Intelligent Search Engine Ranking Algorithm inspired by Recommendation Engines

Ganesh Venkataraman
Sri Venkareswara College of Engineering, Anna University
Pennalur, Irungattukottai – 602 117
Tamil Nadu, India

ABSTRACT
Every step in the evolution of human kind is associated with the inherent quest for knowledge and substantial growth in intelligence. In the modern world, the thirst for information is quenched by search engines that crawl billions of pages on the World Wide Web. This paper endeavors to make the ranking of the indexed web pages more intelligent by using techniques followed by recommendation engines that, with the help of some algorithms, recommend products on e-commerce websites. The focus primarily lies on discovering user groups, finding the degree of similarity between users based on search queries and building a graph that tracks the clicks on search results within the group, enabling the machine to learn which result might meet the expectation of one particular user and rank the results accordingly.

General Terms
Intelligent systems, Algorithms, Machine learning, Web systems, ranking algorithm.

Keywords
Search engine, ranking algorithms, intelligent ranking, recommendation systems.

1. INTRODUCTION
Machine learning is a powerful tool to make web applications more intelligent. The idea of ranking web pages by following recommendation techniques is based on the principle of Collaborative filtering [1]. According to Toby Segaran [2], a collaborative filtering algorithm usually works by searching a large group of people and finding a smaller set with tastes similar to yours. It looks at other things they like and combines them to create a ranked list of suggestions. In this context, 'similar taste' is similar search queries and 'things they like' correspond to the result that people click on. In a nut shell, intuitively, it can be said that recommended results are in some way more meaningful to the user.

There are basically three stages in search. The first step in creating a search engine is to develop a way to collect the data and this is called crawling. This stage is followed by Indexing, where the crawled data are stored in databases. The final step is returning a ranked list of documents from a query.

2. EFFICIENT RANKING
Ranking is the process of giving pages a score for a given query, as well as the ability to return them with the highest scoring results first. The ranking process of a modern, efficient search engine consists of several stages. Effectively, it can be considered three fold.

2.1 Basic (primary) ranking
This content-based ranking is the type of ranking which was used by search engines in the early stages which takes into consideration the word frequency, document location and word distance.

2.2 Secondary Ranking (link analysis)
There are various ranking algorithms that rank pages based on the number of incoming links like the In-degree, SALSA, Pagerank etc. [3]. Of these algorithms, the Page rank algorithm used by Google is the most popular one which is described as follows [3], [4]:
Let u be a web page. Then let Fu be the set of pages u points to and Bu the set of pages that point to
u. Let Nu = |Fu| the number of links from u and let c be a factor used for normalization (so that the total
rank of all web pages is constant). We can now define a simple ranking R which is a slightly simplified version of the actual PageRank:

$$R = c \sum_{v \in Bu} \frac{R(v)}{Nv}$$

This algorithm assigns every page a score that indicates how important that page is. The importance of the page is calculated from the importance of all the other pages that link to it and from the number of links each of the other pages has.

2.3 Third stage (Taking clicks into account)
Conventional methods of taking user clicks into account include just remembering the query and counting how many times each result was clicked or more sophisticated methods like forming neural networks to track clicks are also employed.

3. FORMING GROUPS AND ESTABLSHING SIMILARITY SCORES
This part is the prime focus of this paper. While performing the third stage of the ranking process, rather than just counting clicks all over the world, people are grouped together based on some factors and clicks by similar people are made to add more weight to the final score of the results by mimicking the operation of recommendation engines that recommend products bought by similar users.

3.1 Forming Groups

The total number of people accessing the World Wide Web is seemingly infinite. To make the process more efficient, people can be clustered based on Geo-location (or IP geolocation), MAC address and various other factors to form a finite set of closely knit web searchers. This step however is not necessary if this algorithm is to be implemented on a small network rather than the entire web. In a considerably small group, this algorithm is sure to perform efficiently.

