Recent and Robust Query Auto-Completion

Stewart Whiting and Joemon M. Jose*
School of Computing Science
University of Glasgow
Scotland, UK.
{stewh,jj}@dcs.gla.ac.uk

ABSTRACT

Query auto-completion (QAC) is a common interactive feature that assists users in formulating queries by providing completion suggestions as they type. In order for QAC to minimise the user's cognitive and physical effort, it must: (i) suggest the user's intended query after minimal input keystrokes, and (ii) rank the user's intended query highly in completion suggestions. Typically, QAC approaches rank completion suggestions by their past popularity. Accordingly, QAC is usually very effective for previously seen and consistently popular queries. Users are increasingly turning to search engines to find out about unpredictable emerging and ongoing events and phenomena, often using previously unseen or unpopular queries. Consequently, QAC must be both robust and time-sensitive – that is, able to sufficiently rank both consistently and recently popular queries in completion suggestions. To address this trade-off, we propose several practical completion suggestion ranking approaches, including: (i) a sliding window of query popularity evidence from the past 2-28 days, (ii) the query popularity distribution in the last N queries observed with a given prefix, and (iii) short-range query popularity prediction based on recently observed trends. Using real-time simulation experiments, we extensively investigated the parameters necessary to maximise QAC effectiveness for three openly available query log datasets with prefixes of 2-5 characters: MSN and AOL (both English), and Sogou 2008 (Chinese). Optimal parameters vary for each query log, capturing the differing temporal dynamics and querying distributions. Results demonstrate consistent and language-independent improvements of up to 9.2% over a non-temporal QAC baseline for all query logs with prefix lengths of 2-3 characters. This work is an important step towards more effective QAC approaches.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Query Formulation

Keywords

Time; temporal; recency; query; text; completion; auto-completion

*Supported by the EU-funded project: LiMoSINe (288024)

Copyright is held by the International World Wide Web Conference Committee (IW3C2). IW3C2 reserves the right to provide a hyperlink to the author's site if the Material is used in electronic media.
WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2568009.

1. INTRODUCTION

Web searching is a ubiquitous activity performed by millions of users daily. However, cognitively formulating and physically typing search queries is a time-consuming and error-prone process. Spelling mistakes, forgetfulness and information need uncertainty often make textual query input laborious [21]. In response, search engines have widely adopted query auto-completion (QAC) as a means of reducing the effort required to submit a query [3, 26]. As the user types their query into the search box, QAC suggests possible queries the user may have in mind (which we refer to as *completion suggestions*), beginning with the currently input character sequence (i.e. *prefix*). The primary objective for effective QAC is to present the user's intended query after the fewest possible keystrokes, and at the highest rank in the list of completion suggestions. The most common approach to QAC is to extract past queries with each prefix from a query log, and rank them by their past popularity [3]; this assumes current query popularity is the same as past query popularity. Although this approach provides satisfactory QAC on average, it is far from optimal since it fails to take into account clues such as time or user context which often influence the queries most likely to be typed. As a result, this paper is most concerned with QAC approaches which are sensitive to changing query popularity – where the popularity is not predictable from long-term past query popularity observations.

As the web increasingly becomes a platform for real-time news and media, time plays a central role in information interaction. A substantial proportion of the daily query volume is the result of users turning to search engines for recent information about new or ongoing events [2, 20, 17]. Indeed, 20% of daily Google queries have not been seen in the past 90 days[1]. While the long-tail will inevitably account for a large proportion of these queries, many will be the result of short-term temporal events. We illustrate this with the following example.

Figure 1 shows the four completion suggestions offered by Google for the single character query prefix '***k***' on September 23rd, 2013. The list of completion suggestions indicates the historically most likely queries to be submitted with the prefix, possibly in the context of some undisclosed ranking features such as geo-location. Despite the recency and prominence of the Kenya Westgate mall terrorist attacks, the query '`kenya`' ranks very low in the completion suggestions. In Figure 2 we show the dramatic change in query popularity caused by the events – '`kenya`' becomes by far the most popular query. Yet, despite the fact that '`kenya`' is trending because of the ongoing events, Google's QAC fails to support users searching for information about the event as it ranks completion suggestions based on the past query distribution - which is

