An enterprise search paradigm based on extended query auto-completion. Do we still need search and navigation?

David Hawking
Funnelback Pty Ltd.
Canberra ACT 2602, Australia, and
RSCS, the Australian National University
david.hawking@acm.org

Kathy Griffiths
Centre for Mental Health Research
the Australian National University
Canberra Australia 0200
kathy.griffiths@anu.edu.au

ABSTRACT

Enterprise query auto-completion (QAC) can allow website or intranet visitors to satisfy a need more efficiently than traditional searching and browsing. The limited scope of an enterprise makes it possible to satisfy a high proportion of information needs through completion. Further, the availability of structured sources of completions such as product catalogues compensates for sparsity of log data. Extended forms (X-QAC) can give access to information that is inaccessible via a conventional crawled index.

We show that it can be guaranteed that for every suggestion there is a prefix which causes it to appear in the top k suggestions. Using university query logs and structured lists, we quantify the significant keystroke savings attributable to this guarantee (worst case). Such savings may be of particular value for mobile devices. A user experiment showed that a staff lookup task took an average of 61% longer with a conventional search interface than with an X-QAC system.

Using wine catalogue data we demonstrate a further extension which allows a user to home in on desired items in faceted-navigation style. We also note that advertisements can be triggered from QAC.

Given the advantages and power of X-QAC systems, we envisage that websites and intranets of the [near] future will provide less navigation and rely less on conventional search.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval—*Information Search and Retrieval*; H.3.4 [**Information Systems**]: Information Storage and Retrieval—*Systems and Software*

Keywords

Search paradigms; query sugggestion; query auto-completion.

1. INTRODUCTION

It has become routine for Web and other applications to assist users by suggesting likely completions of partly typed queries. Users have come to expect this behaviour when typing queries into a Web search engine, when searching for a product in an online store, when getting directions in a GPS navigation application, and in many other settings. Query completion can save time, reduce misspellings, and guide users toward queries they might not otherwise have thought of. It is considered to be of even greater value when a person is interacting with a system via a device with a restricted keyboard, such as a smartphone.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
Australasian Document Computing Symposium ADCS '13, December 05 - 06 2013, Brisbane, QLD, Australia Copyright is held by the owner/author(s). Publication rights licensed to ACM. ACM 978-1-4503-2524-0/13/1 ... $15.00. http://dx.doi.org/10.1145/2537734.2537743

Figure 1: A simple query completion system provided by a Web search engine. We assume that the suggestions are those completions of 'ba' which were estimated to be most probable in the context – search by the first author, not logged in, from a machine in Australia, in July 2013.

The phrase "query auto-completion", abbreviated by Shokouhi and Radinsky[10] to "QAC", was originally used to describe the replacement of a partially typed query by a completed version which is then submitted for processing. Such a QAC system is illustrated in Figure 1.

However, the "auto-completion" concept can be extended so that the partial query is matched against the *trigger* field of a tuple:

$$(trigger, display-item, category, action-string, action-type)$$

In this extended form (which we will refer to here as X-QAC), the separation of trigger from both display-item and action-string means that the displayed 'completion' may not even include the characters typed so far, that completions may be categorized, and that selecting a completion may result in an action other than submission of a completed query.

Action types which are supported by some X-QAC systems are listed below. We introduced the E action type to facilitate an efficient form of faceted navigation.

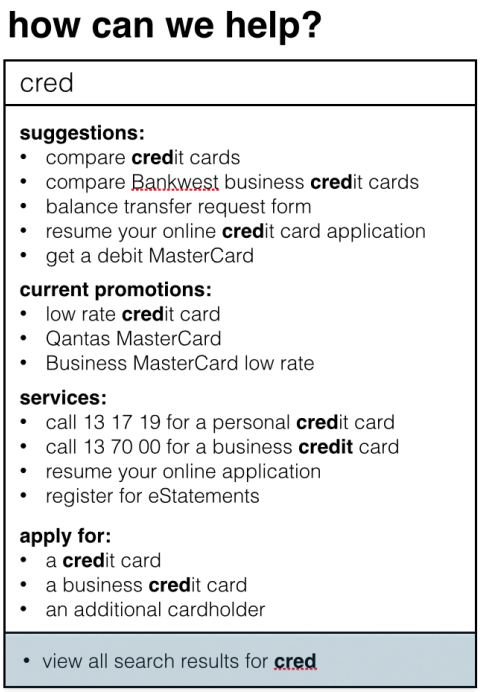

Figure 2: Suggestions from the extended query auto-completion system at `www.bankwest.com.au` on 26 Aug 2013.