3.2 Establishing the similarity scores within the group

Recommendation engines use various algorithms like Euclidian distance, Pearson Correlation etc. to assign similarity scores based on the products bought or rating given. The Pearson correlation method [5] can be effectively used to find similarity between people who search the web based on the queries they use. The correlation coefficient is a measure of how well two sets of data fit on a straight line. The formula for this is more complicated than the Euclidean distance score, but it tends to give better results in situations where the data isn't well normalized such as the search queries that the people of the group make over a period of time.

To visualize this method, you can plot the ratings of two of the critics on a chart, as shown in Figure 1. Superman was searched 3 times by John and 5 times by Ganesh, so it is placed at (3,5) on the chart.

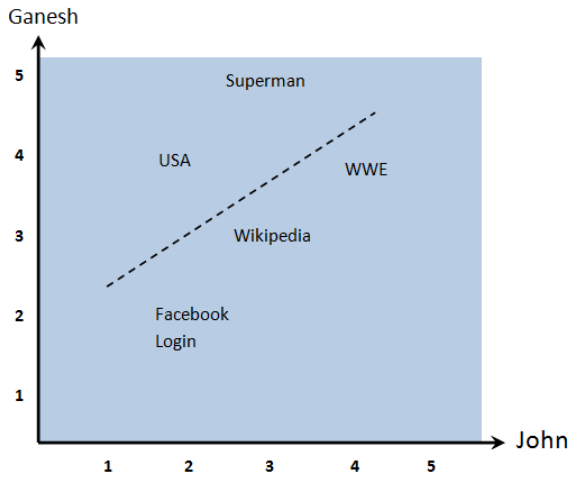

Fig.1: Comparing two people on a scatter plot based on the queries made.

The straight line found in the figure is called the best-fit line because it comes as close to all the items on the chart as possible. If the two people had identical search trends, this line would be diagonal and would touch every item in the chart, giving a perfect correlation score of 1. The algorithm for the Pearson correlation score first finds the queries searched by both people. It then calculates the sums and the sum of the squares of the number of searches for the two people, and calculates the sum of the products of their number of searches. Finally, it uses these results to calculate the Pearson correlation coefficient as shown. This formula is not very intuitive, but it does tell you how much the variables change together divided by the product of how much they vary individually.

A sample python code for calculation of Pearson coefficient is as shown below:

```
# Returns the Pearson correlation coefficient for p1 and p2
def sim_pearson(search,p1,p2):
 # Get the list of mutually rated items
 si={}
 for item in search[p1]:
  if item in search[p2]: si[item]=1

 # Find the number of elements
 n=len(si)

 #if they are no ratings in common,return 0
 if n==0: return 0

# Add up all the number of searches
 sum1=sum([search[p1][it] for it in si])
 sum2=sum([search[p2][it] for it in si])

 # Sum up the squares
 sum1Sq=sum([pow(search[p1][it],2) for it in si])
 sum2Sq=sum([pow(search[p2][it],2) for it in si])

 # Sum up the products
 pSum=sum([search[p1][it]*search[p2][it]for it in si])

# Calculate Pearson score
num=pSum-(sum1*sum2/n)
den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-
pow(sum2,2)/n))
if den==0: return 0
r=num/den
return r
```

This function will return a value between –1 and 1 which gives the similarity score between two people.

4. LEARNING FROM THE CLICKS AND RANKING THE RESULTS

Now that the finite group of people with every person having a similarity score with every other person has been formed, ranking based on this similarity can be done. It is to be noted that the ranks of results for every individual will be different, similar users ending up with mostly the same results at the top.

4.1 Learning from clicks

In the case of a search engine, each user will immediately provide information about how much he likes the results for a given search by clicking on one result and choosing not to click on the others. This section will look at a way to record when a user clicks on a result after a query, and how that record along with the similarity score can be used to improve the rankings of the results. For this purpose, a weighted edge graph with two layers can be used. In this case the first layer is a combination of inputs is a set of words, so you could also think of this as the query layer. The second layer is the output layer. Figure 2 shows the structure of the network. All the nodes in the input layer are connected to all the nodes in the output layer.