[1] `http://googleblog.blogspot.co.uk/2009/12/this-week-in-search-121809.html`

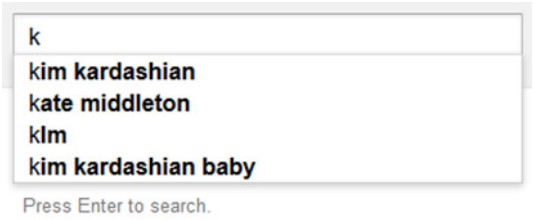

Figure 1: Google auto-completion suggestions for the query prefix '***k***'. Screenshot taken September 23rd 2013, during the ongoing Westgate shopping mall terrorist attack in Kenya.

no longer appropriate. Further compounding this issue, QAC for short prefixes (i.e. 1-2 characters) is often unsuccessful as there are such a large number of possible completion suggestions [3]. It is typically consistently popular 'head' queries that are provided as completion suggestions for such short prefixes (evidenced by the celebrity queries shown in this example). Therefore, there is a need to take the temporal aspect into account for effective QAC. This work is an attempt towards this objective.

Furthermore, queries that need to be included in completion suggestions fall into two main categories. The first category corresponds to *predictably* popular queries which are: (i) consistently popular, (ii) temporally recurring (e.g. at Christmas, in January, etc.) or (iii) known/foreseeable events and phenomena (e.g. TV episodes, sporting events, expected weather etc.). The second category corresponds to *unpredictably* popular queries related to entirely unforeseeable current events and phenomena (e.g. breaking news). Indeed, queries are likely to switch between these categories over time, making longer-term predictions problematic. Therefore, achieving optimal QAC effectiveness for all users, on average, is a trade-off between opposing objectives: (i) time-sensitivity, or *recency*, and (ii) *robustness*. Recency requires that completion suggestions include emerging and increasingly popular queries. Conversely, robustness requires that completion suggestions also reliably include long-term and consistently popular queries. These two goals are at odds – completion suggestions comprised only of short-term popular queries (e.g. in the last hour) might lead to lower ranking of many consistently popular queries. Alternatively, completion suggestions comprised of long-term popular queries (e.g. in the last year) will likely exclude the most recently popular queries. We propose an approach to address this trade-off by developing models which take recent evidence into account when possible.

This paper investigates the following research questions: **RQ1:** How effective is QAC in different scenarios? **RQ2:** Can relying only on recent query popularity evidence improve QAC effectiveness? **RQ3:** Can an optimal trade-off between recency and robustness be achieved for QAC? **RQ4:** Can short-range query popularity prediction (based on recent trends) improve QAC effectiveness? This paper makes four novel contributions: (i) practical novel approaches to QAC based on past query distributions, and optionally incorporating short-range query popularity prediction, (ii) extensive empirical analysis of QAC approach performance based on three real-world query logs (two English, and one Chinese) for varying prefix lengths, (iii) insight into the QAC approach parameters necessary to obtain optimal QAC effectiveness in different settings (e.g. language and temporal dynamics), and (iv) reproducible experimental findings due to the open datasets we study[2].

[2]The source code used to produce the datasets and results in this paper is openly available at `http://www.stewh.com/recent-robust-qac`

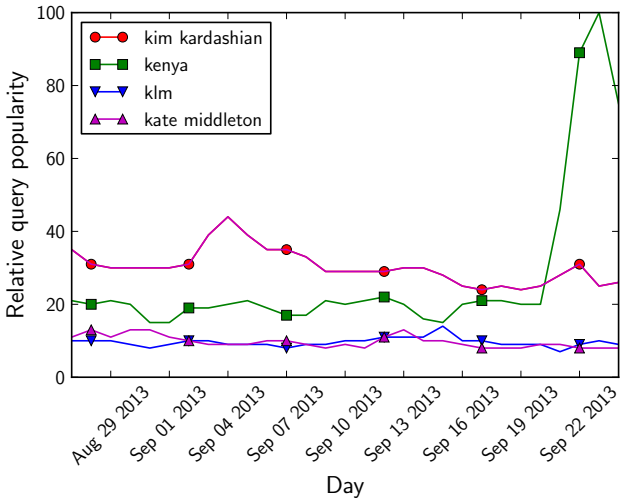

Figure 2: Google Search Trends indicating temporal popularity of the completion suggestions in Figure 1 during September 2013.

1.1 Motivation

Large-scale news, events and world phenomena undoubtedly play a central role in collective searching behaviour, and in turn dramatically affect query popularity (and hence the likelihood of a user typing a query) over time [2, 20, 17]. While relying on a long period of past query log evidence to rank completion suggestions will ensure QAC is robust for consistently popular queries, it will also have the effect of smoothing over sensitivity to recently popular queries. For example, imagine a scenario in which query q_1 is consistently popular, occurring 1,000 times every day in the query log. Aggregating query popularity over the past 30 day period would mean that query q_2, which became popular today, would need to occur 30,000 times before it outweighed the long-term popular query – despite the fact it is far more popular today than q_1 (and so should be a temporally higher-ranked completion suggestion). At the same time, reducing the aggregation period risks not fully representing the long-term q_1 query popularity, allowing arbitrary temporal fluctuations to reduce its ranking. In past work we studied the effectiveness of using a shorter query log aggregation period for QAC [29]. In this paper, we propose novel QAC approaches which address the recency/robustness trade-off to combine both recent and robust queries in completion suggestions.

Rather than relying passively on previously observed query popularity distributions for QAC, there is an opportunity to improve QAC over time by anticipating the current query popularity distribution based on previously observed trends – and better rank completion suggestions at each moment in time. As a result of this, in the past, long-term time-series modelling for query trends has been used to improve QAC for predictably recurring temporal query trends, such as seasonal (e.g. Christmas) or weekday/weekend related queries [26]. However, long-term time-series modelling can break down for many temporal events that are not always constant. For example, holidays (e.g. Easter) and natural phenomena such as the weather/seasons (e.g. planting spring bulbs) occur on an indeterminate schedule, where rigid temporal modelling of such queries might result in increased popularity prediction at incorrect times. Furthermore, for previously unseen and therefore unpredictable recent queries, time-series modelling often proved problematic due to lag and over-fitting [26]. This is due to the fact that time-series models often struggled to model increasing trends

quickly, and likewise, continued to predict increased popularity for some time after brief periods of popularity. Aside from the prediction issues of long-term time-series modelling, we are motivated to explore alternative approaches since many QAC systems will not have long-term and in-depth past query logs available for robust modelling. Therefore, many organisations may only have more recent smaller query logs available, yet still need to 'bootstrap' effective QAC systems with far less data and resources. As such, in this work we take a conceptually different approach to anticipating current query popularity for effective completion suggestion ranking: (i) relying only on recent query popularity evidence, and (ii) short-range query popularity prediction based on recently observed trends.

This paper is organised as follows. In Section 2 we present work related to QAC effectiveness. Section 3 outlines existing approaches to QAC (used as baselines), and proposes novel approaches for QAC. In Section 4 we describe our experimental setup and procedure, including query log datasets and real-time simulation methodology. In Section 5 we present experimental QAC approach results, which we discuss with regard to the research questions in Section 6. Finally, in Section 7 we make conclusions and recommend further work.

2. RELATED WORK

The majority of QAC research has concentrated on the inherent engineering complexity of providing efficient, responsive and scalable approaches that are resilient to typing errors [14, 4, 18, 9, 12]. Despite the prominence of QAC, there have been relatively few studies on improving QAC effectiveness; most likely down to the fact that there are few suitable query logs available outside industrial search engine companies for experimentation.

Exploiting the user's personal context and past query sessions has led to considerable increases in QAC effectiveness. For example, Shokouhi [25] exploits user profiles to model the likelihood a user is to issue certain queries, and therefore personalise completion suggestion ranking. Similarly, Bar-Yossef and Kraus [3] exploit users with shared query activity to improve QAC for other similar users.

Shokouhi and Radinsky [24, 26] use long-term time-series modelling of past temporal query patterns to improve QAC effectiveness for the proprietary Microsoft Bing query log. Popular queries recurring during specific temporal intervals, such as day/night, day of week, month, etc. were modelled to anticipate query popularity for completion suggestion ranking at different times. Shokouhi and Radinsky [26] propose the short time window technique we experiment with in this paper as a hard baseline (MLE-W, which they refer to as p_1, p_3, etc.). They note its relative effectiveness, particularly for correctly predicting short-term highly temporal and unpredictable queries for which time-series modelling is problematic. However, no detailed analysis of the performance impact of the time window period for each prefix length is provided. With related temporal intuition, Sengstock and Gertz [23] demonstrate (but not evaluate) a system which ranks completion suggestions depending on the time of day. In contrast to QAC approaches dependent on a past query log, Bhatia et al. [6] propose techniques to extract and rank completion suggestions such as common phrases found in indexed documents.

Short-range query popularity prediction has seen little attention. Golbandi et al. [16] develop a regression model to detect bursting queries for enhancing trending query detection. Meanwhile, Strizhevskaya et al. [27] measured prediction accuracy of various daily query popularity prediction models for the Yandex query log.

3. APPROACH

In this section we formalise the problem of QAC, and propose four distinct approaches (and one meta approach) to rank query completion suggestions based on query popularity observed over past periods of time. We focus independently on the recent temporal aspect of QAC in this work, and leave integrating user context (e.g. the user's past query topic preferences [25, 3] and location) with time to future work.

QAC is formally defined as follows: a set of possible query completion suggestions S at time q_t are ranked for the user-provided query prefix q_ρ. Only signals prior to q_t (e.g. past query popularity, or the user's query history) are available for ranking completion suggestions. The top k completion suggestions are provided to the user with minimal latency for their optional selection.

Two distinct approaches can be employed for completion ranking, in essence: (i) those which assume current query popularity is the same as recently observed query popularity, and (ii) those which predict current query popularity based on recent query popularity trends (i.e. the expected optimal completion suggestion ranking at q_t). In Sections 3.1, 3.2 and 3.3 we present approaches based solely on past query popularity distributions. Following these, in Section 3.4 we propose an approach which ranks completion suggestions based on short-range predicted query popularity. Finally, in Section 3.5, we propose a meta approach which uses online learning to adaptively optimise the parameters of the aforementioned approaches.

3.1 Maximum Likelihood Estimation

The common and straight-forward approach to completion suggestion ranking is Maximum Likelihood Estimation (MLE) based on past query popularity obtained from prior query log evidence [3]. In essence, MLE assumes that the current query popularity distribution will be the same as that previously observed, and so completion suggestions are ranked by their past popularity in order to maximise QAC effectiveness for all users, on average.

In past work, MLE is used as a baseline and referred to as '*MostPopularCompletion*' [3, 26]. We also use MLE, aggregating *all* query log evidence prior to the time of the user's query prefix input time (i.e. q_t) as our soft experimental baseline **MLE-ALL**.

MLE for prefix ρ and each query q with probability $P(q)$ in all past queries Q_ρ prefixed with the characters ρ, is formalised as:

$$MLE(\rho) = \arg\max_{q \in Q_\rho} P(q) \quad (1)$$

3.2 Recent Maximum Likelihood Estimation

The MLE completion suggestion ranking approach introduced in the previous section relies on the query popularity aggregated since the start of the past query log. Over time, the accumulation of query occurrences for popular queries smooths temporal variation, and thus promotes only the most consistently popular queries as completion suggestions. Recently emerging and popular queries will be outweighed by consistently popular queries, even though the emerging query might be considerably more popular at q_t.

To increase time-sensitivity we therefore propose using only the last N days of query log evidence (where $N = 2$, 4, 7, 14 or 28 days) for computing $P(q)$ at q_t (i.e., a single *sliding window period* of past query log evidence). We refer to this approach as **MLE-W**, and consider it our hard baseline.

The intuition underlying this approach is that a limited recent period of queries may more accurately reflect the current query popularity distribution. Similarly, although consistently popular queries will still be adequately reflected in the distribution, their total fre-

Algorithm 1 Last N Query Distribution (LNQ) for query prefix ρ

Require: N = number of last queries to track, n = flood limit for duplicate queries, $QueryStream_\rho$ = chronologically ordered stream of queries with prefix ρ.