Q	Run the action-string as a query
U	Navigate directly to the action-string, a URL
C	Execute action-string as a JavaScript callback
E	Extend the query by appending action-string to it, then generate a new set of completions

Figure 2 illustrates some of the capabilities of extended QAC. First, the suggestions are categorized. If an item in the first category (*suggestions*) is selected, the display string is submitted as a query, while suggestions in other categories will activate navigation to a particular page. Second, none of the display strings have a prefix of 'cred' and some do not even contain 'cred' as a substring. Third, some suggestions contain information (such as telephone numbers) which may obviate the need to vist a web page. Finally, the displayed suggestions are not derived from query logs. Bankwest reports[1] that the majority of visits to web pages on their site now arise from the X-QAC mechanism, rather than from navigation or conventional search. This observation raises the question at the heart of the present paper:

> *To what extent, within organisational websites and intranets, could an extended query completion paradigm replace the well established paradigms of navigation and search?*

1.1 Contributions

We start with a short tutorial on both simple and extended versions of query auto-completion and show the potential value of more widespread and more advanced use of X-QAC within enterprise contexts. We confirm full coverage of such a system by presenting a guarantee that every suggestion within an extended QAC system will appear in the top k suggestions for at least one prefix. We report substantial potential savings in keystrokes associated with this guarantee. In addition we quantify time savings due to X-QAC in a user experiment involving a university staff look-up task. We report anecdotal evidence from a bank of very high use of their X-QAC system by visitors to their site. Finally, we propose a new X-QAC action-type and show that it can be used to support a very efficient form of faceted navigation.

[1]Personal communication.

2. QUERY AUTO-COMPLETION

This section provides detailed information on QAC systems, outlines the terminology used and discusses previous work.

2.1 Terminology

We will use the term QAC (query auto-completion) to refer to a system or paradigm in which a partial query typed by a user leads to the "instantaneous" presentation of a set of suggestions, for possible selection by the user, as in Figures 1 and 2. In the literature this has also been called "pre-submission query suggestion" or "dynamic query suggestion".

A simple QAC system is one in which all the suggestions are completions of what the user has so far typed, and in which selection of a suggestion will result in submission of the completed query to a search system. An extended QAC (X-QAC) system generalises the suggestions in any of the ways outlined in the Introduction. We will use X-QAC when we need to make it clear that we are discussing the extended version.

We distinguish QAC both from "post-submission query suggestion" and from "instant search". The former suggests query refinements to the user after a full query has been submitted (possibly through QAC). The latter is like an "I'm feeling lucky" version of simple QAC in which the most probable query completion is submitted behind the scenes whenever the system's confidence in a suggestion exceeds a threshold. The completion which generated the instant results is usually shown greyed out, as the user is expected to continue typing unless the instant results are seen to satisfy his or her requirements.

2.2 Antecedents

QAC has its earliest antecedents in the command-line completion features in operating systems from the 1960s, such as the Berkeley Timesharing System and Tenex.[2] Another early technology related to QAC is the predictive typing feature of many smartphones and word processing systems. Witten et al [12] outline the benefits of such technology to people with a disability. A similar problem addressed by Grabski and Scheffer [6] is that of completing sentences in administrative letter writing, e.g. "Please let ... *me know if you have any further questions.*"

Baeza-Yates et al [1] credit Kevin Gibbs with being the first to develop a QAC system for web search (Google Suggest, in 2004 on Google Labs) though they state that Yahoo! Assist was the first to be deployed in production, in 2007.

2.3 Simple query auto-completion

A simple QAC system uses frequencies in a query log to estimate query submission probabilities and uses an index structure, such as an ISAM (Indexed Sequential Access Method) index, trie or inverted file to determine the k most probable queries starting with the prefix already typed. The probability of a suggestion may be estimated purely based on submission frequencies or using a function which combines frequencies or probability estimates with

[2]`http://en.wikipedia.org/wiki/Command-line_completion`

other features such as the difference in length between the prefix and the candidate suggestions.

A prefix-based QAC system may be configured to only activate when the user's typing satisfies a condition, such as exceeding a minimum length or provoking suggestions with a threshold degree of confidence. The prefix is usually defined as any sequence of characters, but could potentially be word based. The system described by Bast and Weber [3] seems to combine query completion with instant search. It suggests completions and search results for a partially typed word based on the result set for the full words already entered. For example, if the user has typed 'conference sig', word completions for 'sig' such as 'sigir' and 'sigmod' are suggested taking into account the search results for 'conference'. At the time of writing[3] `google.com.au` was observed to sometimes operate in a last-word completion mode as well as providing full-query completions.

Baeza-Yates et al [1] note that when deployed in 2007, Yahoo! Assist departed from the prefix completion model by offering some mid-string completions as well.

2.4 Query auto-completion within an enterprise

Bhatia et al [4] note that methods which rely on a large accumulated query log are unsuitable for use within enterprise domains. They present a system which harvests query completion suggestions from among the terms and phrases of the corpus.

Ji et al [7] report a QAC system in use on the staff directory at the University of California at Irvine. They propose efficient fuzzy matching based on trie-accessed inverted lists.

2.5 Extended query auto-completion (X-QAC)

Here we provide more detail on how simple query auto-completion can be extended to provide greater power and flexibility within an organization.

2.5.1 *Structured sources for X-QAC*

Many organizations hold tables of data which can be very effectively searched using query auto-completion. For example, on an e-commerce site the product catalogue contains just about all the information on the site and provides ideal raw material for a QAC system. It is no surprise that Amazon[4] and eBay[5] make extensive use of this technique.