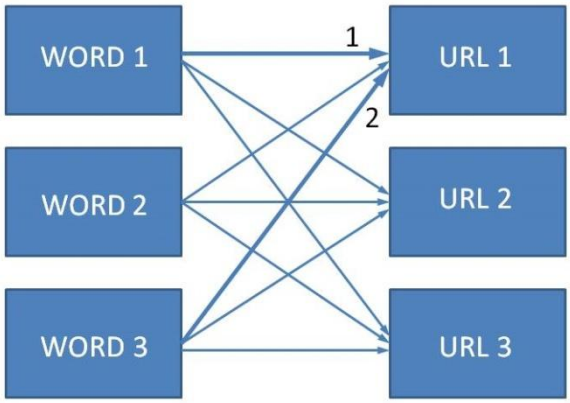

Fig2: Structure of click tracking network.

The weights of all the edges are made 0 initially. Over a period of time, the engine is made to learn which result is best for which user by adding corresponding weights, depending on the clicks by similar users, to the edges connecting the query node and the clicked result node. This can be called the modified backpropogation algorithm as this is similar to the backpropogation algorithm [6] used to train neural networks. It is again important to note that every user will have different weights of the edges. These can be stored against user accounts in big networks, centralized server for a small network of users or as cookies in the user's computer.

When a result is clicked by a person p1, the weight of edges connecting the query and the URL in another person p2's graph is added with the similarity score of p1 and p2. Thus the weight of these edges will be more if the two are more similar. For example, consider that Fig 2 represents p2's graph. If the word1 and word3 were person p1's queries and url1 was clicked, weights of edges marked 1 and 2 are incremented with Pearson Score (p2, p1).

4.2 Scoring and ranking the results

Now that the edges are weighted according to the number of clicks and the 'similarity value' over a period of time, effective ranking can be done based on this. When a query is now made, a forward scoring algorithm can be implemented to rank the pages. Every URL node will have a score calculated by the following formula

$$W = \sum weights\ of\ edges\ connecting\ the\ word\ and\ the\ URL$$

Note:

The net weight W can be positive or negative because the Pearson coefficient ranges between 1 and -1.

$$\text{Score(U)} = 1 - \frac{1}{1+\text{W}} \quad \text{if W} > 0$$

$$\text{Score(U)} = -\left\{1 - \frac{1}{1-\text{W}}\right\} \quad \text{if W} < 0$$

This returns a score between 1 and -1. The more the score, the higher should be the rank of the result. This along with the scores of the pages given by other ranking algorithms like the page rank algorithm should give more meaningful results at the top of the results page.

5. CONCLUSION

Thus, similarity between users is calculated and the search results are ranked based on clicks by similar users, making the results appearing on the top of search results more meaningful. The drawback is that every user has a different click tracking graph which might cause storage issues in big networks like the web on the whole. In smaller networks however, implementation of this algorithm will prove to be effective. With the enormous increase in today's computing power, maintaining individual graphs in bigger networks should not be a problem in the near future.

6. REFERENCES

[1] Algorithms of the Intelligent Web, Haralambos Marmanis and Dmitry Babenko,

[2] Programming Collective Intelligence- Building Smart Web 2.0 Applications, Toby Segaran, 2007

[3] A Survey of Ranking Algorithms, Alessio Signorini, Department of Computer Science, University of Iowa, September 11, 2005

[4] S.Brin, L.Page, The anatomy of a large-scale hyper textual web search engine, Proceedings of the 7th International World Wide web Conference, 1998

[5] Pearson's Correlation Coefficient: http://faculty.uncfsu.edu/dwallace/lesson%2017.pdf

[6] The Backpropogation algorithm. http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf