace estimator).

We propose here to tackle this issue in a different manner: we take all the words within these brackets and calculate the average proportion for each word, and finally calculate the average of the list of averages for all words (See Algorithm 1).

We finally verify whether a word within these brackets has a large frequency variation as compared to the value calculated on the training data (see Table 4).

4.4 Trend detection algorithm

The z-score is calculated for each single word in the titles of topics and polls and compared to a given threshold. The key steps of the algorithm are outlined below.

Algorithm 2 Trends detection

$d, nbD(w), c(d), Po(w), Z(w)_d, f(w)_d, t_d, p(w)_d$;
$zeroFreq \leftarrow 8.91E-4$
if $Po(w) \in [0-5[$ **then**
  if $p(w) \geq zeroFreq * 10$ **then**
    if $Z(w)_d \geq threshold(Po(w))$ **then**
      w is a trend
      return $findCollocationCandidate(w, 5)$
    end if
  end if
else
  if $Z(w)_d \geq threshold(Po(w))$ **then**
    w is a trend
    return $findCollocationCandidate(w, 5)$
  end if
end if

As per this algorithm, once w is identified as a trend, we try to find the best

collocation that involves it.

Algorithm 3 findCollocationCandidate

```
w, window, candidatesList;
for event ∈ topic(w) ∨ poll(w) do
  for i = 0 → event.text.size do
    for j = i + 1 → min(event.text.size, window) do
      col ← event.text[i] ∪ event.text[j]
      if col ∈ candidatesList then
        candidatesList(col) + +
      else
        candidatesList ← col
      end if
    end for
  end for
end for
if candidatesList ≠ ∅ then
  return max(candidatesList)
end if
return w
```

For illustration, let us consider the following example with the word 'bin' on 2011-05-02, which is the day of Bin Laden's death. For w='bin' we obtain:

```
date:2011-04-27, value=0
date:2011-04-30, value=3.424E-4
date:2011-05-01, value=0.01
date:2011-05-02, value=0.06
date:2011-05-03, value=0.04
date:2011-05-04, value=0.03
date:2011-05-05, value=0.03
date:2011-05-06, value=0.03
date:2011-05-07, value=0.02
date:2011-05-08, value=0.02
date:2011-05-09, value=0.01
```

$$Po(w) = 47.84$$

$$\bar{x(w)} = 6.63E - 4$$

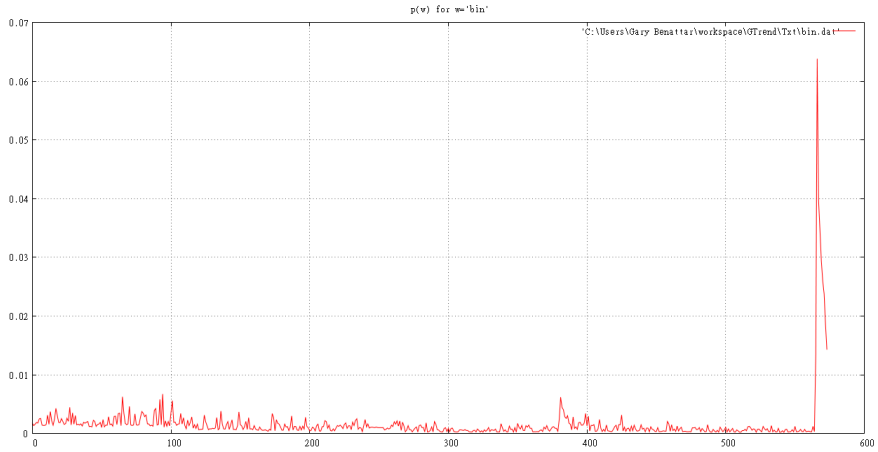
$$\sigma(w)_d = 0.0021$$

$$Z(w)_d = 30.25$$

As we can see w belongs to [40-50 [and the threshold for this bracket is equal to 10, $Z(w)_{2011-05-02}$ is more than 3 times the threshold. On the 2011-05-02, the word 'bin' has been detected as a volcanic trend.