1: $Deque \leftarrow$ FIFO deque
2: $DequeSize \leftarrow 0$ ▷ Track number of queries in the deque
3: **for all** query q at time q_t in $QueryStream_\rho$ **do**
  return Completions at $q_t \leftarrow$ MLE($Deque.Contents()$) ▷ Output completions
4: **if** $Deque.Count(q) + 1 <= n$ **then** ▷ Ensure n for q not exceeded
   $Deque.Push(q)$ ▷ Add query to deque
   $DequeSize \leftarrow DequeSize + 1$ ▷ Increment queries in deque
5: **end if**
6: **if** $DequeSize > N$ **then** ▷ Ensure N not exceeded
   $Deque.Eject()$ ▷ Remove oldest q
7: **end if**
8: **end for**

quency will no longer be great enough to outweigh the frequency of popular queries that only burst for short periods.

3.3 Last N Query Distribution

Prefix popularity, like many linguistic phenomena (such as word and phrase use in corpora) exhibits a power-law (or, *long-tail*) frequency distribution. As such, there are a small number of prefixes that are very common (e.g. '`the`' for a 3 character prefix), and a large number of prefixes that are very uncommon (e.g. '`zzt`'). In exploratory work [29], we found that although imposing a short sliding window for MLE-W could marginally improve QAC effectiveness overall, it often harmed completion ranking for less common long-tail prefixes – which together represent a large proportion of overall queries. As common prefixes account for a large volume of queries, query popularity distributions will be adequately reflected in a short sliding window. However, for less common prefixes, a short sliding window may reduce the available query popularity evidence over time - leading to poor completion suggestion ranking based on only the most recent and potentially randomly occurring query distributions, rather than robust longer-term distributions.

To address this issue, we investigated assigning sliding windows to different prefixes based on their popularity. Common prefixes were assigned a shorter sliding window (e.g. 2-7 days), while less common prefixes were assigned longer sliding windows (e.g. 7+ days). Empirical experiments found this approach problematic as the optimal sliding window identified for each prefix during the training period often changed since the prefix popularity can also change dramatically over time (for instance, if a query with a less common prefix becomes popular). We propose a continuous parameter learning meta approach to tackle this issue in Section 3.5.

Alleviating the need to impose any rigid sliding window time period, we propose the Last N Query Distribution approach – to track the last N queries observed for each prefix, and exploit their popularity distribution to rank completion suggestions at q_t in the same fashion as MLE (i.e. by most popular completion). This technique is inherently adaptive as it does not impose any strict time cut-off for past evidence. The trade-off between robustness and recency is controlled by capacity parameter N – a greater N promotes robustness by ranking completion suggestions using more past evidence, and conversely, a smaller N encourages recency by only using the most recent queries for the prefix in completion ranking.

An extremely popular query, especially for very short prefixes (e.g. 2 characters) may drive out other less common queries when N is relatively small. To avoid this, we introduce a second 'flood limit' parameter to the model, n, to constrain the upper limit of duplicate query instances to track (i.e. the maximum number of times the query '`google`' for prefix '`go`' is considered in the past N queries). When $n \geq N$, there is no upper limit on any single query.

Despite the practicality of this approach, to the best of our knowledge it has been neither proposed nor studied in past literature. We refer to the Last N Query Distribution approach as **LNQ**.

Implementation. LNQ is implemented by maintaining an adapted first-in-first-out (FIFO) double-ended queue (i.e. *deque* [19]) with capacity N for each prefix.

As each query is observed for a prefix (i.e., after the user has submitted the query), it is pushed to the deque, so long as the constraint defined by n is maintained. If the deque exceeds N, the oldest query observed is removed. Abstract implementation of LNQ is presented in Algorithm 1.

3.4 Predicted Next N Query Distribution

As query popularity distributions are not stationary over time [20, 24, 26], the query popularity distribution observed in the past, no matter how recent, will in many cases be different to the actual present distribution at q_t. Consequently, for queries which fluctuate in popularity over time, QAC based passively on past query popularity distributions may always have a lag in effectively suggesting and ranking completions.

Rather than assuming present query popularity distributions are the same as those previously observed, we propose to predict short-range (i.e. current) query counts using recently observed query popularity trends (e.g. stationary/rising/falling query counts). In contrast to past work [26], rather than exploiting long-term repetitive trends (e.g. Christmas, New Year, Easter, etc.), this work relies on predicting current query popularity based only on the most recently available data.