Universities typically maintain a staff directory, a course database, a research outputs database, an events calendar, an online timetable, and a list of departments and other organisational units. Searches of these repositories or combinations of them can potentially benefit from X-QAC.

Structured sources often include information which can add value when displayed as part of the suggestion. When products are suggested, images, prices and ratings can be shown; Staff directory suggestions can show photos and contact details; while university course suggestions may present entry cutoffs, fees and expected graduation salaries.

2.5.2 *Suggestions which don't match the query prefix*

There are circumstances in which it is helpful to return suggestions that are not restricted to words matching the query prefix. It would be desirable for example to suggest "Angela Merkel" regardless of whether the searcher starts typing "Ang:." or "Mer:.". This can be achieved in the tuple model by allowing multiple triggers for a single display item and action. It is clear that triggering on every substring of a suggestion will result in misleading or low value suggestions. For example, "therapist" might be suggested for "era", "rap", and "rapist".

An ideal system needs a model of the probability that a person whose intention is best represented by a particular candidate suggestion S, will start by typing a prefix p. Intuitively, a person seeking a therapist is very unlikely to start by typing a letter such as 'i'. A rule of thumb derived from experience suggests that it useful to trigger on suggestion suffixes which begin with a whole word, unless that word is a stopword. E.g. "Bachelor of Applied Science and Engineering" might be triggered by any of:

```
bachelor of applied science and engineering
applied science and engineering
science and engineering
engineering
```

As noted earlier, the staff lookup system described by Ji et al [7] supports fuzzy name matching. Thus "Mustapha Farrakhan" for "Mustaf…". Another valuable feature in staff name matching is to expand the trigger set to include diminutive forms of first names, such as the English hypocoristics listed at `http://en.wiktionary.org/wiki/Appendix:English_given_names`. More generally, an organization may choose triggers from a locally applicable list of synonyms. For example, the Victoria Police might trigger "Permit to acquire a long arm" from "gun licence", or just "licence".

2.5.3 *Categorising suggestions*

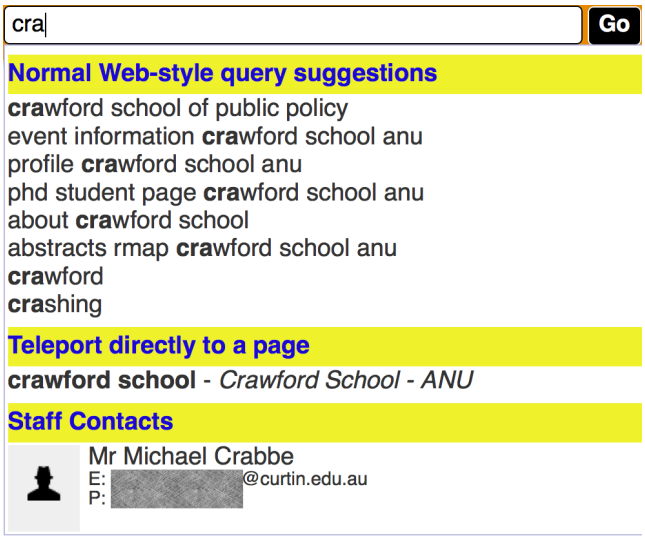

Figure 3: The multi-source completion mechanism deployed at a synthesized university prevents many categories of completions. In this example: selection of an item in the first category submits the item as a query; selection of the item in the second navigates directly to the URL which past users have clicked most frequently for that query; and the item in the third category gives information such as email address, phone number etc for a member of staff.

In many settings, suggestions can be derived from multiple sources. In a university they may come from the types of repository listed earlier, as well as query logs, and from the titles and anchortext of web pages. In a library they can be derived from lists of books, lists

[3] In August 2013, from a machine in Australia

[4] `amazon.com`

[5] `ebay.com`

of DVDs and lists of electronic resources. It is common to associate a category label with each display item and to group display items from the same category under the applicable label when presenting suggestions. See Figure 3 for an example.

2.5.4 *Contextualising suggestions*

Ideally, estimates of the likelihood that a suggestion will be useful should be made in the searcher's context. In the case of a mapping application the value of suggestions for "6 Main St" can be dramatically improved by suggesting the matching address closest to where the search is conducted. In general, the ranking of suggestions may take into account location, language, and any of the attributes used in personalising search results.

Various types of contextualization of simple QAC have been investigated. Bar-Yossef and Kraus [2] prioritize QAC suggestions for web search using knowledge of the searcher's recent search history. Shokouhi [9] describe a machine learning system for personalizing and "groupizing" a simple completion system. Shokouhi and Radinsky [10] and Strizhevskaya et al [11] take account of the time-dependent pattern of query submissions in order to predict query submission frequencies at the time of (simple) QAC suggestion, rather than using overall submission frequencies. For example, the probability of submission of "mistletoe" may be high at Christmas and low for the rest of the year.

