Context Based Query Auto-Completion

Veena K Mawarkar[#1], V. S. Malemath[#2]

***Abstract*— Auto-Completion is the first service the user uses to search and formulate the query. In the increased world of Web today, accessing information is one of the most frequent task. The users have to type each word or character sequentially and spend more time in completing the query. Often for a single query search engines returns thousands of Web pages, which are irrelevant to the user's intended query. Querying techniques are used by most of the search engines .Query Auto-completion service help the user formulate their queries as they type and save users time. It is difficult for query auto-completer to accurately predict what the user is typing. Query Auto-completion often miss predicts the user's query when their input prefix is too short (e.g. one or two characters). In this work we illustrate query auto-completion based on context, recent click through's, user's current location and time, keywords, and sessions and by considering these factors the suggestion quality will be considerably higher even for short prefixes. We used the standard MostPopular Completion, Nearest Completion and Hybrid Completion algorithms. We calculated MRR values of the three algoritms and compared our results. The Nearest Completion algorithm has 32.7% better predictions than Most Popular algorithm. And the average MRR for Hybrid Completion is 0.83 which is higher than both the algorithms in giving more relevant predictions for the user's queries with 1 or 2 of prefix length. We then considered the user's current location for auto suggestion and the predictions of correct suggestions are better with location.**

***Keywords*— query Auto-completion, Keyword search, WordNet.**

I. INTRODUCTION

Query Auto-Completion (QAC) is the process of computing the user's words or phrases for every keystroke and providing the users with the list of suggestions in the real time on the basis of user's past popular queries. This service is adopted by many search engines e.g Google, yahoo!, Bing or Ask search engines to help the user formulate their query as they type in the characters into the search box. In the recently launched google suggest service, frequent queries are suggested [13]. With the increased need of information retrieval in today's world the suggested completions are useful in many cases as the user may not know what words to use or how to describe his information need [9]. The purpose of this study is to predict the user's intended query at the first few input characters (e.g one or two characters) and at the highest rank in the list of suggested completions. For each keystroke of the user, efficient algorithm is needed to process a query within milliseconds at the backend.

The most common approach to QAC is to retrieve past queries with each prefix from a query log, and rank them by their past popularity which is called Most Popular completion algorithm [11]. In this algorithm, the user's previous query in the same session is y and that the current user input (query prefix) is x, thus the search engine suggests the completions of x that are most frequently searched for after y [11]. Although this gives a satisfactory result on an average, but it is not that optimistic, since the given prefix could be short and ambiguous, and the number of the predicted list may be extremely large. Existing methods mostly rank the completions by their popularity in the logs. Consequently, less popular queries are harder to predict. And it fails to take into consideration the factors such as time or user context which often influence the queries that are most likely to be typed.

The main objective of this study is to improve the search quality by predicting the user's query not only based on the past popularity but query similarity, the user's recent click throughs, current location and time, keywords and sessions.

The paper is organized with section II shows the related work, section III deals with the proposed work, section IV has the definitions of the three algorithms, section V discuss implementation, results and discussion are given in section VI and section VII deals with conclusion.

II. LITERATURE SURVEY

Fei Cai [5] proposed a hybrid QAC model by combining both time sensitivity and personalization. The author claims that time, trend and user-specific context when considered, the queries can be better predicted. First the top N queries are ranked by their past popularity and the popularity based on cyclic behavior. Then these top N completions are reranked by user-specific context that is by taking the user's preceding queries to get the list of correct suggestions. To get better suggestions Time series analysis is done and an optimal time window is assigned to find the query's popularity trend.

Li, Yanen, et al [7] proposed a Two-dimensional Click Model for Query Auto-completion, where he notices the real time user's behavior in PC and mobile phones. By observing every keystroke and the system response in a QAC session, a high-resolution QAC query log is collected. It is likely that sometimes user tends to skip the list of top suggestions even if it contains user's intended query, reasons for this may be fast typing speed or too deep to look into. With one or two characters the probability of click is very low, as the user tends to skip the list of suggestions at the beginning. The two user behaviors, namely the horizontal skipping bias and vertical position bias which are crucial for relevance prediction in QAC. This click model consists of a horizontal model that explains the skipping behavior, a vertical model that depicts the vertical examination behavior, and a relevance

ISSN: 2278 – 1323

model that measures the intrinsic relevance between the prefix and a suggested query.

