Development of an Aggregated News Search Engine

Rodrigo Doria Medina and Thibault Roucou
Master Human Media Interaction
University of Twente, Netherlands

January 19, 2012

Abstract

1 Overview of the project

1.1 Description

Aggregated search retrieves search results coming from multiple sources. This project is aimed to provide a Newspaper based on aggregated search. A limited number of results is required (20 documents here). The newspaper presents different type of documents such as videos, pictures and text. The main issues to deal with will be the relevance and the recentness of the retrieved documents.

Twitter is one of the biggest sources for recent news currently available. Its information will be embedded in the core algorithm of this project.

1.2 Sources

Three types of documents are dealt in this project : Text, Video and Image. Therefore three different sources will be used. For videos YouTube, because it has a very well developed and documented API. For images, Flickr will be used also because the API provides full capabilities. Finally for texts, because providing news to the user is the main point, a news search engine is the accurate solution and GoogleNews is the selected one.

1.3 Technologies used

For this project, Java will be used with Google Web Toolkit. A lot of APIs for third parties services are available in Java and the following ones will be implemented : gData API [1] to retrieve videos from Youtube, FlickrJ [2] for Flickr images, Rome [3] to retrieve information from the GoogleNews RSS feed and finally Twitter4J [4] for Twitter.

In order to perform user testing and therefore an hosting solution is needed for the project so that it can be accessible online. For this purpose, Google App Engine [5] is the solution to deploy the project.

1.4 Evaluation methods

In order to evaluate the system, the user feedback is required. For that purpose, the project has been deployed online so a sample of users can test it. The evaluation is a two-step process.

[1] http://code.google.com/intl/fr-FR/apis/gdata/

[2] http://flickrj.sourceforge.net/

[3] http://java.net/projects/rome/

[4] http://twitter4j.org/en/index.html

[5] http://code.google.com/intl/fr-FR/appengine/

The first is the rating of the baseline which was defined as a non so naive Round Robin algorithm. It's actually based on all the default algorithms provided by the APIs but the outcoming order of the results is respected (first a video, then a picture and finally a text).

The second part displays the outcome of the customized algorithm.

Each document can be rated as "Relevant", "Non relevant" and "Don't know".

After all the documents are rated, the results are sent by e-mail. These results contains the computed precision at 5, 10 and 20 documents retrieved. Since the volume of documents retrieved is not an issue, the evaluation is focused on the precision's computation.

2 Building the baseline

2.1 The APIs

2.1.1 Google News

Google News doesn't provide an API anymore. Therefore, the RSS feed and the search system will be used.

A Google News query is of the following form :

"http://news.google.com/news?q=*user_query*&output=rss&hl=en&as_qdr=w"

Explanation of the different parameters :

- q : the query provided by the user
- output : the output format given by GoogleNews, here we choose rss.
- hl : the language of the provided results. We decided to retrieve in priority english results.
- as_qdr : this parameter allow to decide how old can the results be. Here "w" means week so we only retrieve results on week old at worst.

From this query, all the documents are extracted with their title, a summary and a link.

2.1.2 YouTube

For Youtube, the "Data API" provides all the functionalities needed. It must be said that all the results are only in English. For each retrieved video, an associated function getStatistics allows to retrieve all the required values. This function allows to retrieve the number of views and the publication date for each video.

2.1.3 Flickr

For Flickr the java based API used is called Flickj-Android which provides support for Android and Google App Engine.

As specified before, for each document, each picture in this case, we retrieve the URL, the number of views and the Publication Date.

Since Flickr provides public and private photos, regarding our goals, only photos uploaded as "Public" by any user are retrieved.

It must be said, the FLickr API deals with two different dates, the Upload Date and the Taken Date. Since the Taken Date can be modified manually, some pictures have non sens Taken Dates. This being said, both dates have been set it up with minimal and maximal states.

3 The ranking algorithm

The ranking algorithm is splitted into two different rankings, the Passive ranking and the Active Ranking. Being passive implies something that doesn't involve an active participation. The Passive ranking is full based on default rankings and some statistics both provided by the API. Being active implies a certain commitment to action. The Active ranking is based on Tweets since posting one represents a proof of real interest in some topic.

In order to have a deeper understanding of this conception let's consider the following example. Since Passive Ranking is based on some statistics, the number of views for each document seems to represent a relevant information, but how trustworthy is this value regarding the relevance of the results? Watching a video on Youtube could be associated as a passive action because people can reach that video without necessarily a real commitment (It could be an associated video or just a random video that some person sent to another one).