Groupized and time-dependent contextualization may be applied to extended QAC systems in enterprises. In simple QAC systems it is important to ensure that suggested queries actually generate results in the scope in which the search is conducted. When scoped searches are offered it is therefore necessary to provide scoped versions of the QAC indexes. Filtering suggestions for hundreds of different scopes in the context of a large collection poses an efficiency challenge.

2.5.5 *Suggestion weights for structured sources*

In studies of QAC systems for Web search engines, suggestions have been weighted by actual or predicted frequency of submissions of the suggestion as a query. The situation is more challenging in enterprise search, because of the use of structured sources.

Within a single source, assigning weights may sometimes be straightforward. For example, in an e-commerce context it may be logical to weight suggestions by popularity, inventory holdings, profit margins or a function of those things. In other cases, there may be no obvious way to assign weights. A particular challenge arises when ranking suggestions from different sources, each with a different weighting scheme.

2.5.6 *Multi-lingual QAC*

In general, QAC systems must deal with data and queries in multiple languages. See Figure 4. Particular issues arise in the context of particular languages:

1. In languages, like French, which include accented letters, it is desirable that accented completions should be suggested for non-accented prefixes. For example, the prefix 'cite' might result in a suggestion of 'cité des enfants'.

2. In languages, like German, it should be possible to suggest accented completions from prefixes which include a digraph equivalent. For example, the prefix 'muen' might result in a suggestion of 'München', and 'beis' might suggest 'beißen'.

3. Text in Chinese poses additional problems, including the possibility of suggesting simplified text from traditional input or vice versa. Chinese is usually entered by typing Roman characters (e.g. PinYin) into an input system which pops up alternative Chinese character equivalents of what has been typed. A QAC system may make undesirable suggestions based on the Roman typing and the QAC pop-ups may conflict with the Chinese input pop-ups. Similar scenarios apply for other languages.

Figure 4: In general, QAC systems must deal with data and queries in multiple languages.

2.5.7 *Faceting via X-QAC*

An efficient form of faceted navigation [13] can be implemented using X-QAC. Figure 5 shows how a query extension action 'E' allows the user to home in on the wines he or she wants to buy, by extending the query with an additional constraint each time a selection is made and showing more suggestions based on the newly augmented query. In traditional faceting every item selection results in a complete search interaction and page reload. This is slow and an initial search is needed to activate the faceting system. In contrast, the QAC version can be instantaneous and a single character typed can be sufficient to expose a wide set of useful possibilities.

Interacting with a highly responsive QAC system can be very different to searching and faceting. Optimally it is like playing a computer game in which keystrokes and clicks take you closer to the treasure and a 'backspace' key can be used to back out of dead ends. When operating in computer-game mode users are highly attentive to changes on the screen which result from their actions and respond quickly with further actions.

2.5.8 *QAC driven advertising*

A QAC system can easily be used to trigger advertisements or best bets, taking into account that the display-item field is separate from the action and may include images and arbitrary formatting. A university might use this method to promote public lectures and events while an e-commerce site might promote products.

In all the screen shots shown here, the suggestion items are shown in a pane in front of the page content. It would be feasible to use AJAX[6] to alter the content of the page behind, based on the most likely completion.

2.6 User interaction with QAC interfaces

Relatively little data is available on user behaviour when interacting with QAC systems. Strizhevskaya et al [11] report a strong

[6] `http://en.wikipedia.org/wiki/Ajax_(programming)`

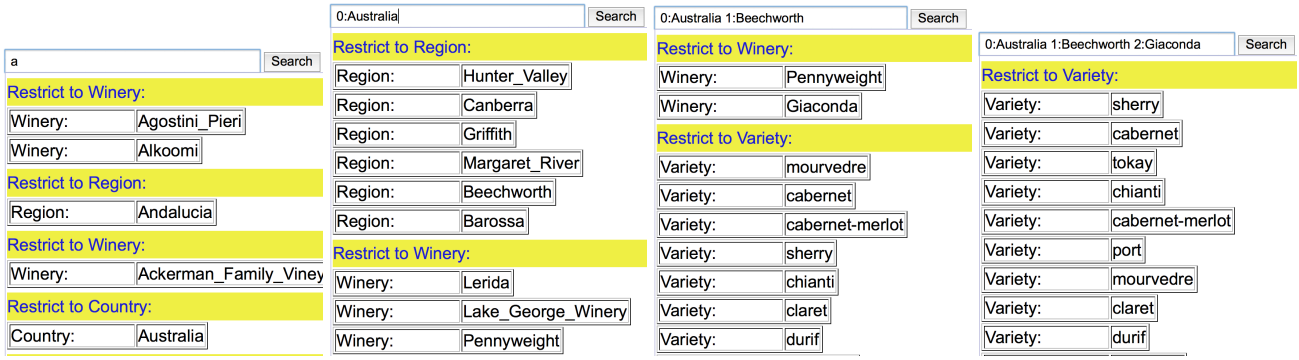

Figure 5: This shows the sequence of interactions between an imagined user and a simulated wine shop. Starting from (a) the user thinks they would like an Australian wine and types 'a'. The system suggests countries, wineries, and regions starting with 'a'. In (b) the user has clicked on Australia and the facet constraint 0:Australia is inserted in the query box, immediately triggering suggestions of regions and wineries within Australia. In (c) and (d) the user continues to narrow the selection. With one keystroke and three clicks this user has been able to see and select among all the regions within Australia, and all the wineries within the Beechworth region. He or she can now proceed to select a variety or just click the search button.

bias toward positions early in the list of suggestions, "a suggest click was in 51.15% cases on the first positions, in 20.8% on the second position, in 10.3% on the third." They also state that, "... an average user does not read all 10 suggestions and continues typing even if an intended query has already appeared in the tail of the posting [i.e. list of suggestions]". It is unclear to what extent the bias toward the top of the list is due to the success of the search engine (Yandex in this case) in ranking the best suggestions at the top, or to users scanning down the list and giving up early. It is also unclear to what extent the observation that users ignore a suggestion which they go on to type themselves is due to delay in generation or transmission of the suggestions to the user.

Another possible explanation is that some users operate in a mode unlike the computer-game style suggested earlier. A proficient typist may be executing a "motor program" [8] which will result in fast, accurate typing of the query. Stopping that program and starting a new one requires attending to external stimuli (the suggestions) and is inefficient since it involves different actions (scrolling and clicking rather than typing) and probably moving a hand from one device (keyboard) to another (mouse or trackpad).

QAC response times must be very small, as a set of suggestions resulting from a prefix one or two characters shorter than the partial query visible in the search box, are unlikely to be useful. This means that QAC suggestions for an n-character partial query should be generated, transmitted, and rendered, before the user has committed to typing the $(n+1)$-th character. Since a proficient typist can type at a sustained rate of 60 words per minute, or one character every 167 msec, and since burst rates may be significantly higher, an effective QAC system should achieve round-trip times of under 100 msec. This implies that when a user is located on a different continent to the search system with which they are interacting, a local copy of the QAC index will be needed to achieve the required responsiveness.

2.6.1 *QAC logging and learning*

A fully logging QAC system might record:

- for each keystroke a suggestions-displayed record including the partial query and list of suggestion tuples.
- for each relevant click, an item-selected record, indicating the tuple clicked.
- whenever suggestions are displayed, and the user pauses typing for more than t seconds, a user-pause record.

Each record would of course include a timestamp, and IP address / userid. Suggestions-displayed records not followed by a selection or a user-pause indicate situations where the user has been given suggestions but has ignored them.

Identifying situations where the user has obtained the information they need from the QAC suggestions themselves is very difficult without the use of eye-tracking equipment. The user-pause records attempt to capture such situations but are subject to noise, as users may pause for other reasons.

Suggestions-displayed records are verbose, cause lots of network traffic, and are of more use in studying user behaviour than in tuning the QAC system. A system may choose instead to forego recording suggestions-displayed records, but augment the other two types with partial query and list of suggestion tuples.

Frequency of item-selected and user-pause records may be used to tune the weights associated with suggestion tuples.

2.7 General quality issues

Baeza-Yates et al [1] note a number of issues which need to be addressed in a practical QAC system. These are listed below. They relate to a simple QAC system on the Web but apply to a greater or lesser extent in an extended QAC system deployed in an enterprise.

1. Ensuring that suggestions are correctly spelled.
2. Filtering out "inappropriate" suggestions, such as those which are obscene or hate-filled or which promote violence. There have been a number of recent legal cases brought by people whose names have been associated with prostitution and bankruptcy through query completions.
3. Elimination of duplicate or quasi-duplicate suggestions, such as variants of a basic query are presented in singular and plural forms.
4. A possible need to diversify a set of suggestions.

5. Freshness. A QAC system needs to implement methods for ensuring that new content (e.g. breaking news, or an event scheduled at short notice) is accessible via completions.

6. Personalization or groupization of suggestions as discussed in several papers referenced here.

Another concern on the Web is that of spamming by artificial manipulation of suggestions by robotic query submission and clicking. Spamming and innappropriate suggestions are much less of a concern in the enterprise context, where suggestions do not rely solely on user interactions, where the audience is far smaller, and where the enterprise has control of its own QAC system.

2.8 Evaluation of QAC systems

Thus far, most query auto-completion evaluation has been restricted to simple QAC systems.

Shokouhi [9] assumed that the query eventually submitted by the user is the only correct suggestion and compared QAC methods on the basis of mean reciprocal rank of the correct suggestion, for all possible prefixes of that suggestion. Strizhevskaya et al [11] employ a similar methodology, based on all possible prefixes, but evaluate the top $k = 3$ suggestions against real submission frequencies. This type of method is not suitable for evaluation of X-QAC systems.

Ji et al [7] evaluated a staff directory auto-completion system but only on the basis of algorithmic efficiency and the amount of typing saved by the system.

Bhatia et al [4] employed user ratings: **Y** - suggestion is meaningful; **N** - suggestion is not meaningful; **D** - suggestion is a duplicate. They evaluated 40 test queries against two document collections with only two types of prefix: **Type A** - The prefix is the first word of the query; and **Type B** - The prefix is the first word of the query plus a random prefix of the second word.

The Bhatia method could be used to evaluate X-QAC; however, neither type of evaluation is able to answer whether all available information is accessible via query completion.

In the next section we investigate whether all suggestions in an X-QAC system can be guaranteed to be accessible, and measure the distribution of savings in keyboard effort across several X-QAC sets. We also empirically compare the times taken by users to locate staff contact details using a conventional search interface and an X-QAC system.

3. EXPERIMENT AND ANALYSIS

3.1 Our X-QAC system

The enterprise query completion system used in our experiments uses a tuple structure like that mentioned in the Introduction. Each tuple includes a weight which can be used to rank suggestions. The user interface relies heavily on the autocomplete widget in the open source JQuery UI project[7]. Like the system of Bhatia et al, ours is able to suggest corpus terms and phrases. In addition it can suggest past queries which resulted in at least one click[8] and strings derived from metadata fields, such as title, product name, etc., or from document annotations such as anchor text strings or folksonomy tags.

3.2 Coverage

An X-QAC system can be modelled as a list L of candidate suggestions, lexicographically sorted by trigger. It is assumed that there are no duplicates among the triggers. A user-typed prefix p is matched against the triggers. Obviously all matches for p are contiguous in L.

[7]http://jqueryui.com/about/

[8]Filtering out queries which do not result in a click avoids the possibility of suggesting a query which produces no useful results.

Table 1: A hypothetical sorted list L of candidate suggestions for use in a simple QAC system. If the system shows a maximum of $k = 1$ suggestion, prefixes of up to 4 characters are needed to guarantee all candidates are displayable. For $k = 3$ a prefix of only one character provides this guarantee.

Candidate	Length	Min-p ($k = 1$)	Min-p ($k = 3$)
Albert Abraham Michelson	24	1	1
Hendrik Lorentz	15	4	1
Henri Becquerel	15	4	1
Lord Rayleigh	13	1	1
J.J. Thomson	12	1	1
Marie Curie	11	1	
Philipp Lenard	14	2	1
Pierre Curie	12	4	1
Pieter Zeeman	13	4	1
Wilhelm Röntgen	15	1	1

Table 2: Details of university suggestion lists for which minimum prefix length distributions have been calculated for $k = 10$. SL, Q, P and C are each taken from a different university. Since we are only studying the X-QAC suggestion, there is no need for an index of documents from those universities.

	Description	List len.	Ave. len.	Pref. len.
SL	Staff list	21,943	13.5	5.0
Q	Clicked queries	28,887	18.3	5.9
P	Publication titles	99,819	83.0	7.2
C	Courses offered	251	19.2	1.7
U	Union of SL, Q, P, &C	128,260	56.6	6.8

Let us assume that the QAC system produces a set of up to k suggestions, and that there are m matching candidates for a given p. If $m > k$ then some matching candidates will not be displayed for that p. To guarantee that all of those candidates can be displayed, the user must continue typing.

Ideally, every possible suggestion should be able to be displayed by a QAC system. For example, for every name in a staff directory there should be a prefix (possibly the full name) which is guaranteed to show that name among a set of k suggestions. If this is not the case, then the QAC system fails and the user must resort to search or navigation. Clearly, this could happen in an unranked system whenever there exists a complete trigger which is a prefix of at least k other longer triggers. That problem can be avoided by ranking suggestions by increasing length.

THEOREM 3.1. *Let L be a list of X-QAC tuples. Let l_t be a member of L, whose trigger is t. Assume that no trigger is repeated. Let p be the query text so far typed by a user. Assume that the X-QAC system shows a set S of up to $k, k > 0$ suggestions whose triggers are prefixed by p. Let $|p|$ and $|t|$ represent the string lengths of p and t. Further assume that if there are more than k triggers starting with p then they are ranked by increasing length and the k shortest triggers are presented. Then $\forall t \exists$ some p such that $l_t \in S$.*

PROOF. There are two cases. Either $\exists p, p$ prefix of $t, |p| < |t|$, such that l_t appears in S, or not. If so the claim is satisfied. If not, the user must continue typing until $p = t$, whereupon l_t must appear in S because the suggestions are ranked by increasing length

