Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 5; April-June, 2015 pp. 407-410
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/acsit.html

Personalized Query Auto Completion Using Social Networks' Login

Anshul Kushwaha[1], Antra Katiyar[2], Inderpreet Kaur[3] and Ashish Kumar[4]

[1,2]B. Tech(CSE) Final Yr Galgotias College of Engg & Tech, Greater Noida
[3]Research Scholar, Mewar University, Chittorgarh
[4]B. Tech(CSE) Final Yr Galgotias College of Engg & Tech, Greater Noida
E-mail: [1]anshulkushwaha1@gmail.com, [2]antrakatiyar@gmail.com,
[3]kaur.lamba@gmail.com, [4]galgotia.ashish@gmail.com

Abstract—*Query auto completion in search engines aim to suggest the most accurate queries matching the few characters entered by the user. Present search trends and query logs are the commonly used criteria. This can be widely improved by incorporating user specific details from social network profiles. Most of the Social Networks allow access to the user info via apps that use their API. For example, Facebook offers user info by default which includes their name, gender, location and other details. This can be further extended to even more details regarding user activity on the social network. Such info can lead to a much more personalized and user-centric auto completion experience.*
We demonstrate the user details that can be of great value for QAC and the ways of getting these details from the most popular social networks out there. Later we provide a working model consisting of AOL query log, trending topics and Facebook likes.

1. INTRODUCTION

Query auto completion is one of the most visible features in Web Search today. It is offered by all major search engines and in almost all their search boxes. Query auto completion helps the user formulate her query, while she is typing it. Its main purpose is to predict the user's intended query and thereby save her keystrokes. With the advent of instant as-you-type search results (a la the Google Instant), the importance of correct query prediction is even more acute, because it determines the speed at which the user sees the suitable results for her intended search and the amount of irrelevant results that are displayed to her along the way. The basic principle that underlies most query auto completion systems is the wisdom of the crowds. The search engine suggests to the user the completions that have been most popular among users in the past (we call this algorithm MostPopularCompletion). For example, for the prefix am, Bing suggests amazon and american express as the top completions, because these have been the most popular queries starting with am. As the user is typing more characters, the space of possible completions narrows down, and thus the prediction probability increases. Clearly, during the first few keystrokes the user is typing, the search engine has little information about her real intent, and thus the suggested completions are likely to wrongly predict her query.

Social networks have become a vital and indispensable part of the life of the present generation. Most of the youngsters are addicted to sites like Facebook, Instagram etc. With the increasing use of social networks, their importance and data value is also increasing and so is the amount of user details available in them. User profiles are more and more like mirrors or rather windows to the user's personality, likes, interests and preferences. Thus a lot can be concluded about a user by viewing there social network profile.

This can be of great help for applications that aim to personalize the user experience. Most of the social networks enable developers to access user info via an Application Program Interface (API).

An API is a method of accessing the user details on the social network via an "app" of the social network itself. Such APIs mostly work by giving access of user info to an App via a Login button. Once the user logins on the site using their login credentials, she is given an option to deny or allow access to her details to the app. These buttons have an added advantage of easy account creation to the user. So it a win-win situation for both the user and the developer.

Once we have access to the user details and activity on the social network, we can extract valuable info including user's age, gender, location etc. These bits of information go a long way in creating a personalized user experience.

These Even more details like the user likes, interests, keywords including the celebrities, movies, shows etc. can be collected.
Query auto completion can be highly improved by taking these details into account while predicting the user queries. A simple method to do this is to cross check a larger list of possible suggestions with the user interests in hand. This can help in re ranking the suggestions and presenting a more user-specific list of results.

2. RELATED WORK

Query auto-completion. Auto-completion is being widely used in most modern text editors, browsers and search engines. In *predictive auto-completion* systems, the candidates are matched against the prefix on-the-fly using information retrieval and NLP techniques (Darragh et al.[1] ; Grabski and Scheffer [2] ; Nandi and Jagadish [3]). Bickel et al. [4] learned a linearly interpolated n-gram model for sentence completion.