Time-series modelling techniques have been used to predict query popularity. Auto-regressive time-series modelling has proven effective for long-term predictably recurring queries [26, 27], e.g. those likely to be seen daily, monthly, quarterly, etc. Such techniques require a vast amount of past data to avoid overfitting, fall down with indeterminate time schedules and are often ineffective for unforeseeable newly popular queries. Trending query detection can be improved through query count prediction using minute-based linear auto-regression models for the 30 minutes prior to q_t [16]. Of course, trending queries are concentrated in the very recent query log, making such specific short-range prediction feasible.

Time-series query popularity prediction approaches require adequate query popularity data to construct a robust time-series and then make reliable predictions. However, since query popularity exhibits a power-law distribution, the majority of queries are relatively low frequency, and subsequently have relatively sparse time-series, making accurate predictions problematic. Yet, these queries still need to be effectively ranked in QAC.

Henceforth, to side-step the issue of sparse and potentially erratic query popularity time-series, we frame the problem of query popularity prediction for completion suggestion ranking as: *given trends in the last N queries observed with a prefix, what is the chance of observing the query again in the next N queries with the prefix?* The last N queries may have been observed in the last 5 minutes, or 5 hours, and so forth, depending on the prefix popularity. Although this approach is still time-aware, it is not dependent on query popularity in a rigid temporal scale, but instead the rate of all queries with the prefix observed over time. We note our approach is not exclusive to long-term temporal modelling [26], and we leave combining the two distinct approaches to future work[3].

[3] Currently, the short-term query log data we have available severely limits any longer-term temporal modelling opportunities.

Using the predictive model outlined in the following section, this approach ranks completion suggestions based on their predicted popularity distribution at q_t. We refer to the Predicted Next N Query Distribution approach as **PNQ** in Section 5, where we note predictive model accuracy and present QAC experimental results.

3.4.1 Incrementally-trained Predictive Model

To capture recent periods of query popularity, a sequence of successive LNQs is employed. Rather than using a single LNQ containing the last N queries observed with a given prefix (as in the previous section), we chain multiple LNQs together to track windows of the last N queries with the prefix. For instance, three $N = 100$ LNQs can be used to track the last 100, 101-200 and 201-300 queries observed for a given prefix. In Section 5 we denote the successive LNQs used for the regression model in PNQ experiments with the nomenclature $M \times N$, where M is how many LNQs are chained together, and N is the capacity of each separate LNQ.

We propose a multiple linear regression model based on the query counts observed in the successive LNQs, to predict the count of recently observed queries in the next 100 queries with the same prefix (note, a single model is trained for all prefixes). We found multiple linear regression to be the best performing regression scheme in early empirical studies.

Conventional machine learning techniques rely on batch learning – in which a finite and stationary set of training examples is used to produce a static model that best fits all known training instances. This is perfectly adequate when the number of training instances fits in memory, and the nature of the model is unlikely to change. However, for query popularity prediction, there is an online stream of training instances, and unforeseen temporal factors which may affect the model, such as the time of day [16], day of week, public holidays and weather, etc. Hence, a model trained on one period may fail to generalise to a later period.

Recent research attention has focused on developing machine learning techniques (e.g. for classification, regression and ranking) that can incrementally adapt their model based on new training instances seen in an evolving data stream [31], such as that available from a query log. Conceptually, in an online scenario such models provide predictions based on past training, however, they continuously incorporate training instances as they become available to further reduce error through feedback. In this work we utilise Stochastic Gradient Descent (SGD) to incrementally learn a regression model which minimises prediction error over time. Efficient SGD implementation details can be found in [30]; we set an SGD learning rate of 0.01. Since SGD is sensitive to scaling, query counts are normalised between $[0, 1]$. Traditional k-fold cross-validation is not applicable to streaming settings since it would disorder the temporal data sequence [15].

For PNQ, the model training routine is as follows:

(**1**) After every 100 queries observed with a given prefix, the successive LNQ past query counts for each respective q with the prefix are stored as a future training instance, (**2**) Following a further 100 queries, the prediction target variable – each q's count in the following 100 queries with the prefix, is now known, (**3**) SGD adapts the regression model co-efficients based on each q training instance.