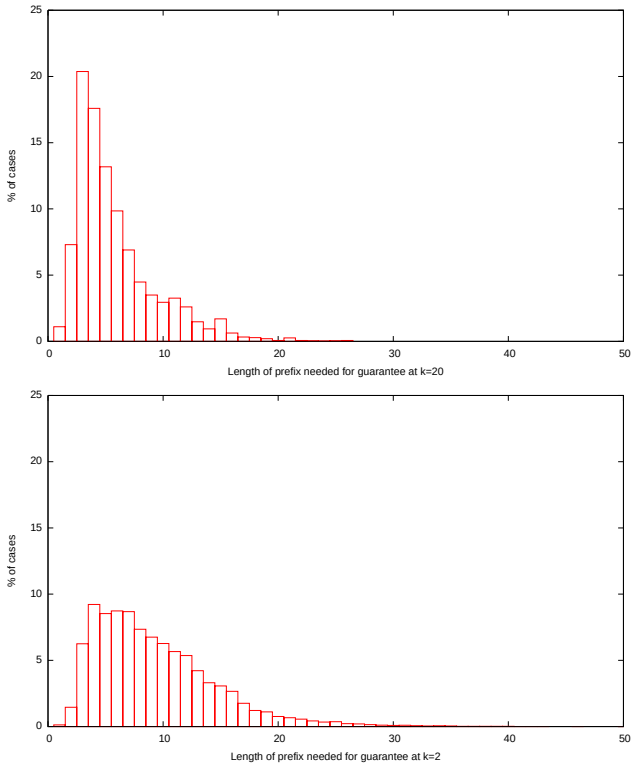

Figure 6: Histogram of prefix lengths needed to guarantee that every suggestion in the union list will appear, for two different values of k.

and there are no duplicate triggers. □

An indicator of the performance of a simple QAC system is the distribution of minumum prefix lengths necessary to guarantee that every suggestion in L can appear among the k suggestions. An example is shown in Table 1.

We have developed software to calculate minimum prefix-length distributions given L and k. Results for a number of suggestion lists and $k = 10$ are shown in Table 2. The histogram of required prefix lengths for the union of those lists is shown in Figure 6. For $k = 2$ 3.76% of cases require the query to be typed in full and rely on the QAC system prioritising shorter suggestions in the matching set. For $k = 20$ that percentage drops to 0.76%.

3.2.1 Discussion

Analysis in the previous section shows that it is possible to guarantee that every suggestion in a query auto-completion system can be accessed via a prefix, provided that there are no duplicates and if, at least in the last resort, the system favours the shortest suggestions.

The same guarantee can be made for a system which removes the requirement that the substring being matched must be a prefix of the suggestion. This is because a generalised matching system can be converted to a prefix one by adding an additional suggestion tuple for each prefix to be matched.

The empirical results presented in the previous section confirm that all suggestions are accessible for a range of different suggestion sources and for the union of all the sources. In every case there was a major saving in the amount of typing required. In many cases a suggestion can be located by typing only one or two characters.

The union suggestion set contained more than 100,000 items. In theory, those items could be located through navigation by making five choices among ten levels of navigation links. In practice the process of choosing among the labels at each level is time-consuming and highly error prone.

3.3 A user experiment

The aim of this experiment was to test whether users can satisfy an information need faster using an X-QAC system than when using a conventional search system. A convenience sample of 45 participants was asked to complete six tasks (following a practice task) using both a conventional search interface (Screen A) and an X-QAC system (Screen B). Each task required the participant to locate contact details (phone number and email address, if available) for a randomly chosen researcher from a list of 2184. The search system behind Screen A accessed an index of simple profile pages, one profile per researcher. The X-QAC system behind Screen B included multiple suggestion records for each researcher, each of which displayed the contact details for the suggested researcher, avoiding the need to submit a search or to visit a profile page. One trigger for each researcher included their full name and title, e.g. “Associate Professor Britney Spears”. Other triggers for the same person included fullname-without-title, e.g. “Britney Spears”, and surname-first, e.g. “Spears Britney”. Extra triggers were used when the name include a particle such as “von” or “van der”. There were an average of 4.01 triggers per researcher.

Participation was completely anonymous. Participants used a browser on their own computer. Results for the same person were linked using a cookie containing a random ID which was assigned when they visited the consent page and clicked on the ’I consent’ button.

In each task a participant was first shown screen C, including the full name and title of a randomly chosen researcher (e.g. “Associate Professor Britney Spears”). This is the person whose contact details were to be located. The participant was asked to type (not copy-and-paste) the query they planned to use when conducting a search. This was to assist them to remember the name they were searching for, and how it was spelled. It also encouraged them to think in advance about what query would be effective when searching for the person. After registering this query, they were taken to either screen A or screen B where they conducted a timed search. Then they proceeded to the other search interface and performed the task again.

To control for order effects the order of presentation of screens A and B alternated from one task to the next, and whether a participant saw A or B first in the first task was randomly determined.

Table 3: Results of the user experiment. *Ave. time* records the mean time taken to locate the needed contact details using this interface. Standard deviations are shown in parentheses. *No. of wins* records the number of users whose average time for this interface was less than for the other.

	A	**B**
Ave. time (sec.)	11.7(2.81)	7.11(2.65)
No. of wins	1	44

The time taken to complete the task for a particular screen was defined as the interval between presentation of the search screen (A or B) and the moment when the user clicked to acknowledge either that they had seen the contact information for the person, or that they had failed to find it (e.g. they arrived at the wrong profile or gave up for some reason.) Results were only included in the analysis if the user succeeded on both interfaces.

Table 3 shows that with only one exception, users were able to locate contact information faster using the X-QAC system. For the one dissenting voice only one trial was completed (with interface A first) and the time for the X-QAC interface was seven seconds longer than the average for any other user.

A paired t-test showed that the X-QAC interface was significantly faster than the conventional one ($t(44) = 11.97, p < 0.001$). The overall mean time to locate the contact information using the conventional search interface was 61% greater than that for the X-QAC interface. In terms of effect size, the mean standardized difference, Cohen's d was 1.52.($95\%CI = 1.05 - 1.99$).[9] The correlation between the conditions was 0.60, $p < 0.001$. This indicates that a person's relative performance on one interface was a reasonable predictor of their performance on the other.

All participants reported using either a laptop or a computer.

3.3.1 Discussion

Anecdotally, it seemed that much of the advantage of the query completion interface accrued not from the saving in keystrokes but from the avoidance of the need to load and interact with two additional screens (the search results and the staff profile.) The conventional search interface could have assisted by ensuring that contact details were shown in the snippets for each result returned.

However, another way in which the experiment was unrealistic was that the only documents indexed were the approximately 2000 user profiles, each in exactly the same format. The conventional search interface often returned only a single fully matching result, meaning that time taken to scan the results page was minimal. Time taken to locate contact information by conventional search may be slower when staff profiles are indexed along with hundreds of thousands of other web pages, and where the profiles may not have a standard format.

In order to avoid biasing the experiment in favour of one or other interface, our experiment involved a rather artificial task, in which the participant was given a correctly spelled full version of the target name. In a more realistic setting, the searcher may know only part of the researcher's name, may not know how to spell it, or may only know the person's nickname. Both interfaces could be augmented to improve performance in these cases.

4. CONCLUSIONS AND FUTURE WORK

Experience in commercial deployments and the results of our small user experiment suggest that X-QAC has considerable potential and in many scenarios can beneficially replace other methods of accessing information. We therefore envisage that websites and intranets of the [near] future may feature fewer navigation links and that the importance of the type-a-query, examine-results-page, click-on-a-link paradigm will decline.

Many avenues remain for future research work. To list just a few:

1. Modelling probabilities that a user whose intention matches suggestion S will start by typing p.
2. Devising a model for assigning weights to suggestions in extended QAC.
3. Determining how to combine the probabilities in 1 with the weights in 2.
4. Deciding what suggestions to present and how to present them when there are multiple suggestions from each of several categories.
5. Studying user interactions with QAC systems. Asking users why they sometimes ignore suggestions that they later go on to type.
6. Repeating the experiment with participants using smartphones and tablets
7. Determining the optimum number of suggestions to display.
8. Considering possible further extensions to the X-QAC model and to modes of interaction with it.

[9] I.e. the difference between the means of A and B is 1.52 standard deviations (difference relative to the combined variability of the populations). Cohen [5] considered $d > 0.8$ to be "large".

Acknowledgements. We thank the participants in our user experiment. The protocol of this experiment was approved by the Australian National University's Human Research Ethics Committee.

5. REFERENCES

[1] R. Baeza-Yates, B. Ribeiro-Neto, and Y. Maarek. *Modern Information Retrieval*, chapter Web Retrieval, pages 484–488. Addison Wesley, 2nd edition, 2011.

[2] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *Proceedings of WWW 2011*, pages 107–116, New York, NY, USA, 2011. ACM.

[3] H. Bast and I. Weber. Type less, find more: fast autocompletion search with a succinct index. In *Proceedings of ACM SIGIR 2006*, pages 364–371, New York, NY, USA, 2006. ACM.

[4] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. In *Proceedings of ACM SIGIR 2011*, pages 795–804, New York, NY, USA, 2011. ACM.

[5] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2nd edition, 1988.

[6] K. Grabski and T. Scheffer. Sentence completion. In *Proceedings of ACM SIGIR 2004*, pages 433–439, New York, NY, USA, 2004. ACM.

[7] S. Ji, G. Li, C. Li, and J. Feng. Efficient interactive fuzzy keyword search. In *Proceedings of WWW 2009*, pages 371–380, New York, NY, USA, 2009. ACM.

[8] R. A. Schmidt and T. D. Lee. *Motor control and learning*. Human Kinetics Publishers, 5th edition, 1988. ISBN 0-87322-115-X.

[9] M. Shokouhi. Learning to personalize query auto-completion. In *Proceedings of ACM SIGIR 2013*, pages 103–112, New York, NY, USA, 2013. ACM.

[10] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *Proceedings of ACM SIGIR 2012*, pages 601–610, New York, NY, USA, 2012. ACM.

[11] A. Strizhevskaya, A. Baytin, I. Galinskaya, and P. Serdyukov. Actualization of query suggestions using query logs. In *Proceedings of WWW 2012*, pages 611–612, New York, NY, USA, 2012. ACM.

[12] I. Witten, J. Cleary, J. Darragh, and D. Hill. Reducing keystroke counts with a predictive computer interface. In *Proceedings of IEEE Computer Society Conference on Computing to Aid the Handicapped*, November 1982.

[13] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proceedings of ACM SIGCHI 2003*, pages 401–408, New York, NY, USA, 2003. ACM.