

Collaborative Search Engine Extension

Andrei TOMA

Economic Informatics Department,
Academy of Economic Studies, Bucharest, Romania
andrei.toma@ie.ase.ro

Abstract: *The present paper defines a possible approach towards alleviating the problems stemming from automatic indexing of web pages by introducing a collaborative mechanism as moderation between close results. While classical metrics are used for indexing, these are not the mechanism which grants the results cohesion, but rather the collaborative correction mechanism.*

Keywords: *collaborative, search engine, indexing*

1. Introduction

While classical mechanism based on statistical processing have been used for a significant period of time with generally good results, they are vulnerable to a series of attacks. While all these attacks are detrimental to the objective of obtaining accurate search results, some of them are considered to not pass the tolerance limit imposed by the search provider and have generated a separate technical field, search engine optimization.

Considering, however, that the objective of the search provider is to deliver the most relevant pages to the user, a method which moderates the statistical data (which can be manipulated as stated above) should be considered.

This paper considers a special case, that of a product search engine, however, the mechanism can be extended to other data e.g. to indexed word data from web pages or keyword meta data.

2. Collaborative systems/crowd generated content

Collaborative systems are systems designed with the purpose of helping people achieve common goals through a mechanism involving collaboration. Collaboration may involve parallelism of the actions performed by the actors involved or intersection of the actions. Through parallelism I understand that each actor interacts with a specific part of the system and delivers the results to the other users, while intersection assumes that the components themselves are the result of the collaboration.

Although a collaborative system can be any system aimed towards collaboration between the actors, the most interesting kind of collaborative system is what I will call "crowdware". Through crowdware I understand a system where the components themselves and the partially the functioning of the system are determined through the interaction of the users.

3. Search engines, between automated and collaborative

In designing search engines, two approaches stand out. One of them is ranking results

based strictly on statistical indicators calculated according to a series of metrics applied to the entities that are to be searched. Using purely this approach means failing to integrate any data relating to the actual attitude of the users towards the results and thus an inability to adjust the results dynamically in time. The second major approach is to design a human search engine, based purely on the ranking that users associate to a certain page. This is also an inefficient approach due to the fact that the users who rate the search results might have externally determined bias towards the success of certain results.

A hybrid approach is to include user input by indicators which mediate their insertion into the ranking data. Such an approach is the page rank algorithm [4] which is the basis of the Google search engine. While one might design more effective methods of introducing the user input, the most important lesson to be learned from the Google implementation is that user generated data cannot be introduced directly by the user, but rather processed in such a form that the user has the least chance of affecting the general outcome on purpose.

While a system which would actually ask the user to confirm the results would, in the absence of human moderation, be exposed to malicious users, an indirect confirmation by the user would alleviate this problem.

Such is the proposed approach, which relies on the documents that users click after a search as a means of confirmation of the search results. Since the effect that the clicks have on the result is not transparent to the users themselves, one could avoid a user purposely affecting the results as long as brute force attempts are made impossible. If automated clicks are detected by the system, then the user can confirm the results by his/her clicks and a human attempt at deforming the results would fail because the effect would be too small by comparison to users who are just searching for information.

4. Search engine metrics

For our model search engine we will use a series of selected metrics which will be applied to the searchable data [1]. While these metrics are not necessarily the ideal ones and the quotients necessary for their composition will not be discovered in this article, the click correction mechanism can be applied no matter what metrics are used.

The above being said, there is a number of possible metrics that have relevance when classifying documents according to the words they contain. For example, one can use as metrics statistical metrics such as the position of the word relating to the start of the document. When applying this metric to a multi word query, the value will be that of the average distance to the start of the document. When the document is divided between a series of different fields, the average will be weighed with the importance that is associated to the respective field.

Another useful metric is the distance between the words in a particular document, when the query is multi term. This particular metric has no significance for single word query, where it will not contribute to the final score.

A final indicator that is used in the calculation of this somewhat basic indicator of word importance in a document is the frequency rank, which considers that a word is more important when it has a high number of appearances in a document. This indicator will be regarded as a product of the individual frequencies in the case of a multi term query. As an observation, common words such as “the” will have to be omitted for this indicator to have any significance. Moreover, the results are more significant if digram or trigram data is used instead of individual frequencies.

Since the present article seeks to present the mechanism of click correction, no other indicators are used and no user data is included at this stage.

I have to stress that the click correction mechanism would allow for the search results to converge towards a useful set of values even if the frequencies were randomly selected. However, in order to obtain useful results in reasonable time, a composite indicator of the metrics described above should be a reasonable estimation.

Such an indicator includes all the indicators above in a weighed form and with the proper signs (since some metrics define relevance as a positive value and others as a negative value). The form of the indicator would thus be:

$$R_w = w_0 * R_f + w_1 * R_p + w_2 * R_f$$

where

R_w is the total weighed ranker

R_f is the frequency ranker

R_p is the position ranker

$w_{0,1,2}$ are the respective weights

5. Click correction

The search engine in the form described in the previous section has no possibility of adjusting to user preference, nor any kind of mechanism through which the users themselves can correct errors in the rankings.