Now let's try to investigate inside opinions. First of all, the list below contains some topics created on 2011-05-02 containinig 'bin':

Figure 6: $p('bin')$



[Do you think the killing of Osama Bin Laden will
help President Obama get re-elected?,
Is bin laden really dead?, Probably done to death,
but has the death of Bin Laden made the World a safer place ?,
Do you think that Bin Ladins' followers will retaliate ?,
Bin Laden is dead according to news reports.
Is that a good thing for the peace process?,
Osama Bin Laden has been killed by the U.S. troops.
What do you think this event will bring? Peace,
unity of all humans or horrible turmoil and retaliation?],

Now, inside each opinions about this trend, we detected the top collocations:
List of top collocations for the word 'bin'.

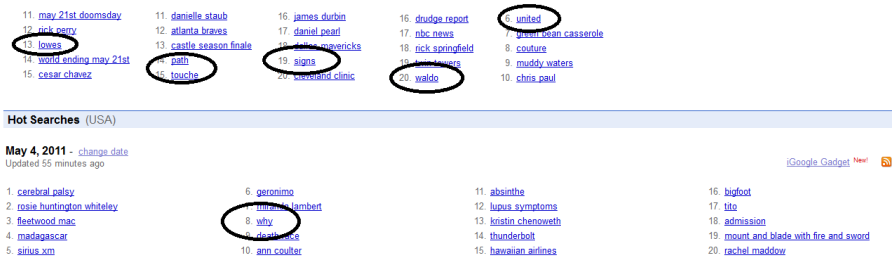
[laden bin=3582, bin laden=3546, bin osama=2302,
osama bin=2252, bin people=2051, bin death=1963,
dead bin=1959, bin dead=1902, laden osama=1686,
osama laden=1626, death bin=1502, laden people=1492,
bin body=1460, laden death=1458, bin killed=1425,
dead laden=1405, laden dead=1403, bin ladens=1131,
bin world=1103, laden body=1070, death laden=1062,
laden killed=1037, osama people=933, bin glad=929,
dead osama=920, osama death=901, bin news=894,
bin attacks=893, osama dead=874, bin feel=852,
laden ladens=830, bin obama=826, dead people=822,
bin hope=817, laden world=810, dead death=789,
obama bin=758, ladens bin=750, bin revenge=731,
bin troops=714, death osama=709, death people=707,
bin retaliation=703, bin true=699, bin bad=681,
laden glad=680, osama body=678,...]

Here there is one single top collocation, which is 'laden bin' with a number of occurrences equal to 3582, so the trends 'bin' will be replace by 'laden bin'. In this specific case, we see that the collocation brings most value. Indeed 'bin',

which is the volcanic single trend is not understandable without context. Collocation is the magic recipe to transform a 'statistical trend' into an 'interpretable' trend.

The bigram 'bin laden' was detected as a collocation trend on 1, 2, 3, 4, 5 May 2011.

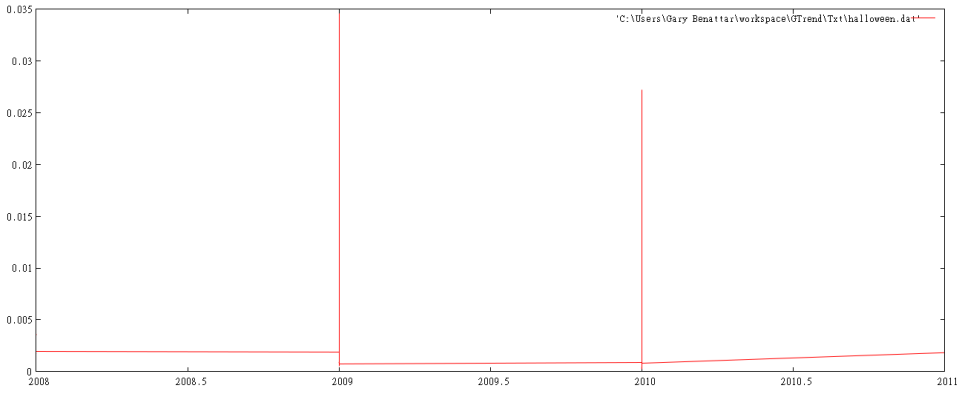
If we compare ourselves to Google trends, we see that by displaying only collocation trends rather than single words as Google Trends does from time to time, we avoid pitfalls such as showing empty words such as 'why' as might happen with Google Trends as shown below.



5 Results

- Recurring trends[7]
  - Halloween
    Detected as a trend on 2008-09-13, 2009-10-31

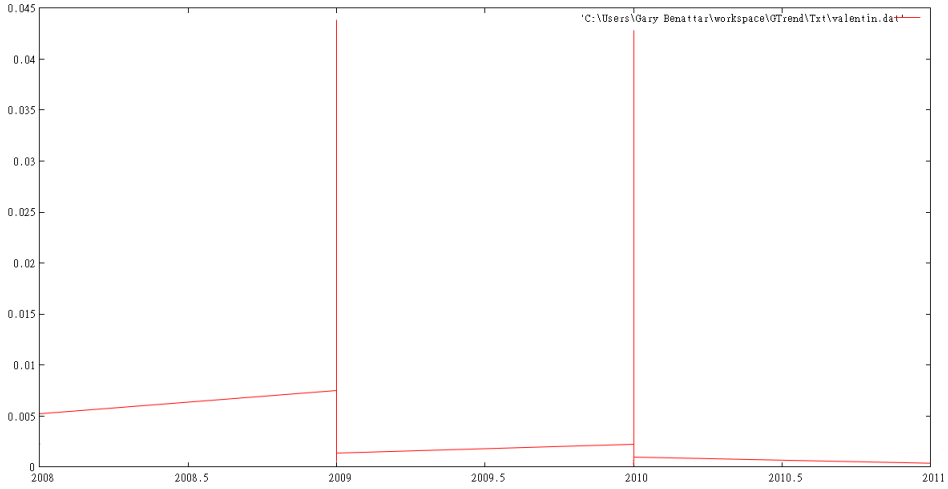
Figure 7: $p('halloween')$



  - Valentines's day
    Detected as a trend on 2009-02-14, 2010-02-14, 2011-02-14

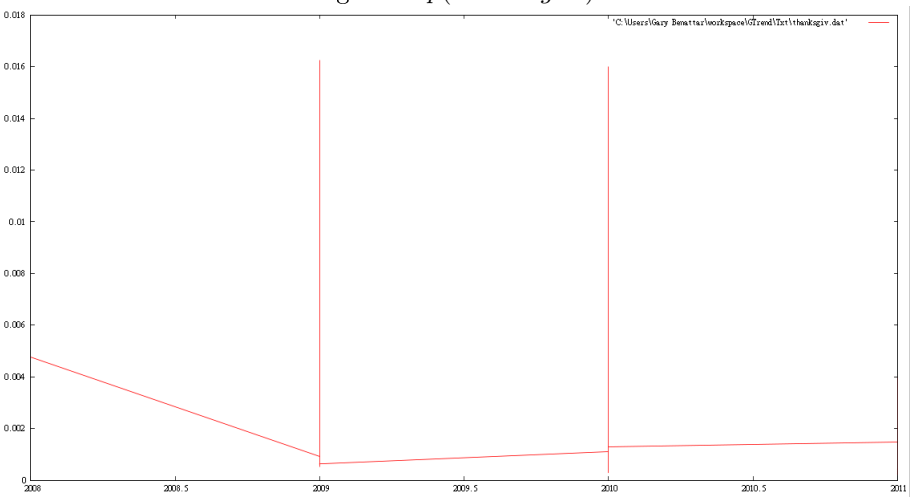
[7] Full list available upon request

Figure 8: $p('valentin')$



– Thanksgiving
Detected as a trend on 2008-12-25, 2009-10-17, 2009-11-27

Figure 9: $p('thanksgiv')$



- Interesting Trends

Table 5: Interesting Trends

w	d	top collocations
swayze	2009-09-15	dirty patrick - dancing patrick - patrick cancer
kanye	2009-09-15	beyonce opinion
joanna	2009-05-07	war lumley - ex lumley - gurkha lumley
lockerbie	2009-08-20	crime believe - people crime - released crime
swine	2009-04-30	phase flu - flu die
japan	2011-03-11	japan people - earthquake japan - tsunami japan
vat	2008-11-26	spend money - spend reduction - spend vat
sachin	2010-10-11	surely overs - current overs - player overs
veterans	2010-11-11	thank day - veterans day
lebron	2010-07-09	lebron cleveland - miami cleveland - fans cleveland
anna	2011-04-08	hazare anna - anna hazare - anna corruption

Table 6: Japan earthquake

w	d	top collocations
japan	2011-03-11	japan people - earthquake japan - tsunami japan
japan	2011-03-12	japan help - help japan - japan quake
japan	2011-03-13	japan country - japan happened - japan future
japan	2011-03-14	japan nuclear - japan news - japan security
japan	2011-03-19	japan nuclear - japan power - nuclear japan