Liang, Wu [6] most search engines adopt clustering based approach for suggesting queries. Clustering reduces the sparseness of data, produces noises, and ignores the sequential information of query refinements in search sessions. He proposed to solve this issue by filtering out irrelevant queries from the suggested list and predicted the refinement direction. Queries are grouped according to their clicked patterns, and suggestion candidates are selected from the clusters where the query belongs. Then, machine learning and data mining techniques were used to rank the suggestion candidates. The sequential information is captured by two major refinement behaviors, one which simplifies the original query and second the queries in which the user wants to be more specific.

Eero [14] has worked on auto-completion at semantic level. Generally, for the user's query the list of matching completions is the words in vocabulary and he proposed a novel approach to get the similar words for the user's input, and matching ontological concepts whose labels are not related to the input on the literal level. He proposed semantic auto-completion in multi-facet search where the contents are organized and retrieved using multiple hierarchical structures. The data is categorized into different facets and are indexed using the keywords. For example, if the user types in the word 'bank', this could be completed into categories 'river bank' and 'bank (financial)' and the result set includes an union of both geographical and organizational hits. The suggestions are in the form of hierarchy, which helps the user in getting the right information.

Lorand [12] modeled personalized auto-completion systems which used the demographic features of the specific-users like age, gender, income and location as the context will differ as there will be different queries. This study stresses on news search, as the users are often interested in logging into news website when doing search. The age and income of the users were divided into ranges and the location that appeared more number of times is only considered. The demographic data was available only from the logged in users and the users who were not registered on the websites their location was extracted from the IP address and no demographic data was available. Thus the queries are ranked preferably taking different lengths of the prefix of the query and taking the time (hour of the day) and information of the users into consideration.

Christian [8] proposed a dynamic query suggestion model called CONQUER, which gives efficient query suggestions for a given query prefix and based on the number of context observations like user location (country domains of the clicked URLs(Uniform Resource Locator)) and time of day.This model is based on the probability of sequential query patterns, context observations and it uses an extended FP-Growth algorithm for mining the frequent sequential query patterns. For suggesting more relevant queries hour of day as a timestamp and clicked URLs are extracted. The suggestions are ordered by their score. To demonstrate the time slider and a check-box was provided which allowed the user to change the context parameters and to set the weight of a context. The suggestions are dynamically changed as the slider is moved.

III. Proposed Work

The user has some context which reveals some information about their intent. The user's recent queries are used as a context. To determine if the context is relevant to the user's current query, we use the Nearest Completion algorithm that is based on the similarity assumption. We have used the Word Net database to get the similar words and listed the suggestions. Suppose for a given user input x and a context y, the algorithm suggests to the user the completions of x that are most similar to y. the suggestions that are most similar to the context are indeed more likely to be the user's intended query.

If the user has entered the query for the first time, context of that query is empty. Nearest completion works well when the context is non-empty, it fails to produce relevant suggestions since it has no information related to the current query. To overcome this problem we used the Hybrid Completion algorithm[11] which is the combination of both MostPopular Completion and NearestCompletion.

We have calculated the MRR of all the 3 algorithms and compared the results. We also tried to auto-complete based on the user's location by allowing the user to select his current location and compared the MostPopular Completion with and without location.

IV. Query Auto-Completion

A. Query Auto-Completion Definition

Query Auto-Completion (QAC) is the process of computing the user's words or phrases for every keystroke and thus providing the users with the list of suggestions in the real time. The algorithm accepts an input x which is the prefix of a complete query q, that the user tends to enter into the search box and returns a list of k completions, which are the list of possible completions for queries, from which the user can select.

B. MostPopular Completion

Queries are suggested from the previous query popularity. The MostPopular QAC algorithm [11] returns the top k completions for an input x that have been most popular among the users in the past. The preferences for the suggestion list are given by their number of frequent occurances.For example- if just before entering the characters ba the user has searched for Bank related queries it is more likely that the user

ISSN: 2278 – 1323

is looking for Bank interest rates, bank of America than for basketball or barracuda. This algorithm makes use of TRIE data structure discussed in (section V).

C. Nearest Completion

The Nearest Completion algorithm [11] uses the recent queries as context of the user input x. When the context is related to the query the user is typing, the algorithm has better chances of producing a hit. This algorithm is based on Lucene indexer where each query is indexed by its set of eligible prefixes, so that the user can retrieve all the completions of a given prefix quickly.

D. Hybrid Completion

For an input x and a context C, Hybrid Completion produces to ranked list of suggestions of x: L_{NC} consists of the top ℓ completions returned by NearestCompletion and L_{MPC} consists of the top ℓ completions returned by MostPopular Completion [11]. Finally we produced L_{HC} ranked list of suggestions combining the two lists.

V. IMPLEMENTATION

A query database is built which contains a large collection of queries. This database is built from a query log by extracting the most frequently searched queries. Here we are making use of AOL query log [3] which is a large collection of queries. Each QAC algorithm has its own constraint that depicts whether a query q is an eligible completion of input x. Traditional QAC algorithms require that q is a proper string completion of x(i.e. x is prefix of q).For instance, president is a proper completion of the input pres.

At run-time, the QAC algorithm accepts an input x, and lists the top k eligible completions for x. Completions are ranked by a quality score, which depicts how likely each completion is to be a hit. Since the algorithm needs to provide the user with the suggested completions as the user is typing the query, it has to be ultra-efficient. To achieve this high performance, the algorithm needs a data structure that supports fast lookups into the query database using prefix keys.

There are different algorithms and data structures for indexing and searching strings within a text, the TRIE data structure is one good example. For example from a large set of dictionary, if we have to look for a word or single string from that dictionary, trie datastructure is the one which can be used for fast lookups from a huge set of words [4].Although there are other data structures that can be used like set<String> and Hash tables, but it can search only the words that exactly matches with the word we are looking for, whereas the tries allows to search words that have prefix in common, single character different, a missing character etc. With tries web browser can auto-complete user's text or show many possibilities of the text that the user could be writing in very fast way.

A. MostPopular Completion

In MostPopular Completion algorithm we used TRIEs to index AOL log. The trie is a tree where each node represents a single word or a prefix. The root indicates an empty string (""), the nodes of the root indicates prefixes of size 1, the nodes that are 2 edges of distance from the root indicates prefixes of size 2, the nodes that are 3 edges of distance from the root indicates prefixes of size 3 and so on. In other words, nodes that are k edges of distance of the root have a related prefix of size k. Let n1 and n2 be two nodes of the trie, and assume that n1 is a direct father of n2, and then n1 must have a related prefix of n2. In this format the AOL log data is stored.We first created the user's session, when the user logs in and begins typing into the search box; the queries matching with the prefix from the AOL log are extracted and returned to the user. If the suggested queries match with user intent the completion is said to be hit and the selected query is stored in the database. The algorithm then returns the top completions based on the previous clicks and which he has searched for most frequently.

For Location based search the user's current location was detected and the relevant suggestions are provided based on that user's location.

Algorithm: MostPopular Completion
Step1: User Logs in
Step 2: Detect Location
Step 2: Enter characters into the search box
Step 3: If no location, check for the history into the database and get the queries, if no history is present goto step 5.
Step 4: if location is detected get the queries related to the location and goto step 7.
Step 5: Read AOL query log
Step 6: Extract the queries and generate the rank
Step 7: Display results by Auto-completing the queries.

B. Nearest Completion

Searching is faster if the queries are indexed. Lucene is a powerful Java search library that makes it easy to add search functionality to any application [10]. Lucene allows indexing the document, we can perform queries on this index and it then returns a list of ranked results where the ranking is based on the relevance of the query or sorted by an arbitrary field such as a document's last modified date. Lucene is faster in searching a text as it searches for the index of the text rather than finding the text directly. This is equivalent to retrieving pages in a book related to a keyword by searching the index at the back of a book, as opposed to searching the words in each page of the book [1].

Indexing involves adding Documents to an IndexWriter, and searching involves retrieving Documents from an index via an IndexSearcher. Searching requires an index to have already been built. It involves creating a Query (usually via a QueryParser) and handing this Query to an IndexSearcher, which returns a list of hits. A TopScoreDocCollector class of Lucene is instantiated to collect the top 10 scoring hits. In the NearestCompletion algorithm we indexed the AOL log and extracted the queries from the indexed document. The success of this algorithm is based on similarity assumption. We

ISSN: 2278 – 1323

extracted the recent queries as context from the database considering the current data and time of the query from the user's session. And then found the similar wordforms of these queries from a WordNet database. WordNet interlinks not just word forms—strings of letters—but specific senses of words. As a result, words that are closely related to one another are semantically disambiguated. WordNet is a huge lexical database of English. Nouns, verbs, adjectives and adverbs that are grouped into sets of cognitive synonyms (synsets), each giving a unique concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing [2].