The most flexible way of adjusting the results is through a collaborative mechanism, the principle of which is that the user is the one tasked with the validation of the search results.

There are two readily apparent approaches to this problem

- Adjusting the calculated scores on each click that confirms a result. This involves ascertaining which of the presented items the user prefers and adjusting the score of the item upwards while all the other items are adjusted downwards. The quantity with which the scores are adjusted should be more significant in the case in which the score is adjusted upwards. The reason for this is that while a user might prefer a certain item, there is no reason to believe that he/she considers all the other items wrong results. Adjusting the scores will be done through the simple formula:

$$s_{new} = s_{old} + d$$

where d can be either positive or negative as stated above. The value of d for each of the two cases will be determined experimentally. One disadvantage of this approach is that the scores will have to be stored and cannot be calculated “on the spot”.

- The second approach is to represent the scores as a feed forward neural network [2][3] which has as input values the statistical scores and as outputs the search results. This approach allows for greater flexibility as it is likely to produce more accurate results even for the pages which have not yet been accessed by sufficient users. The training of the network is continuous as any user choice is fed back into it.

6. Automated clicker avoidance

One precondition for the system to produce the desired results is that automated clicks should not be possible in order to avoid the circumstance where a malicious user tries to skew the results by simulating confirmations. Such a system should rely on trying to ascertain if the engine is dealing with a machine or a human. There are a number of approaches such as relying on the time interval between user clicks which is more likely to be random for a human being and regular for a machine.

7. System modeling

A possible model of the data involved in the system is presented below, for the case where the indexed documents are actually product descriptions:

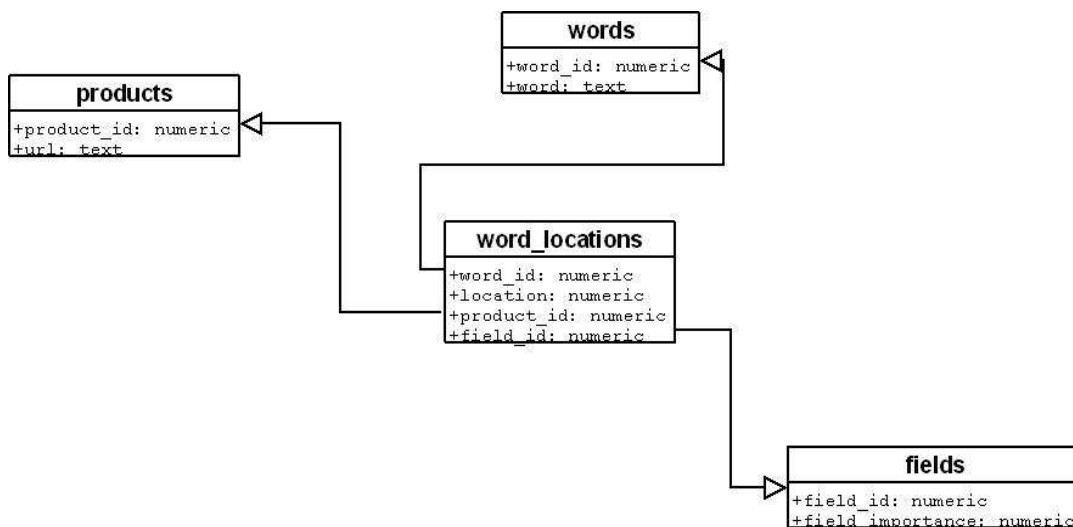


Fig. 1. Model of the data involved in the system

Some clarifications about the data structures are needed.

The *word_locations* entity stores the locations of the words in the documents

- *product_id* stores the respective searchable item
- *word_id* the word in the stored location
- *field_id* the document field in which the word is located, for the event that the documents are divided in distinct fields
- *location* the position of the word in relation to the beginning of the containing field

The *products* entity stores the id's of the products and the possible relevant information relating to each product and has the role of allowing possible extensions to a real life situation.

The *fields* entity stores the fields defined in each document entity. Documents can be preexistent, such as web pages, but can also be constructed dynamically from the available data.

The *words* entity stores the words encountered in the data, associating the actual word to a word id.

The classes needed for implementation of the simpler, statistics based ranker is presented in the following diagram:

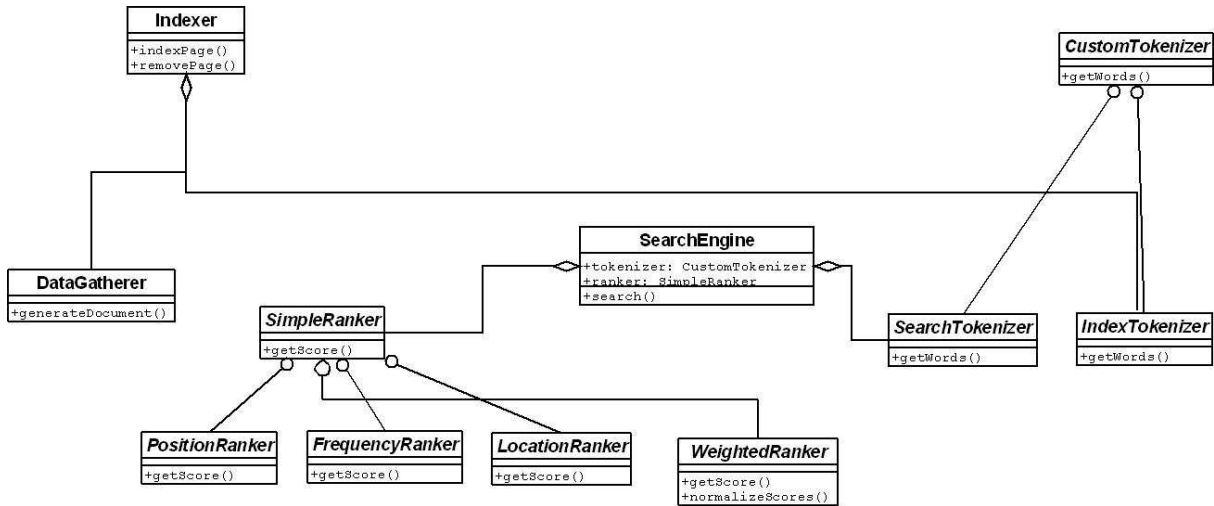


Fig. 2. Classes needed for implementation of the simpler

The model contains a series of interfaces and classes which are described below:

CustomTokenizer is an interface which establishes the fact that a tokenizer has to have at least a `getWords()` method.

SimpleRanker is an interface which is implemented by all rankers and establishes the fact that all rankers have to have a search function.

DataGatherer handles the generation of artificial documents based on the available data. If the data is already organized in documents, it will simply deliver them to the classes in charge with the actual processing.

The *Indexer* handles the adding and indexing of documents. Also, it has the role of eliminating the documents that have changed or have disappeared. Its methods should be called anytime a document change occurs. Adding a document to the index is done by constructing an artificial document (which is handled by the *DataGatherer* class) from the available data, tokenizing the fields and adding the word locations to the locations entity as well as the product to the products entity. New words will also be added to the words entity, while new fields will be added to the fields entity.

SearchEngine implements the actual search mechanism. The search is done by tokenizing the search string and calling a ranker.

A ranker constructs scores for documents by a series of criteria. *PositionRanker* is a ranker used to test the results obtained based on the position of the search terms relating to the beginning of the document. The position can be calculated by weighing the actual position within a field with the importance which is associated to the field. For a multi word search, the average of the positional scores is returned.

Frequency ranker is a ranker used to test the results obtained based on the frequency of the words. Frequencies are calculated on the totality of the fields. For a multi word search the return value is calculated considering the probability of apparition for the words independent (the product of the frequencies).

Distance ranker is a ranker used to test the results obtained based on the distances between words. This ranker has no significance for single word searches. The returned score is based on the minimal distance found in any field between the search terms. The distance between two words is the module of the difference between their respective positions.

WeightedRanker is a composite ranker. It uses all the above mentioned methods, weighing the scores and inverting the sign where needed (some scores are better when they

are larger, some when they are smaller).

Possible implementation:

If the first approach is adopted, correcting the scores with a given value on click validation by a user, the system remains largely unmodified.

If, however, a neural networks based approach is adopted, the system has to be extended. Such an extension will be presented in a future paper.

This second approach also has the advantage of considering the click relevance in a larger scope and thus alleviating the possible problem of the system being exploited for comparatively irrelevant sets of search results.

8. Conclusions

The present paper contains a new approach towards integrating both user feedback and statistical data in generating search engine results. The advantage is that, as long as automatic clicks are detected, the users will most likely contribute to the system by their actions. This presents some advantages between a thumbs up/thumbs down approach (where the user is asked explicitly if the results were accurate), as there is no need for human moderation of the feedback.

The second approach towards correcting results via user clicks will be detailed in a future paper, as it improves significantly on the first, simpler one.

References

- [1] Programming Collective Intelligence: Building Smart Web 2.0 Applications, Toby Segaran , O'Reilly, 2007
- [2] Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems), Ian H. Witten and Eibe Frank, Morgan Kaufmann, 2005
- [3] Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems) , Jiawei Han and Micheline Kamber , Morgan Kaufmann ,2005
- [4] The anatomy of a large-scale hypertextual Web search engineComputer Networks and ISDN Systems, Volume 30, Issues 1-7, Pages 107-117, Sergey Brin, Lawrence Page, 1998

Authors



Andrei Toma has a background in both computer science and law and is interested in an interdisciplinary approach to IT Law related issues. He has graduated the Faculty of Cybernetics of the Academy of Economic Studies in Bucharest and the Faculty of Law of the University of Bucharest. He is currently conducting doctoral research at the Academy of Economic Studies. His fields of interest include IT Law related issues, as well as various artificial intelligence topics.