Figure 10: $p('japan')$

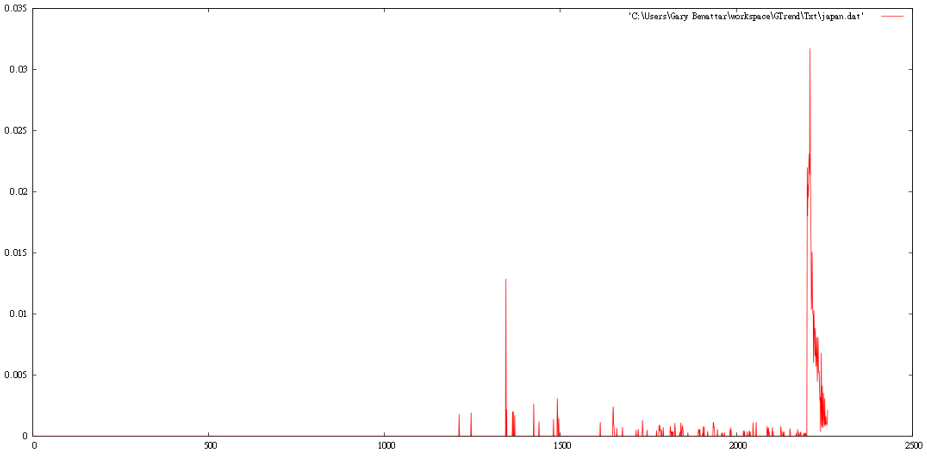


Figure 11: Japan top collocations timeline

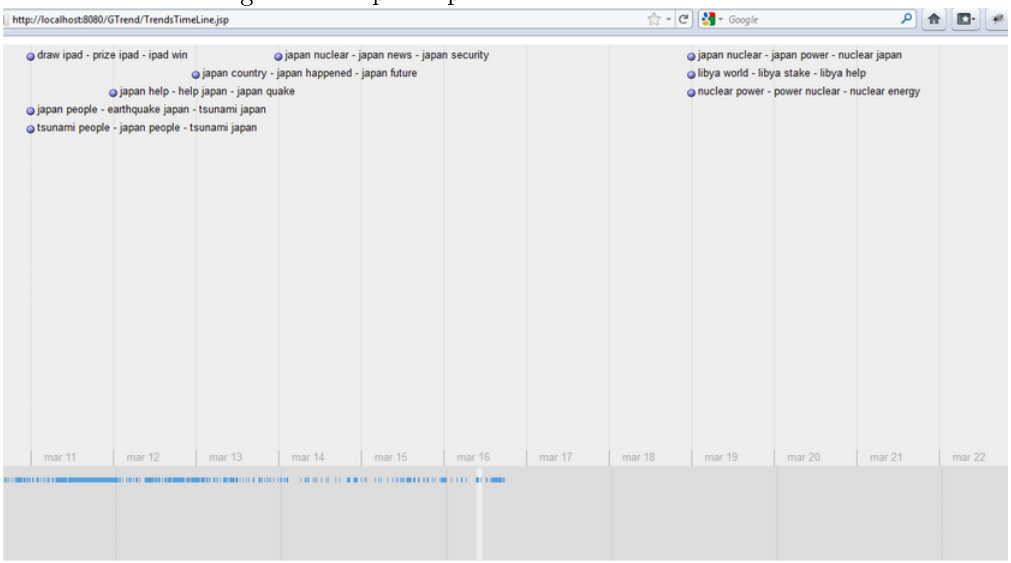


Figure 12: Bin Laden and Royal Wedding top collocations timeLine

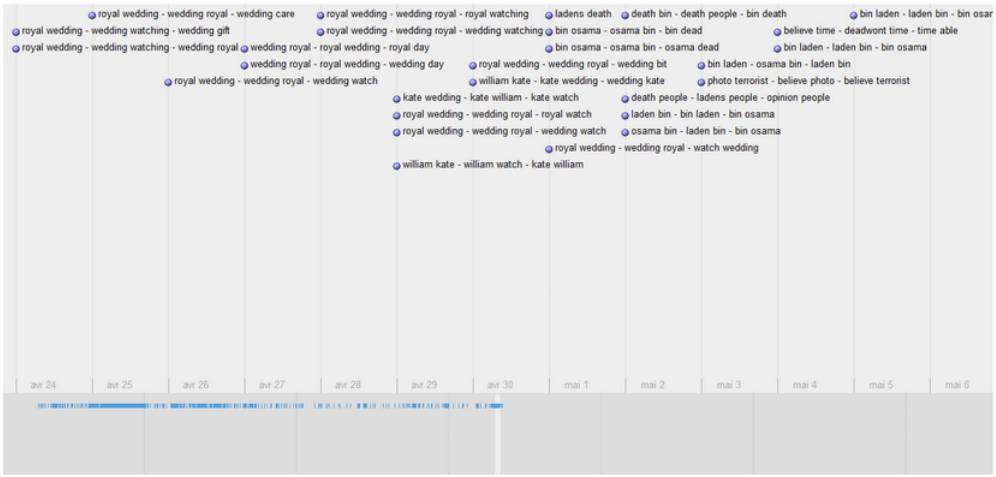


Figure 13: Top collocations on timeline

Figure 14: $p('swine')$

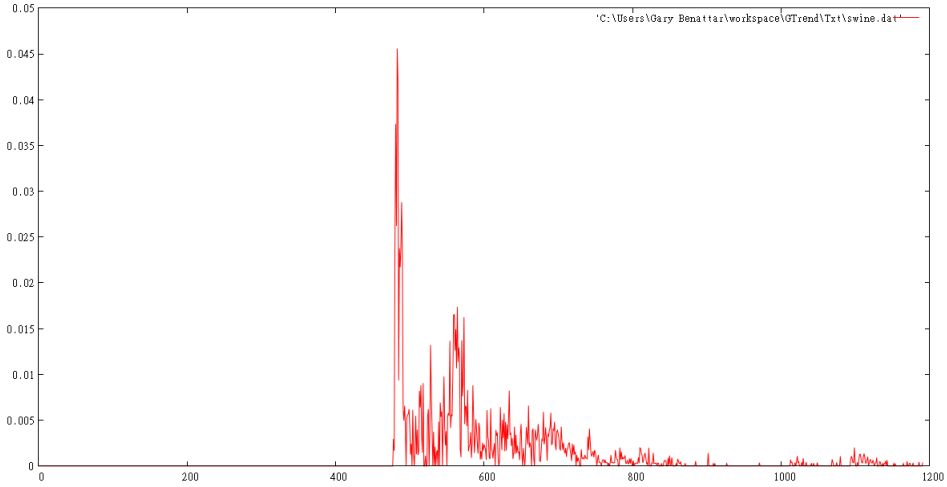


Figure 15: $p('swayze')$

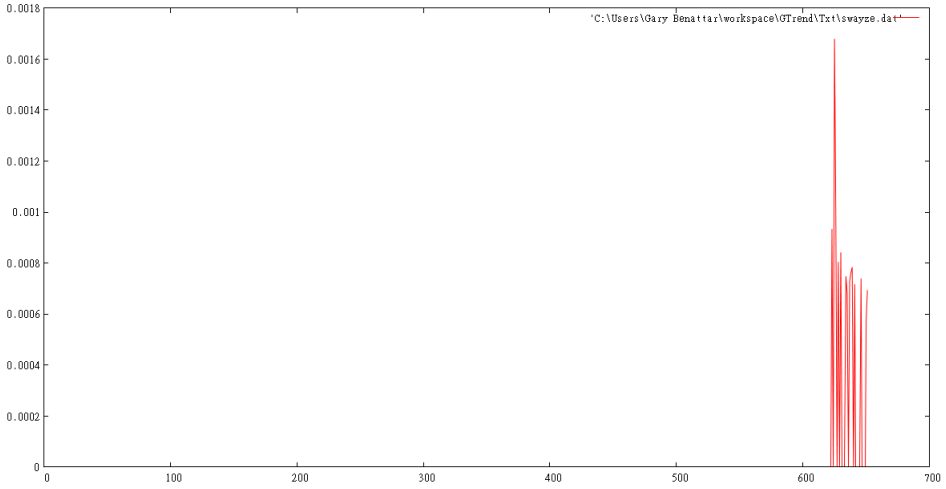
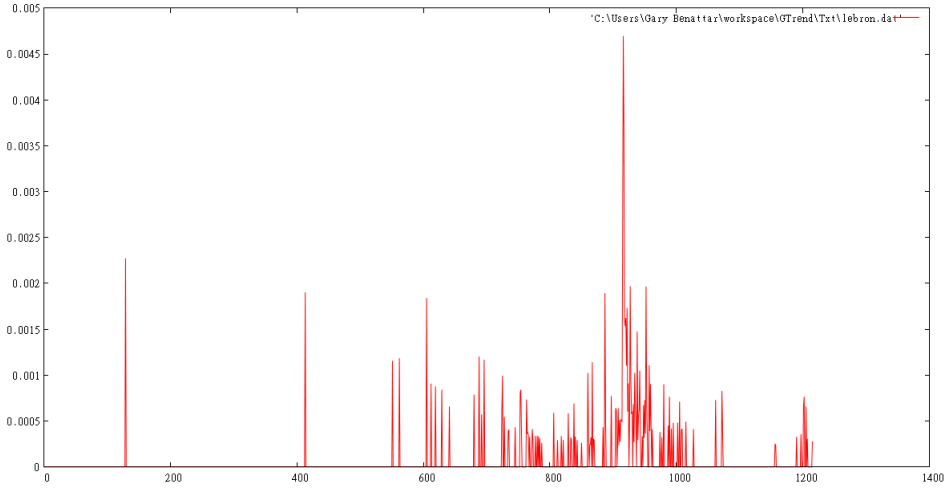