When using Twitter, a Tweet represents a real commitment between the user and the topic posted, in that sense it's an active action.

3.1 Passive ranking

For the three different sources, the system only retrieves the result of the last 7 days.

Regarding the default ranking: For Youtube, the default ranking is based on the number of views per video. For Flickr, it is based on a parameter called "Interestingness". Finally, for Google News it is based on relevance.

Regarding the use of statistics: The API provides for each document the number of views and the publication date. The ratio between number of views and days of publication has been computed for each document. This coefficient after being normalized will be added to the default score.

3.2 Active ranking

3.2.1 The Twitter API

The active ranking uses twitter to rank documents. An implementation of the Twitter API in Java called Twitter4J is used to extract the information needed for the algorithm.

The search API allows to retrieve tweets that are less than 7 days old. In order to have a useful sample of tweets, 1000 tweets are retrieved for each query and some computation is done on this sample to extract useful information for the ranking algorithm.

3.2.2 Using Twitter to judge the "interestingness" of documents

In order to define which are the most interesting documents, the most "trending" words for the user's query are searched in Twitter. If these words are present in a document its ranking score will be inscreased.

Getting "interesting" words from Twitter

A large amount of tweets is retrieved (1000 in this case) using the search API with the user's query. The second step in then to concatenate all the tweets retrieved, to remove the words which are in the query, the stopwords, the punctuation and the links. With the "cleaned" corpus, the occurrences are counted for each words to determine the top 10 words.

Ranking system

A score is then given to each of the ten words according to their number of occurrences on a scale of 100 :

$$\text{Score for a word} = \frac{\text{Occurrences of the word}}{\text{Total number of occurrences of the 10 first words}} \times 100$$

When all the scores are defined, each one of these 10 words are searched in the description or in the title of the documents. For each word found in the description of a document, the corresponding score will be added and if it is in the title, we will add two times this score.

3.2.3 Using Twitter to define the most relevant vertical

One big problem in aggregative search is to define the percentage of each verticals for a query. Indeed, is it possible to know if a user wants more videos, more images or more text. The user maybe doesn't even know what she wants at the beginning but will see in the results if one vertical is better than an other for her query. To try to solve this problem, Twitter is again used. Indeed, according to Twitter [6], in 2010, 25% of tweets were containing links and with most of the query tested in this project, it was far more. So these links will be analyzed to know if they lead to an image, a video or something unknown.

Getting the verticals percentage

1000 tweets are again retrieved from the user's query and analyzed. The links containing these following words : "youtu", "video" or "vimeo" are considered as referring to a video. The same has been done for images with the following list of words : "img", "pic", "flic", "photo", "instagr", "yfrog". All the remaining links are considered as "unknown". As a lot of links in twitter are shortened, it is impossible to know if they lead to a video or an image in a reasonable time. After some test, it has been decided that the number of unknown link could be divided by 5.

Ranking system

it is then possible to set a score for each vertical :

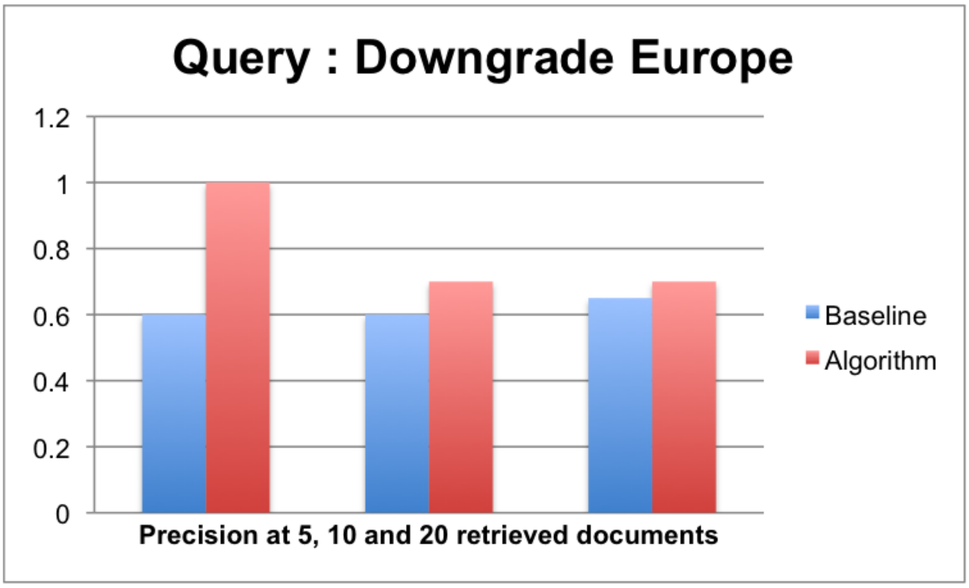
$$\text{Score for a vertical} = \frac{\text{Occurrences of the vertical}}{\text{Occurence of images} + \text{Occurence of videos} + (\text{Total unknown links})/5} \times 100$$

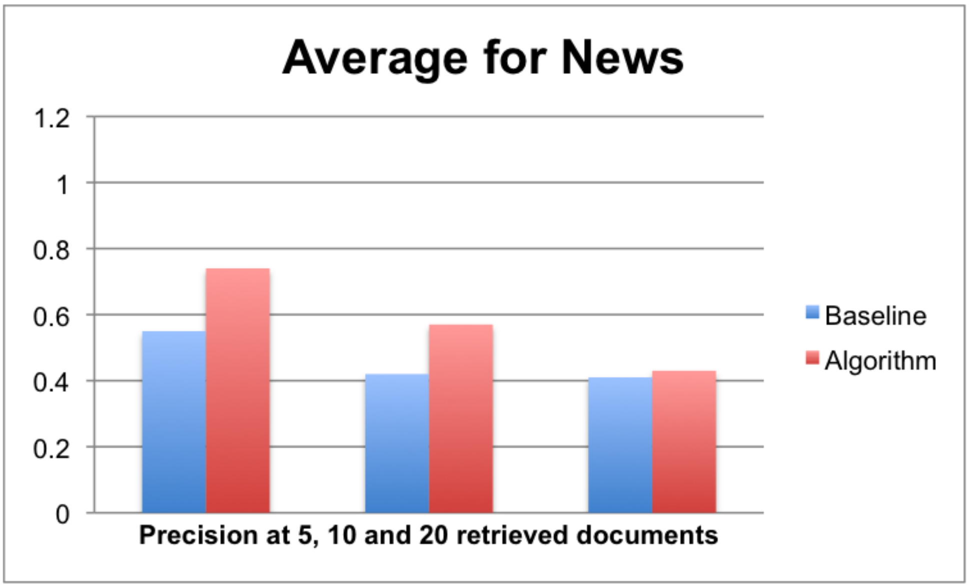
When all the scores are defined, the corresponding vertical score is added to each videos and images. Nothing was added to the text documents because they are already advantaged by the "interestingness" ranking because they contain more words than for videos or images. This ranking will therefore give a better balance to the results.

4 The evaluation

This part is not written yet....

[6] http://techcrunch.com/2010/09/14/twitter-seeing-90-million-tweets-per-day/





5 Discussion and Conclusions

5.1 Discussions about the system

During the use of Twitter in the algorithm, when finding interesting words, the first words found can be "obvious" and therefore not meaningful for the currents topics about the query. For example when the user search for "Obama", the first two words being returns will be "president" and "barrack" so results with these words will have a higher score. But it's not giving any information about the news related to "Obama". So it's better if the user enter "president barrack obama" as a query, the results are improved in this case.

5.2 Difficulties encountered

Lack of possibilities for Google News Because there is no API provided for Google News, we had to use an RSS feed. But this RSS feed lack in information about the documents. For example, we can't have information about the number of views of a news like for YouTube or Flickr.

Lack of information in a Flickr image With Flickr, most of the images don't have any descriptions and sometimes, the title is not very explicit. So it was really difficult to find appropriate content using this source. Maybe it would be better to use another source better suited for this kind of search.

Twitter links As we have seen in our algorithm we used Twitter links to rank the documents. However, most of the links published using Twitter are shortened and therefore it is impossible to know where this link is leading. We could have found the real link behind the shortened ones but it would have took too much time and the results would not have been given to the user in an acceptable time.

The system is slow Our system is for the moment a little bit slow and take around 10seconds to deliver the newspaper. As everybody is used to have instant search like on Google, Yahoo or Bing, it is hardly acceptable. Probably some optimizations in the code could have been made to solve this issue.