Disambiguation of User Search Query

Udita Gupta, Puja Jagani, Darshan Hundia, Khushali Deulkar
Dwarkadas J. Sanghvi College of Engineering
Mumbai, India
uditagupta93@gmail.com, puja.jagani93@gmail.com,
darshanhundia101@gmail.com, kushali.deulkar@djsce.ac.in

ABSTRACT: *With the advent of World Wide Web, anyone can obtain the information they want by typing in a query to the search engine. But every human has a different thought process and the results obtained from the search engine may not match their expectations. This happens due to ambiguous queries. This creates the need to personalize the search results in accordance with the user preferences. To overcome the problem of ambiguous queries, we propose a new approach to collect the user preferences. In the proposed approach, clickthrough data forms the basis of capturing the user's interest. Clickthrough data is analysed using three factors: the URL clicked, the parent domain of the URL and the category to which the clicked URL belongs to. All the three factors aid in creating a dynamic user profile which is updated using implicit feedback. To display the results, each result from search engine is assigned weight according the user profile and the results are displayed in descending order of the weights. Thus, results which interest the user are displayed at the top. We show a detailed case study of how the proposed approach works and also demonstrate its effectiveness in comparison to Google search results.*

Keywords: Search Engine, Clickthrough, Web Personalization, User Profile, URL Categorization

Received: 30 December 2014, Revised 28 January 2015, Accepted 5 February 2015

1. Introduction

With advancement in internet, various kinds of facilities are available to the end user which are just a mouse click away. One of the most important resource available online is the large amount of data presented as meaningful information. Search engines facilitate in fishing out this information as per the user demand. Search engines employee sophisticated algorithms to obtain the relevant results according to user query by using indexing and ranking. Some search engines also provide suggestions as the user types his query. But the suggestions or the results presented to the user may or may not meet his needs.

The problem arises when the query entered by the user is ambiguous. A study conducted showed that 16% of queries are ambiguous [1]. Ambiguous queries are queries have more than one meaning. This could be due to polysemy and English vocabulary morphology [2], for example the word "crane" can mean a bird or the machine use to lift. It could also be due to named entities, for example the word "apple", it can mean a fruit or the well reputed company. There are multiple such ambiguous queries. So if a user enters such an ambiguous query word, the results obtained may not be according to the meaning he intended. Irrespective of the cause of ambiguity, it is important to deliver proper results to the user.

Most traditional search engines provides same results for a query without considering the user's interest. This is where web personalization comes into picture. Understanding the user's query word and its context plays an important role to resolving ambiguous query. This can be done by collecting user preferences and his interests. Capturing user's interest is a problem in itself. Explicit feedback is an option but it is not used since users generally are not willing to specify their interests. Some systems may require a user to login and create an account to maintain his user preferences. This can be tedious. Thus, using implicit feedback is preferred over explicit feedback.

Implicit feedback means obtaining user preferences without asking the user explicitly. One such technique is clickthrough data. When a user clicks on web snippet, which contains the topic and brief description, it indicates that the user in interested in it. Hence, using clickthrough data to obtain implicit feedback has proved to be reasonably accurate on an average [3]. It is free of cost to obtain and demonstrates the user's natural use of search engine. In addition to this, it is distinct for each user [4]. Thus, clickthrough data forms the basis of our proposed approach to create the user profile.

Most existing user profile strategies, focus only what user is interested in but not in depth. For example, a user may be interested in growing apples, little interested in nutritional facts about apples and least interested in products of Apple company. A good user profile builder should consider what user likes and what user dislikes. The proposed approach captures user's both positive and negative preferences. This helps in recording user preferences in finer aspect.

2. Related Work

Here we describe the related work that has been done in this domain.

2.1 Personalization using ODP categorization

ODP stands for Open Directory Project. It is an Open Content Directory of the World Wide Web links. ODP categorizes [5] the various manually added Web Pages. Each category has many web pages linked to it. In this approach [5], the audience is known beforehand. Since the users are well known, the user's interests are recorded into various categories. These interests are used to build a user profile for each user. The search results are retrieved for the submitted query and then re-ranked according to the user's constructed profile. Each search result's category is obtained using the ODP. The approach then maps the user profile's categories to the categories retrieved from ODP and then finally re-ranks the search results.

2.2 Personalization using Personalized Ontologies and Web Page Classification

In this approach [2], the user profile is represented as a taxonomic ontology. Here they mainly attempt to improve the search result's presentation i.e. they are trying to re-rank the retrieved results. The post-retrieval [6] algorithm uses web page classification to organize the search results into a meaningful and personalized hierarchy [6] of results. Here the user is given the choice of creating his own hierarchical ontology based on his interests using a tool [6]. The ontology is supported by sample textual documents in the form of web documents to describe each concept in the formed concept hierarchy. The documents are classified using Multinomial Naïve Bayes (MNB) [6] into a hierarchy according to the user's perspective.

2.3 Importance of Categorization of the Web Pages

Categories can be very helpful when a search result which is relevant to the user's need is ranked at the bottom. It can get very difficult for the user to search through long lists of search results in order to fit his or her needs. Categorization [7] of the search result documents is done by using the word or phrase frequency count. The various stop words are removed and the most occurring phrases/words [7] are displayed as categories on the side for the user to browse through easily in case the search engine's result ranking fails to satisfy the user's needs. A system called Findex [7] was implemented using this approach and it was tested by giving it to 16 users for two months. The results showed remarkable progress and easy browsing for the users.

2.4 Concept based Results Categorization

Query expansion [8] is used here to retrieve results which are more specific. Here, UMBEL [8] is used to categorise the retrieved search results into concepts. For every search result retrieved, it is augmented with its UMBEL concept. On the client side, all the search results which have the same or similar concepts are grouped together to form a concept lens [8]. This means that each concept lens represents search results of similar concepts. Once a user selects a concept lens according to his interest, an automatic personalization [8] is achieved. Even inside each concept lens, the results are re-ranked to achieve further personalization.

3. Architecture

The user enters the query into the search bar. The query is send to the traditional search engine. Each result obtained from the search engine is given to the re-ranker component. It accesses the user profile to obtain the user preferences and re-calculates the weights accordingly to display the final results. The user clicks on the link which is relevant and this is captured as clickthrough data. This clickthrough data is analysed in User Profile Builder on basis of three factors: clicked URL, parent domain and the category of the webpage. Next time when the user enters the query, his updated profile is used by the re-ranker to display the relevant results on top.

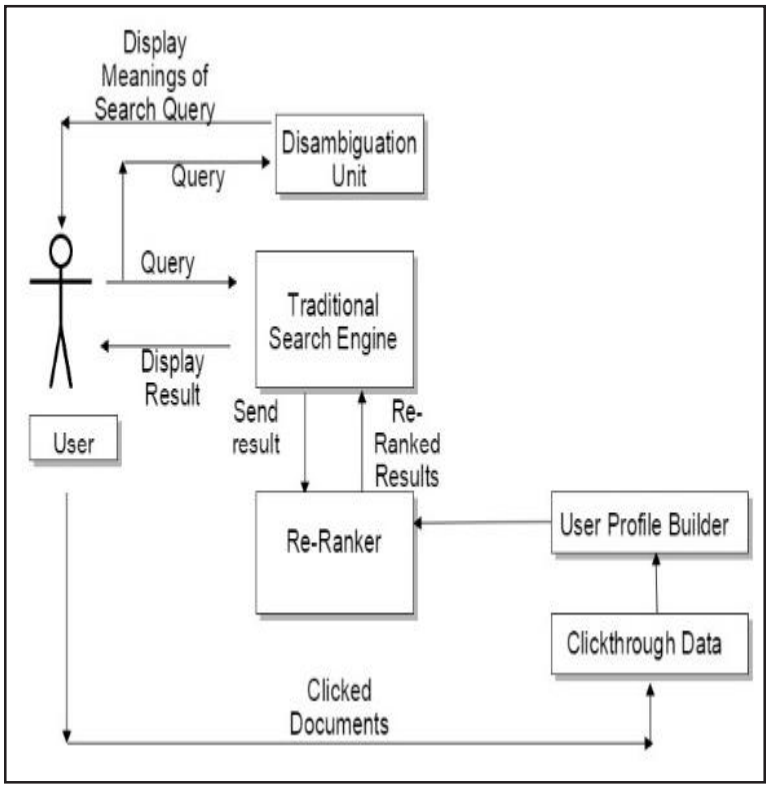

Figure 1. System Architecture

The components are described below in detail.

3.1 Clickthrough

Clickthrough data is a part of implicit feedback which makes it easier to capture the user's interests. It is an economical way to collect user's interest without causing an extra burden on the user. The table 1 illustrates clickthrough data.

According to the approach proposed by Joachims [9], a user browses the results retrieved from top to bottom. If the user has clicked on a result R_n at rank n, and has skipped the result R_m at rank m, when $m<n$, then it assumes that the user has scanned the result R_m but has decided to skip it. Thus, he is interested in result R_n. For each result clicked, we are analysing it using three factors to build the user profile.

3.2 User Profile Builder

The three factors are: URL clicked, parent domain of the URL and category to which the webpage belongs.

URL clicked is added to the user profile and the number of clicks on that URL is also stored. Parent domain of the URL is important because each user has a certain trust factor with each website. For example, when a user enters a query word "*orange*", one of the first results is Wikipedia page. Similarly, for other factual information, the user may prefer a Wikipedia page only. For a programming doubt, a user may prefer sites like stackoverflow.com and quora.com. Hence in case of a query related

to programming doubts, results from these two websites are preferred among the top results. Hence, we are using parent domain as one of the factors in deciding the user's preference.

Categorizing the clicked URL generates the domains which user prefers. These preferred categories are added to the user profile. However, web page categorization differs significantly from standard text classification in the following aspects:

1. Each web page has its own structure embedded into HTML.
2. There are various kinds of banners and advertisements on the web page.
3. There are different types of navigation bars.
4. Various hyperlinks are present.
5. Numerous visual contents are embedded like videos, audio files etc.

As we can see, there are various reasons as to why a normal text classification algorithm won't yield the correct results. Hence we are using a web summarization-based [10] technique. We use a Web summarization-based classification algorithm which is ensemble classifier using the improved summarization techniques. We extract all the relevant and important concepts [10] from the web pages and then pass them onto the standard text-based classification algorithm.

There are four different methods that we use for summarization [10] :-

1. An adaptation of Luhn's [11] summarization technique.
2. Latent Semantic Analysis [12] on Web pages.
3. Determination of the important content body as the basic summarization component [13].
4. Viewing the summarization as a supervised learning task.

After these methods, we combine them into an ensemble of various summarizers. Subsequently, we use this ensemble summarizer for the Web page summarization.

Using Web page summarization [10] we achieve the important features of a web page. Once we have extracted these features, we put them through a standard text-based classifier to get the classification results. Standard classifiers such as – Naïve Bayes [14] and Support Vector Machine [15][16][17] are used.

The following steps illustrate the algorithm:

Step 1: When the user enters the query, the retrieved results are temporarily stored.

Step 2: The user clicks on the results relevant to him. If the URL exists in the profile, its count is increased, else these URLs are added in the user profile database.

Step 3: Parent domain is obtained and saved in the user profile database. If the parent domain exists in the profile, its count is increased else the parent domain is added in the user profile database.

Step 4: The webpage category is obtained using the "*Ensemble Classifier*" [10]. If the category exists in the profile, its count is increased, else the category is added in the user profile database.

Step 5: Finally, the user profile is updated.

3.3 Re-ranking Component

The re-ranking component ranks the results in accordance with the user's interests.

The following steps illustrate the algorithm:

Step 1: For each result temporarily stored, check the following conditions:

a) If the URL exists in the user profile, then the weight is increased.

b) If the parent domain exists in the user profile, then the weight is increased.

c) Obtain the category for each result. If the category exists in the user profile, then the weight is increased.

Hence the total weight of each result is obtained. This step is repeated to calculate total weight for each result.

Step 2: Sort in descending order of the total weight.

Step 3: Display the re-ranked results.

3.4 Disambiguation Unit

The disambiguation unit generates various senses of an ambiguous query. Humans tend to respond faster to visual representation. Hence, an image is attached along with its description for each sense. This helps the user to identify the context of the query. The disambiguation unit uses the Wikipedia documents to obtain various senses. The Wikipedia library is used to identify if the word is ambiguous or not. If the word is ambiguous, we retrieve the various senses from the library.

4. Results

To analyze the effectiveness of our approach we worked on a large number of ambiguous queries. Various users' clickthrough data was collected and their results were personalized. The figure 2 depicts how the results were re-ranked for a particular query over a period of four weeks for a user.

The query "*galaxy*" has multiple senses. Initially, all results are ranked as given by traditional search engine. Top 16 results are considered since most users do not look beyond these results. Over a period of time, as user clicks on the results that interest him, the re-results are personalized. This is depicted in the graph. The query term "galaxy" belongs to categories like physics, biology, electronics, business, art and games. Initially, the results belonging to physics category are displayed on the top. After a week, few results from electronics category are displayed at the top along with other categories. After two weeks, the user profile showed that user's major interest lies in electronics. Hence all the results belonging to electronics are category are displayed on the top followed by the remaining results. Thus, user's ambiguity is resolved based on his interest.

We also analyzed the system for two users having different interests for the same query term "Crane". Based on his interest he can select the appropriate meaning displayed by the disambiguation unit. User A selects Crane (Bird) and User B selects Crane (Machine). The table below shows the number of relevant results obtained by just typing in Crane and after user re-fires the query from the suggested queries according to user's interest. Our system proved to produce more relevant results as per user's interest. Our system proved to produce more relevant results as per user's interest.

System/Criteria	Number of Relevant Results	Number of Irrelevant Results	Solving Vocabulary Problems
Google (Original Query)			
User A	5	11	NO
User B	8	8	NO
Google (User Re-fired Query)			
User A	13	3	YES
User B	12	4	YES

Table 1. Google Search Results for Original and Re-fired Query

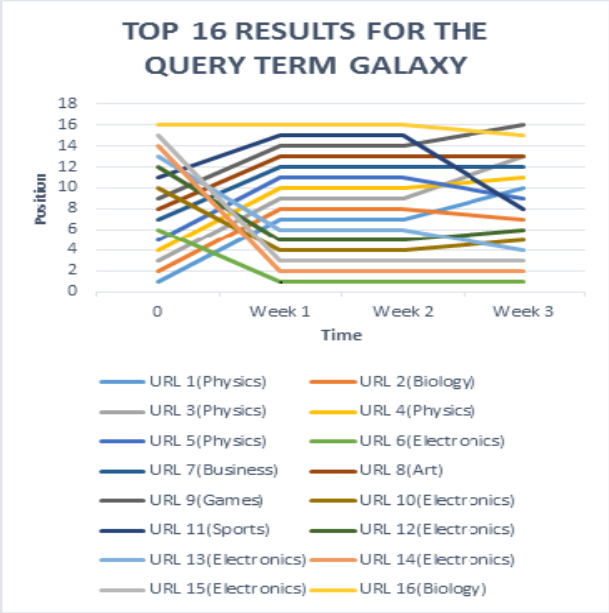

Figure 2. The top sixteen results for the query term "Galaxy"

5. Conclusion and Future Scope

Using clickthrough data we obtain the implicit feedback from the user. The user profile is created by using three parameters: the URL clicked, the parent domain and the category of web page. The final results are re-ranked on the basis of user's preferences from the user profile. For every query word, its senses in different contexts are displayed to resolve the ambiguity. Thus, enhancing the user's web search experience and making it easier for him to obtain the desired results.

There are several directions for extending the work in the future. First, when a user clicks on a web-snippet, it may or may not be relevant. So a method for calculating the degree of relevance would be helpful. Second, the user profile can be used to predict the intent of unseen query. When a user submits a near query, personalization can be used to suggest related queries. Even feedback from other users for related queries can be integrated into the user profile. Third, the amount of time a user spends on each webpage can be used as a parameter to obtain his preferences.

6. References

[1] Song, R., Luo, Z., Wen, J., Yu, Y., Hon, H. (2007). Identifying Ambiguous Queries in Web Search. In WWW '07: *In:* Proceedings of the 16th International Conference On World Wide Web, p. 1169-1170, New York, NY, USA, ACM.

[2] Dwivedi, A.P., Dwivedi, S.K., Mishra, A., Pathak, V. (2012). Impact of web query morphology and ambiguity on search engine's performance. *In:* Proceedings of the International Conference on Pervasive Computing and Communication, p.13-18. New Delhi, India.

[3] Joachims, T., Granka, L., Pan, B., Hembrooke, H., Gay, G. (2005). Accurately Interpreting Clickthrough Data as Implicit Feedback. In SIGIR '05: *In:* Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, p. 154-161, New York, NY, USA, ACM.

[4] Joachims, T., Radlinski, F. (2007). Search Engines That Learn from Implicit Feedback. *In:* IEEE Computer Society, 40 (08) 34-40.

[5] Ma, Z., Pant, G., Liu Sheng, O. R. (2007). Interest-based personalized search. ACM Transaction on Information System, 25 (1) Article 5, February. .

[6] Singh A., Nakata, K. (2005). Hierarchical classification of web search results using personalized ontologies. In HCI International '05: *In:* Proceedings of the 3rd International Conference on Universal Access in Human-Computer Interaction, Las Vegas, NV, USA.

[7] Kaki, M. (2005). Findex: Search result categories help users when document ranking fails. *In*: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, p. 131–140, New York, NY, USA.

[8] Sah, M., Wade, V. (2013). Personalized Concept-based Search and Exploration on the Web of Data using Results Categorization. *In:* ESWC.

[9]Joachims, T. (2002). Optimizing search engines using clickthrough data. *In:* Proceedings of ACM SIGKDD Conference, p.133-142, New York, NY, USA.

[10] Shen, D., Chen, Z., Yang, Q., Zeng, H., Zhang, B., Lu, Y. (2004). Web Page Classification through Summarization. In SIGIR '04: *In:* Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, p. 242-249, ACM.

[11] Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts. *In:* IBM Journal of Research and Development, 2(2), p. 159-165, April. .

[12] Gong, Y. H., Liu, X. (2001). Generic text summarization using relevance measure and latent semantic analysis. *In:* Proceedings Of the 24th Annual International ACM SIGIR, p. 19-25, New Orleans, Louisiana, United States.

[13] Chen, J. L., Zhou, B. Y., Shi, J., Zhang, H. J., Wu, Q. F. (2001). Function-based Object Model towards Website Adaptation. *In:* Proc. of WWW10, Hong Kong.

[14] McCallum, A., Nigam, K. (1998). A comparison of event models for naive bayes text classification. *In:* AAAI-98 Workshop on Learning for Text Categorization.

[15] Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. *In:* Proceedings of ECML-98, 10th European Conference on Machine Learning, p. 137-142.

[16] Vapnik, V. (1995). The Nature of Statistical Learning Theory. *In:* Springer-Verlag, NY, USA.

[17] Cortes, C., Vapnik, V. Support vector networks. Machine Learning, 20:1-25, 199