Figure 16: $p('lebron')$



6 Implementation

As described, we have more than 20 millions events in our data, which needs to be split by words and count. Moreover, we have millions of events every day, which need to be parsed and classified in real time. Given the size of this data, we decided to use an appropriate large scale system for processing and analyzing the data. We explain below why we went for Hadoop and HBase. We will conclude by sharing our (failed) experience in a preliminary implemetation based on MySQL.

6.1 Scalability

- Hbase

  Hbase is a distributed database which was the open-source answer to Google BigTable [2]. A table in Hbase contains rows keys (byte arrays) ordered by Comparator with atomic updates, even if it comprises a thousands of columns.

  Columns are grouped by families, with a column family prefix and a qualifier which is the value of the family. Each combination of single couple (row, family) contains a Timestamp and a Value. All the elements in a table are bytes array.

  The example below is a row from the table 'frequencyAllEvent' with row value = 'laden' and family = date and qualifier = 2011-05-02, the standard notation for this data is 'laden',date:2011-05-02

```
date:2011-05-02 timestamp=1305375619910,
        value=0.057473875511131305
```

  The most important characteristic of HBase is its capability of splitting a table into regions. In fact, one can define the start and end row for each region. Each region belongs to a region server, and the content of each region server is indexed on a master, which redirects the client to a good node. Data transfer between region server and clients is done directly without the help of the master.

  The Apache foundation also provides a library in order to communicate with hbase through the sofware. We list below a partial list of all the common operations conducted with Hbase:

  - open a table

```
HTable z-scoreAllEvent =
          new HTable(config, 'z-scoreAllEvent');
```

  - get row from a table

```
Get rGet = new Get(Bytes.ToBytes('rowValue')
Results res = z-scoreAllEvent.get(rGet);
```

  - get all families for a given row

```
Get rGet = new Get(Bytes.ToBytes('rowValue')
for(KeyValue kv : z-scoreAllEvent.get(rGet).list){
        byte[] qualifier = kv. getQualifier();
        byte[] family = kv.getFamily();
        byte[] value = kv.getValue();
}
```

  - get all the rows and scan the entire table

```
ResultScanner scanner =
        tableOpinions.getScanner(new Scan());
```

```
for (Result res = scanner.next();
        res != null; res = scanner.next()) {
}
```

When we started this project, as were coming from the MySQL world, the transition to HBase was not trivial. The issue was not in exporting the data from MySQL tables to HBase but rather in re-designing the architecture. We started with four MySql tables Polls, Opinions, Topic and Answers and we designed our HBase architecture (Table 7) in order to be able to answer to this kind of questions:

- How much topics was created at a given date?
- What are the opinions linked to a specific topic?

Table 7: HBase table architecture

Table name	Row	Family	Value
Topics	Date	TopicId	Title
Opinions	TopicId	Date	Count
Frequencies	Word	Date	Count
Polls	Date	PollId	Title
Answers	PollId	Date	Text
Stem	Stem	Stem	Real Word

Hbase is based on HDFS (Hadoop Distributed File System) which is a portable file system written in Java for the Hadoop framework. While HBase is the open-source equivalent to Google's proprietary BigTable, HDFS[7] was initiated by Yahoo! as an open-source answer to Google File System (GFS)[4].

Finally, we needed a last piece in order to handle additional tasks inherent to distributed needs such as configuration, naming and synchronization tasks. All these are critical when running on a large cluster, with constantly failing disks, failing servers, replication and shifting roles between nodes.

- Zookeeper

  We chose to use Zookeper[5] for this purpose. Zookeeper is used here to synchronize the HBase database and allow clients to locate the adequate region server. It ensures that only one master is active and also improves efficiency by preventing inactivity after opening a table.

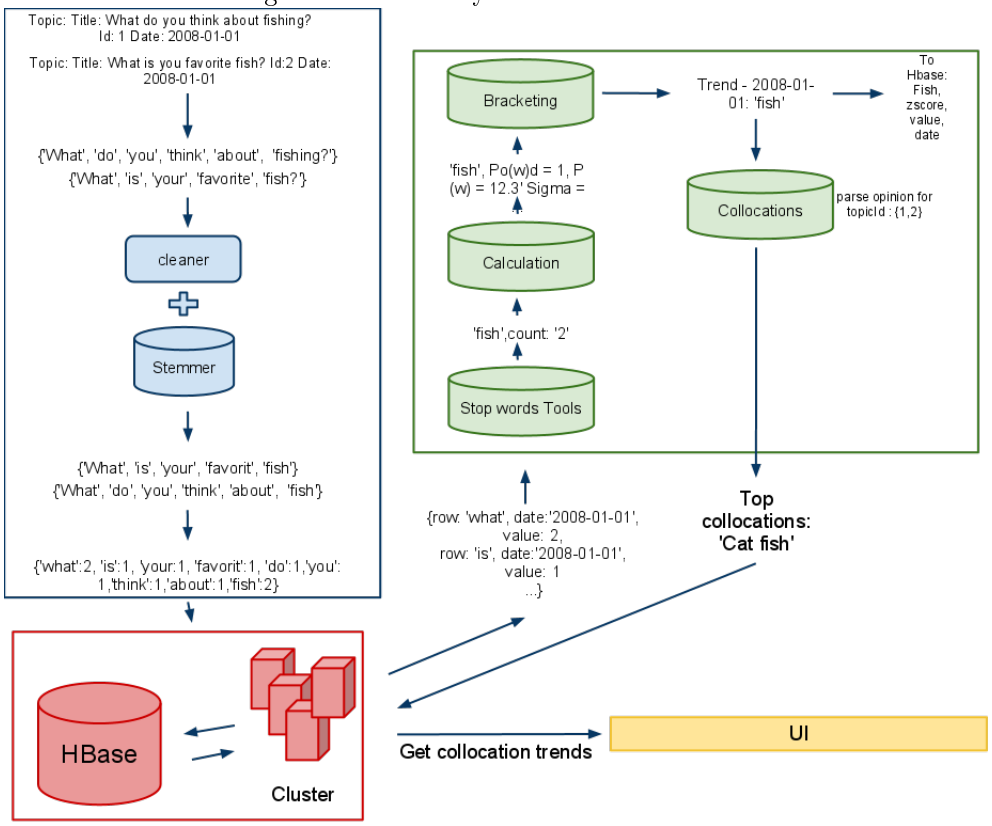
6.2 Software architecture

This system comprises the following components:

- Stemming component: Rather than re-implementing for instance the Porter's algorithm for our own stemming component, we used the Snowball open-source package, which provides the most commonly used English stemmer. Note that we reduced our analysis to English content only at this stage.

- Training component: We implemented this component to support all the statistical training methods, and address issues like the zero-frequency problem thresholds. This component accesses the data-set offline and is never used in production mode.
- Base component: We implemented in this component the objects used in our system.
- Trend Detection component: This component is one of the critical part of the system, it reads data from the database, and extract trends. It interacts with the Stemming, Utils (described later) and Base Components.
- Utils component: This component contains all the utilities used during the analysis stage as well as the formatting utilities needed for displaying the results in the user interface. Examples include export to CVS,XML,TXT, export into log file, ArrayList, HashTable (increment element, sort, ...), String util methods (cleaning a String, date to String format, ...), common database operations (set a connection, queries, ...), math utils (mean, standard deviation, number of day between two dates, ...). It interacts with the Trend Detection, Web (described later) Components.
- Test component: This component should also never be used in production mode, it contains some basics test.
- Web component: This component contains web servlets and it is linked to WebContent which contains all the JSP files.