Algorithm: NearestCompletion
Step1: User Logs in
Step 2: Enter characters into the search box
Step 3: Check for the recent queries into the database and get the queries, if no recent clicks are present goto step 5.
Step 4: if recent clicks are present get the similar wordForms related to the recent query matching with the current query.
Step 5: Read AOL query log
Step 6: Extract the queries from step 3, 4 and 5 generate the rank
Step 7: Display results by Auto-completing the queries.

C. Hybrid Completion

We collected the top k lists of both the MostPopular Completion and Nearest Completion. And aggregated the results of the two and produced a final list of ranked completions.

Algorithm: Hybrid Completion
Step1: User Logs in
Step 2: Enter characters into the search box
Step 3: Check for the recent queries and history (previous clicks) into the database and get the queries
Step 4: if recent clicks are present get the similar queries related to the recent query, matching with the current query and the queries occurring frequent times in the history
Step 5: Aggregate the queries and generate the rank
Step 6: Display results by Auto-completing the queries.

VI. Results And Discussion

Our study is to compare the best configuration of the three algorithms that is MostPopular Completion, NearestCompletion and Hybrid Completion algorithms and then estimate the average MRR of these algorithms based on the query log. We made use of AOL query log in our experiments,which is publicly available and contains huge number of queries to guarantee statistical significance(other public query logs are either access-restricted or are small).

To perform the empirical study we implemented the standard MostPopular, Nearest Completion and Hybrid Completion algorithms.The query database used by these three algorithms was built from the queries that are present in the AOL log. We kept track of each user's session (there was no any session boundary or time interval assigned to the session). All the search results we got were searched in earlier days by the user. For the MostPopular Completion we showed the results which were searched for most frequently by keeping track of user's previous clicks. And the Nearest Completions shows the results by taking context into consideration based on the recent clicks. The success of this algorithm lies in the context. If the context is relevant to the current query there is more possibility of producing a hit. WordNet database is used to understand the context of the recent queries. It gives the similar word forms of the query. Figures below shows the snapshots of these two algorithms

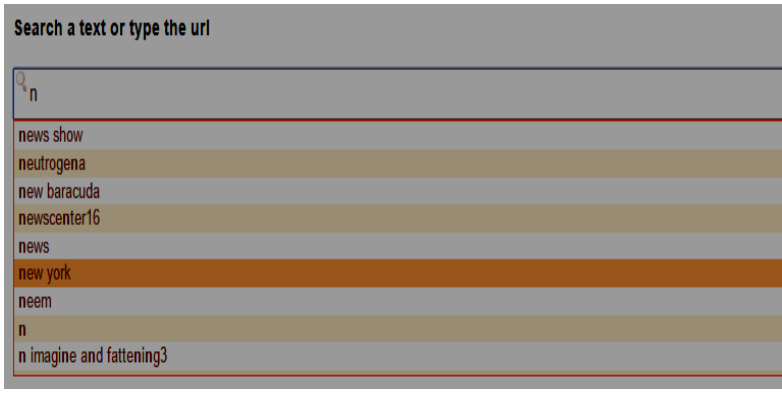

Fig. 6.1: MostPopular Completion

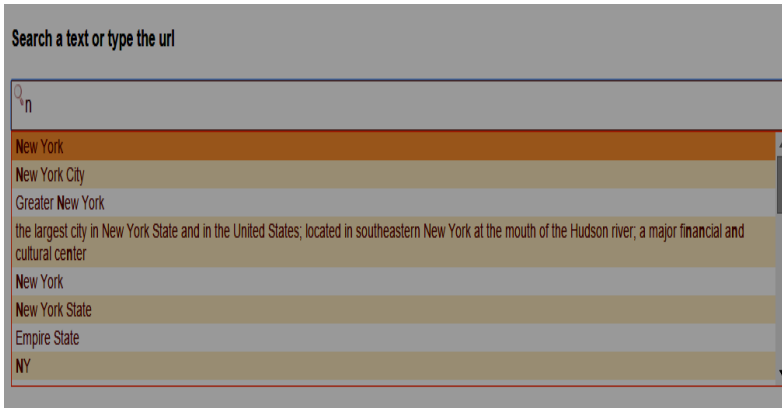

Fig. 6.2**:** Nearest Completion

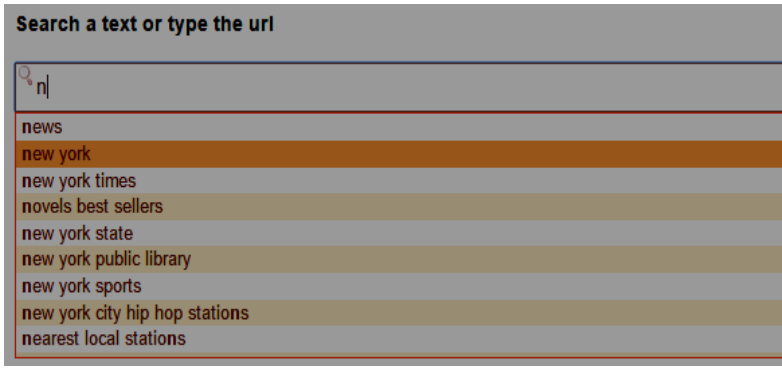

Fig. 6.3**:** Hybrid Completion

ISSN: 2278 – 1323

We then aggregated the results of these two algorithms and produced the Hybrid Completion algorithm.

Suppose the user has searched for NY recently. As shown in Fig. 6.1, when user wants to search for the New York city, News has been most popular search among the user in the past, so the queries related to news is ranked at 1st position and New York is ranked in the 6th position, thus the MostPopular Completion fails to understand the user's intent. Where as in Fig. 6.2, Nearest Completion gives the list of suggestion by seeing the recent query's context of the user .In other words, it takes the recent query as context and completes the user's relevant queries based on the context. The Hybrid Completion in Fig. 6.3 gives the combined list of results of both the algorithms. We aggregated the list by considering the most frequent and recent searches of the user. In Fig. 6.3 New York is been ranked in the 2nd position also this algorithm tries to give more similar suggestions based on the query frequency and context.

We collected the sample queries to test and compare our results. The following table shows the comparison.

TABLE I: The Top 5 Completions for 1 or 2 Prefixes Provided by The 3 Algorithms on (Query, Context) Pairs Taken from the AOL Log.

Context	Query	MostPopular Completion	Nearest Completion	Hybrid Completion
NY	New York	News show, neutrogena, new barracuda, news center 16, news	**New York** New York city, Greater new york city, the largest city in US, Empire state	News, **New York**, ny, news, intelligence, tidings
President bush	George bush	Gearsource, Gmail.com, ga tech athletics, Gambling in branson, Gemini	**Bush,** George W Bush, George Walker Bush, President George Bush,Dubyuh	**George Bush** Gemini **,** geological study, ge profile range, Gambling in branson
Publication	pubs	**Pubs**, publication, pub restaurant, pa auto dealer license, pa childcare	Publication, a copy of a printed work, issue, publishing, bring out.	**Pubs**, pub restaurant, publication, Public house, pub.

As shown in Table 1, in the first example the Nearest Completion is likely to produce hits as the previous context is relevant to the current query and hence it provides the user with most relevant suggestions,while the MostPopular Completion fails to produce hits in this context.In the second example all the 3 algorithms are likely to produce hits with different ranking scores. In the third example the nearest Completion fails as the context is irrelevant to the current query where as the MostPopular and Hybrid Completion are likely to produce hits because it gives the suggestions that have been searched for most of the time by the user and also taking context into consideration of recent searches thus help user formulate their query based on previous frequent searches.

Evaluation Metric: For a particular query q and a context C, an algorithm is said to produce hit at position ℓ,if the context C is received and the prefix of q, the algorithm returns q as the ℓth completion for this prefix. In this case we write hits as hitrank(A,q,C)= ℓ(if A has no hit at all, then hitrank(A,q,C)= ∞).Mean Reciprocal Rank(MRR) is a standard measure for calculating the average retrieval. The reciprocal rank of an algorithm A on a particular (query, context) pair (q,C) is 1/ hitrank(A, q,C) (note that the reciprocal rank is 0 when the algorithm has no hit). MRR is the expected reciprocal rank of the algorithm on a random (query, context) pair. To estimate MRR, we take a random sample S of (query,context) pairs, and compute the mean reciprocal rank of the algorithm over these pairs[11]:

$$MRR(A) = \frac{1}{|S|} \sum_{(q,C)\in S} \frac{1}{hitrank(A,q,C)}$$

Based on the lists of completions shown in Table 1 we calculated the average MRR (Mean Reciprocal Rank) of the three algorithms. The average MRR of MostPopular completion is 0.33 and the average MRR of the Nearest Completion is 0.66. The NearestCompletion algorithm is 32.7% better than that of MostPopular in producing relevant results. And the average MRR of Hybrid completion is 0.83.

TABLE II: The Top 5 Completions for 1or 2 Prefixes Provided by the MostPopular Completion With and Without Location.

Query	Location	MostPopular With Location	MostPopular Without location
Pubs	Goa	**pubs in goa**, pub & beaches, portuguese architecture, panjim, places to see in goa	**Pubs**, publication, pub restaurant, pa auto dealer license, pa childcare
George bush	USA	**George bush**, GeorgeClooney production.co, ge profile range ,gearsource	Gearsource, Gmail.com, ga tech athletics, Gambling in branson, Gemini
New York	USA	New York times , **New York**, New city hiphop stations, Novels best sellers, New York public library,	News show, neutrogena, new barracuda, news center 16, news

ISSN: 2278 – 1323

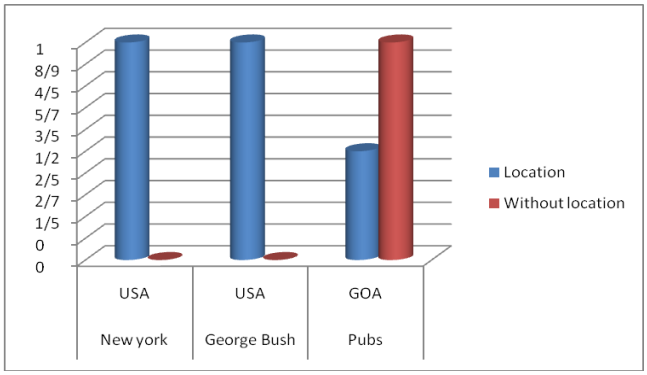

Fig. 6.4: The ranking of suggested completion for MostPopular Completion with and without location

The graph shown in Fig. 6.4 x-axis gives the fraction of ranked queries and y-axis shows the query and location selected by the user. We compared the fractions of different queries for particular location and without location. And we observed that predictions are better when searched according to the location than without location

From the table shown in Table 2 and the graph shown in Fig. 6.4, we observed that the MostPopular Completion works well when the current location is considered. It is more likely that the user is interested in searching the information like popular places, history or current events happening in that particular location

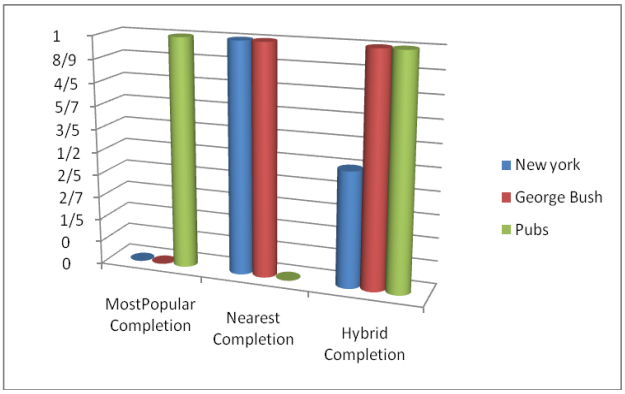

Fig. 6.5: The Reciprocal ranking of the 3 algorithms on the number of (query, context) pairs.

In the graph Fig. 6.5 x-axis shows the fractions of (query, context) pairs and on y- axis the three algorithms on which the MRR values are calculated and compared. Results are for all pairs, for pairs in which the context is non-empty and for pairs in which the context is empty. The MRR values of the 3 algorithms for some queries are higher than that of the other algorithm.

In the graph Fig. 6.6 on x-axis we show the calculated Average MRR values for three algorithms and y-axis shows the 3 algorithms. We compared the calculated MRR values of 3 different algorithms and observed that the percentage of correct predictions is higher when hybrid Completion is used.