In *pre-computed auto-completion* systems, the list of matching candidates for each prefix are generated in advance and stored in efficient data structures such as prefix-trees (trie) for fast *lookups*. As the user types more characters, the list of candidates is updated by exact prefix matching although more relaxed lookups based on fuzzy matching have been also explored (Chaudhuri and Kaushik [5] ; Ji et al.[6]). Once the matching candidates are filtered, they can be ranked according to different criteria. For instance, in an online store such as amazon.com, suggestions may be ordered according to price or review scores of products. In web search scenarios, the common approach is to rank suggestions according their past popularity. Bar-Yossef and Kraus [7] referred to this type of ranking as the *MostPopularCompletion* (MPC) model and argued that it can be regarded as an approximate maximum likelihood estimator. Given a search log of previous queries Q, a prefix P, and the list of query-completion candidates that match this prefix C(P), the MPC algorithm is essentially applying the following *Maximum Likelihood Estimation* for ranking:

$$MPC(\mathcal{P}) = \arg\max_{q \in \mathcal{C}(\mathcal{P})} w(q), \quad w(q) = \frac{f(q)}{\sum_{i \in \mathcal{Q}} f(i)} \qquad (1)$$

Here, *f*(*q*) represents the past query frequency for *q* in the Q logs. Shokouhi and Radinsky [8] later extended the MPC model and replaced the past frequency values *f*(*q*) in Equation (1) with predicted frequency values $\hat{f}(q)$. They showed that the predicted values produced by applying timeseries on query history are more effective for ranking autocompletion candidates. Strizhevskaya et al. [9] also modelled the frequency trends of queries by time-series for improving the auto-completion ranking.

In the context of query auto-completion, the closest studies to ours are done by Bar-Yossef and Kraus [7] and Weber and Castillo [10]. The *NearCompletion* method considers the user's recent queries as *context* and takes into account the similarity of QAC candidates with this context for ranking. Their hybrid model computes the final score of each candidate by linearly combining the popularity-based (MPC) and context-similarity scores. We go beyond session-based features and explore the effectiveness of considering users' age, gender, location and longer search history in auto-completion ranking. Further, in contrast to the NearCompletion model that uses a linear combination of two features for ranking, we propose a supervised framework for ranking, and define a novel objective function for optimizing it. It is also worth noting that while the NearCompletion approach is not applicable to single-query search sessions (more than 50% of traffic Jansen et al., [11]), our personalized model can be applied on all search queries. Weber and Castillo [10] mostly focused on showing differences in query likelihoods across different demographics. They briefly discussed a special case of auto-completion for predicting the second term in a query based on an unsupervised probabilistic model generated according to phrase counts across different demographics. They did not discuss the effectiveness of individual features (e.g. age versus gender), and did not report any results for more general cases where only the first few characters of queries are available. Their work is still based on *aggregation* over different demographic groups and they do not address how such features can be combined with other userspecific features (e.g. session history) in a unified framework for ranking auto-completion candidates.

Query Suggestion. Query suggestion and auto-completion are closely related. Apart from differences in *matching* that are largely imposed by stricter latency constrains in autocompletion, the main distinction is in the type of user input – a query in query suggestion and a prefix in autocompletion. Query prefixes are by definition shorter than submitted queries and hence they are more ambiguous. Thus, the potential for disambiguation using personalized features is arguably higher in auto-completion. Nevertheless, the problems are sufficiently similar that covering some of the related work on query suggestions might be worthwhile.

3. USING FACEBOOK LOGIN

The main title Facebook allows accessing a user's data via a Facebook App. This can be implemented on almost any platform including web, Android, Windows Phone or iOS very easily. All we need to do is to get an Authorisation and include a "Login with Facebook" button on our website/app.

Once the user taps on this button, he is redirected to Facebook to login and choose which information he wishes to share with our App. When a person logs into our app via Facebook Login you can access a subset of that person's data stored on Facebook. Permissions are how we ask someone if you can access that data. A person's privacy settings combined with what we ask for will determine what we can access.

Permissions are strings that are passed along with a login request or an API call. Here are two examples of permissions:

- Email - Access to a person's primary email address.
- User_likes - Access to the list of things a person likes.

3.1 Categories

Permissions are placed into categories that reflect how they are presented to people and the review process.

Some sets of permissions are more sensitive than others and people can opt-out of providing access to them, even if you ask. Optional permissions show up on a separate dialog during the login process, and the person can press the 'Skip' button to not grant your app permission to access that set of data.

Permissions That Do Not Require Review

Public profile (default) permissions. The default includes some basic attributes about the person, which are part of a person's public profile on Facebook. The default permissions are included as part of every permissions request, but require slightly different handling on the web and native mobile platforms.

App friends. This optional permission grants your app the ability to read a list of friends who also use your app.

Email permissions. This gives you access to the person's primary email address.

Permissions That Require Review

Permissions that require review are generally reviewed within 3 business days. Some permissions may take up to 7 days to review, and are marked as such in the reference.

Extended profile properties. These permissions are all sensitive properties that may or may not be part of a person's public profile.

Extended permissions. These include the most sensitive pieces of profile information. One of the these permissions is publishing stories to a person's Facebook profile. All extended permissions appear on a separate screen during the login flow so a person can decide if they want to grant them.

Open Graph permissions. These permissions are for gaining access to any Open Graph data stored in someone's profile.

Page permission. This permission allows you to administer any Facebook Pages that the person manages.

Public_profile (Default)

Provides access to a subset of items that are part of a person's public profile. A person's public profile refers to the following properties on the user object by default:

- Id
- name
- first_name
- last_name
- gender
- locale

Table 1: Some Useful Facebook Permissions

Permission	Used for
user_actions.books	Provides access to all common books actions published by any app the person has used. This includes books they've read, want to read, rated or quoted.
user_actions.music	Provides access to all common music actions published by any app the person has used. This includes songs they've listened to, and playlists they've created.
user_actions.news	Provides access to all common news actions published by any app the person has used which publishes these actions. This includes news articles they've read or news articles they've published.
user_hometown	Provides access to a person's hometown location through the hometown field on the User object. This is set by the user on the Profile.
user_interests	Provides access to the list of interests in a person's Profile. This is a subset of the pages they have liked which represent particular interests.
user_likes	Provides access to the list of all Facebook Pages and Open Graph objects that a person has liked. This list is available through the likes edge on the User object
user_location	Provides access to a person's current city through the location field on the User object. The current city is set by a person on their Profile.

4. USING GOOGLE LOGIN

The Google+ API is the programming interface to Google+. We can use the API to integrate our app or website with Google+. This enables users to connect with each other for maximum engagement using Google+ features from within our web application.

Many API calls require that the user of the web application grant permission to access their data. Google uses the OAuth 2.0 protocol to allow authorized applications to access user data.

Every request that the web application sends to the Google+ API needs to identify the web application to Google. We have to use an access token on behalf of user if we are making call.

For Google+ API calls that do not need to identify a particular user, an application API key can be used. This is useful for server-side applications, or web applications that do not require the user to sign in with Google.

Google supports *incremental authorization*, which enables your app to request initial permissions at sign-in and later can request additional permissions, typically just before they are needed.

A valid successful login with Google+ API provides certain recommended scopes such as access to the social features of the user.

This is the recommended login scope providing access to social features. This scope implicitly includes the profile scope and also requests that your app be given access to:

- The age range of the authenticated user
- The list of circled people of the user.
- The methods for reading, writing and deleting app activities (moments) to Google on behalf of the user.

After user had signed in with Google, we can access the user's age range, language, public profile information, and people that they have circled. If app requests the plus. profile. emails. read scope, we can also get the user's email address. With this rich social data, we can build engaging experiences and an instant community in the web app.

By using the above scopes provided by the Google+ API we can provide more relevant and accurate suggestions to the user.

5. IMPLEMENTATION

We have implemented this method by coding a query auto completion demo using HTML/CSS, Javascript, PHP and MySQL. Our project uses four criteria for suggesting queries-

- Most Popular Completion (AOL query log)
- Trending topics (hawttrends.com)
- Facebook Likes
- Context (Recent searches)

The AOL Dataset has ~3 milllion queries which is reduced to ~1 million unique queries with corresponding frequencies.

The user has the option to login with Facebook, which gives us access to his likes.

Whenever the user types a few starting characters, top ~100 matching queries with the highest frequencies are extracted from the database.

These are further processed and there frequencies modified to get the final list :-

- If any of the trending topics or Facebook likes of the user matches the user's string it is inserted into list of suggestions with a high frequency
- Frequencies of those suggestions that match user's recent searches(last one hour or so) are incremented to bring them on top
- The final list is truncated to ~10 top suggestions that are displayed to the user.

This method can be further improved by having a rich dataset of queries with details like location, age, time of year about the query. We have used AOL's dataset released in 2006 that lacks the required level of freshness and details.

6. CONCLUSION AND FUTURE SCOPE

In this paper we have shown how we can utilize social network logins and user details to return more personalized and accurate predictions. With the easily available and straight forward API, integration of social logins has become very simple.

Using social network login makes way for a huge amount of personalization in query auto completion. Merged with context sensitive and time sensitive QAC, it can return results that are very close to what the user wishes to search. We may also be able to predict what the user will search next and give search suggestions for the user to know more on the topic she is researching.

REFERENCES

[1] J. J. Darragh, i. H. Witten, and m. L. James. The reactive Keyboard: a predictive typing aid. Computer, November 1990.

[2] K. Grabski and T. Scheffer. Sentence completion. In Proc. SIGIR, Sheffield, United Kingdom, 2004.

[3] A. Nandi and H. V. Jagadish. Effective phrase prediction. In Proc. VLDB, pages 219–230, Vienna, Austria, 2007.

[4] S. Bickel, P. Haider, and T. Scheffer. Learning to complete sentences. In Proc. ECML, volume 3720 of Lecture Notes in Computer Science, pages 497–504. Springer, 2005. ISBN 3-540-29243-8.

[5] S. Chaudhuri and R. Kaushik. Extending autocompletion to tolerate errors. In Proc. SIGMOD, pages 707–718, Providence, Rhode Island, USA, 2009.

[6] S. Ji, G. Li, C. Li, and J. Feng. Efficient interactive fuzzy keyword search. In Proc. WWW, pages 371–380, Madrid, Spain, 2009.

[7] Z. Bar-Yossef and N. Kraus. Context-sensitive query autocompletion. In Proc. WWW, pages 107–116, Hyderabad, India, 2011.

[8] M. Shokouhi and K. Radinsky. Time-sensitive query autocompletion. In Proc. SIGIR, pages 601–610, Portland, Oregon, USA, 2012. ISBN 978-1-4503-1472-5.

[9] A. Strizhevskaya, A. Baytin, I. Galinskaya, and P. Serdyukov. Actualization of query suggestions using query logs. In Proc. WWW, pages 611–612, Lyon, France, 2012. ISBN 978-1-4503-1230-1.

[10] I. Weber and C. Castillo. The demographics of web search. In Proc. SIGIR, pages 523–530, Geneva, Switzerland, 2010. ISBN 978-1-4503-0153-4.

[11] B. J. Jansen, A. H. Spink, C. Blakely, and S. Koshman. Defining a session on web search engines. Journal of the American Society for Information Science and Technology, 58(6):862–871, 2007.

[12] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. In Proc. SIGIR, pages 795–804, Beijing, China, 2011. ISBN 978-1-4503-0757-4.

[13] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In Proc. SIGKDD, pages 875–883, Las Vegas, NV, 2008. ISBN 978-1-60558-193-4.

[14] N. Matthijs and F. Radlinski. Personalizing web search using long term browsing history. In Proc. WSDM, pages 25–34, 2011. ISBN 978-1-4503-0493-1.

[15] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In Proc. InfoScale, New York, NY, USA, 2006. ACM. ISBN 1-59593-428-6.

[16] Y. Song and L.-w. He. Optimal rare query suggestion with implicit user feedback. In Proc. WWW, pages 901–910, Raleigh, NC, 2010. ISBN 978-1-60558-799-8