Figure 17: Overall system architecture



6.3 Lessons learned

At the beginning we started this project in Python with Django for the web interface and a MySql database, we realized very soon that it was not the good solution. It was hard to find a good IDE and trying to analyse a huge amount of data with MySql was impossible.

Training even with a very small data set would take way too long. This was worrisome as intuitively we believed that no good results could be obtained without a minimum of 1 million documents. So we investigated other technologies and the answer was brutal: MySQL would not scale to the level we needed and the literature pointed is to 3 Google-originated technologies: MapReduce [3], BigTable [2] and GFS [4]. After much reading, we decided to re-design the entire system, move to a Hadoop environment (the Yahoo!-initiated open-source equivalent of MapReduce) and adopt HBase, We also decided to reimplement everything in Java as this is the native language of Hadoop (even if one can interface with Hadoop in Python or C++, Java is still the language of choice when working in Hadoop).

The first results were not that good, many strange words and misspellings appeared. This led us to investigate our settings and especially the z-score threshold. It seems that a single z-score threshold could not be used for all the proportion computations. We started to build all the brackets and ran lot of

training, it was better but for the first bracket (greater than zero and less than five percent) we had 90% of bad words, most of us was miss-spelling words. We were confronted to the zero-frequency. After more investigation, We found an article where the authors solved the zero frequency problem with a Laplace estimator.

Inspired by this article, We decided to search for a dedicated threshold for these problematic words, and we experimented with the mean of the mean. Results were satisfactory. In order to validate everything we asked Toluna's database team to provide us a clean and up to date database. we connected the new database to the software. Results were also satisfactory.

At this stage, the system extracted decent trends, yet they were not easy to interpret for users not familar with the dataset. As we followed the news, we knew that Bin Laden was dead so when we extracted the word 'bin', we did the connection, but our goal were to understand without any background the displayed trends and for us 'bin' was really not a good trend for the user.

We had the chance to talk to ToLuna SVP, Frank Smadja, an ex-computational linguistics researcher, who happens to be one of the pioneers of collocation extraction and analysis. After a short brainstorm, we realized that collocations were the key to this interpretatibility problem. At this moment we took the decision to find single words trends in polls and topics only and to specify trends found with collocations inside opinions and answers, and 'bin' became 'bin laden', 'royal' became 'royal wedding', 'patrick' became 'patrick swayze'.

The last step was to design a user interface. Fortunately HBase provided already some very convenient utilities, We just had to add some export tools and devise some charts and timeline plug-ins.

Data visualization is always a key components in any data-mining effort especially when the output needs to be displayed to the user. As an example Google trends while being very sophisticated with its rich dashboard, might be a bit too complex for casual users. Yahoo! Clues that was just launched in the US targets more naive users (as Yahoo! typical users are) and in our opinion with its simplicity and attractive facets might attract less sophisticated users if it reaches them.

Finally, we have not included any evaluation in this report. However we do plan to conduct thorough comparisons between our approach, Yahoo! clues and Google trends in order to validate our contributions.

7 Acknowledgements

I am thankful to the Department of Computer Science at Paris 6 University for providing me with a rich environment and giving me the opportunity to graduate in one of the best institutions in the world. Many thanks to my advisor, Philippe Trebuchet for his advice on my research and this work, Philippe has been an inspiration for me both professionally and personally and it was a real pleasure to work with him. I would like to also thank Frank Smadja for his insights on text analysis and collocations, and Yoelle Maarek for having the patience to proofread this report and make dozens of useful comments. Many thanks to my brother Jeremy for sharing my passion with trends and for challenging me with common sense questions.

References

[1] Edward I. Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *Journal of Finance*, September 1968.

[2] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: a distributed storage system for structured data. pages 15–15, 2006.

[3] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[4] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. pages 29–43, 2003.

[5] M.; Junqueira F.P.; Reed B. Hunt, P.; Konar. Zookeeper: Wait-free coordination for internet-scale systems.

[6] M.F.Porter. An algorithm for suffix stripping. 1980.

[7] Sanjay Radia. Yahoo! hdfs under the hood.

[8] Frank Smadja. Retrieving collocations from text: Xtract. *Comput. Linguist.*, 19:143–177, March 1993.