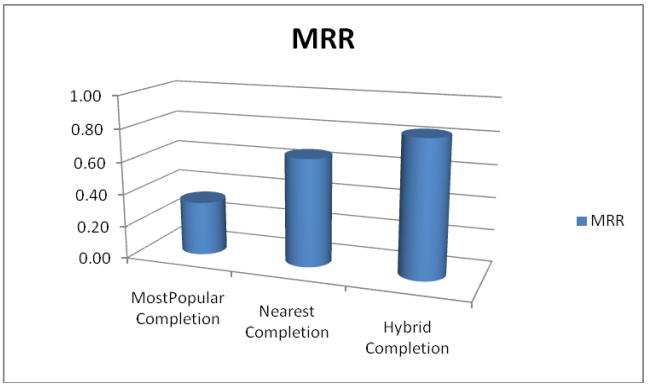

Fig. 6.6: Average MRR of the 3 algorithms on the number of (query, context) pairs

VII. Conclusion

Query Auto completion is an important feature for any user to lessen the typing. By considering the factors such as user's recent click's, keywords, session, current location and time we improve the searching quality for the user and save the time.By providing predictions at only 1 or 2 characters. TRIE data structure and Lucene provides fast look ups into the database by invoking the queries in split seconds as the user is typing. In this work we have shown that the Nearest Completion algorithm has 32.7% better predictions than Most Popular algorithm. And the average MRR for Hybrid Completion is 0.83 which is higher than both the algorithms in giving more relevant predictions for the user's queries with 1 or 2 of prefix length. We then considered the user's current location for auto suggestion. It was observed that in the MostPopular Completion, the predictions of correct suggestions are better with location as it is highly possible that the user intends to look up for the information like popular places, history or current events happening in that particular location.

Future Scope

As we have considered the previous click through's of the user's query we will try to understand the subject, predicate and object of those query and find the relationship with the current user input and produce the list of suggestions.

Acknowledgement

The authors are immensely thankful to the valuable guidance provided by the department of Computer Science and Master of Computer Application, KLE Dr.M.S.Sheshgiri College of Engineering and Technology, Belagavi, Karnataka, India.

References

[1] http://www.Lucenetutorial.com/basic-concepts.html 09:01pm 5/17/2015
[2] Princeton University "About WordNet." WordNet. Princeton University. 2010. <http://wordnet.princeton.edu> 09:25 pm 5/17/2015
[3] Pass, Greg, Abdur Chowdhury, and Cayley Torgeson. "A picture of search."*InfoScale*. Vol. 152. 2006.
[4] https://www.topcoder.com/community/data-science/data-science-tutorials/using-tries 8/6/2015 9:20 pm
[5] Cai, Fei, Shangsong Liang, and Maarten de Rijke. "Time-sensitive personalized query auto-completion." *Proceedings of the 23rd ACM*

ISSN: 2278 – 1323

International Conference on Conference on Information and Knowledge Management. ACM, 2014.

[6] Wu, Liang, et al. "Improving query suggestion through noise filtering and query length prediction." *Proceedings of the companion publication of the 23rd international conference on World wide web companion*. International World Wide Web Conferences Steering Committee, 2014.

[7] Li, Yanen, et al. "A two-dimensional click model for query auto-completion."*Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014.

[8] Sengstock, Christian, and Michael Gertz. "CONQUER: a system for efficient context-aware query suggestions." *Proceedings of the 20th international conference companion on World wide web*. ACM, 2011.

[9] Kastrinakis, Dimitrios, and Yannis Tzitzikas. "*Advancing search query autocompletion services with more and better suggestions"*. Springer Berlin Heidelberg, 2010.

[10] McCandless, Michael, Erik Hatcher, and Otis Gospodnetic. *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co., 2010.

[11] Kraus, Naama, and Ziv Bar-Yossef. "Context-Sensitive Query Auto-completion." *dated Mar* (2010).

[12] Dali, Lorand, Blaž Fortuna, and Jan Rupnik. "Personalized Query Auto-Completion for News Search."Jožef Stefan International Postgraduate School,Artificial Intelligence Laboratory - Jožef Stefan Institute.2009

[13] Bast, Holger, Christian W. Mortensen, and Ingmar Weber. "Output-sensitive autocompletion search." *String Processing and Information Retrieval*. Springer Berlin Heidelberg, 2006.

[14] Hyvönen, Eero, and Eetu Mäkelä. "Semantic autocompletion." *The Semantic Web–ASWC 2006*. Springer Berlin Heidelberg, 2006. 739-751.

Veena K Mawarkar Mtech in Computer Science & Engg. *KLE DR M S Sheshgiri College of Engg. & Tech, Belgaum*

V.S Malemath Prof Computer Science & Engg. *KLE DR M S Sheshgiri College of Engg. & Tech, Belgaum.* ,Image Processing, Pattern Recognition, Artificial Intelligence and Data Mining

ISSN: 2278 – 1323