3.5 Online Parameter Learning

All the experimental approaches we have outlined in the past three sections (i.e. MLE-W, LNQ and PNQ) have one or more parameters that require tuning for optimal QAC performance (e.g. window size for MLE-W, N/n for LNQ, and $M \times N$ for PNQ).

In Section 5 we explore the parameters necessary to achieve best performance for each prefix length and query log, on average. Of course, since different prefixes have varying characteristics (recall the power-law distribution of prefix popularity we discussed in Section 3.3), prefixes of the same length (e.g. 'th' and 'zh') may have different optimal QAC approach parameters, and these could change over time as new behaviour emerges (e.g. a major new event causing a great deal of change for a previously stationary prefix).

The final approach (in essence, a *meta*-approach) we propose in this paper is based on online parameter learning for the aforementioned approaches. This requires that the best performing parameters for each approach, and prefix, are continuously re-evaluated and selected to improve QAC at q_t. To this end, we exploit the nature of QAC whereby the correct completion suggestion is known almost immediately after suggestions were initially provided – whether because the user has selected it, or manually typed it. This characteristic enables online measurement of the effectiveness of each parameter, in real-time, and thus facilitates online QAC optimisation. Since all the QAC approaches we propose are lightweight, this approach is still very practical in real-world applications.

Online parameter learning tracks the QAC performance (i.e. the mean reciprocal rank, discussed in Section 4.2) of each individual approach parameter for each prefix in a training horizon of the last Δ queries typed. At q_t, the best performing parameter is selected to rank the completion suggestions for the prefix, hence, the parameters are free to change if another becomes optimal.

Since we use the previous approaches in combination with this meta-approach, we refer to these as **O-MLE-W**, **O-LNQ** and **O-PNQ** in Section 5.

4. EXPERIMENTAL SETUP

We conduct experimentation using AOL 2006, MSN 2006 and Sogou 2008 query log datasets. By experimenting with the millions of queries contained in each query log, we obtain a representative view of how each approach would perform in a real-world setting. Comparing results from multiple query logs validates QAC approaches in different scenarios in which there may be diverse user population biases, temporal behaviours and sparsity levels.

4.1 Query Log Datasets

AOL [22], MSN [1] and Sogou 2008[4] are publicly available query log datasets. Each dataset is provided in a raw format containing all user interactions in chronological order. Each interaction can represent a query submission (with no result clicks), a result click or result page change, etc. Each interaction in the dataset therefore duplicates the query originally typed for identification. For realistic QAC experimentation we extract only the queries a user physically input (which we refer to as *typed queries*).

We consider the first appearance of a query in a user's session (and the associated timestamp) as typed queries. Where not provided in the dataset, sessions were identified using a 30 minute interaction time-out [7]. Details of each query log and its associated typed queries dataset are provided in Table 1.

AOL Issues. Early analysis discovered a relatively large number of short bursts of what we suspect is bot/spamming interaction activity in the AOL query log. Many generic queries such as '`personalfinance`', '`aolcelebrity`' and '`computercheckup`' (both with and without spaces) occur with notably uniform interaction spacing (e.g. every 30-60 seconds).

Although AOL contains more queries than MSN, it also has greater breadth by covering a three month period. However, the dataset is sampled by user, rather than randomly over all queries

[4] `http://www.sogou.com/labs`

Table 1: AOL, MSN and Sogou query log dataset statistics.

	AOL	MSN	Sogou
Language	US English	US English	Simp. Chinese
Start date	2006-03-01	2006-05-01	2008-06-01
End date	2006-05-31	2006-05-31	2008-06-30
Training days	28	14	14
Testing days	64	17	16
Days covered	92	31	30
Typed queries			
Total	18.1M	11.9M	25.1M
Avg. per hour	11,764	29,069	65,364
Avg. per day	196,411	382,589	836,660
Query length (chars)			
Avg.	17.03	17.38	6.53
Stdev. ($\pm$)	11.02	12.15	4.01

[2]. Therefore, the breadth of coverage may have biased querying distributions since many web search users engage in information re-finding behaviour (as many as 40% of queries are for re-finding) [28]. Moreover, there is a missing day of data (May 17th) [2].

Additionally, we filtered a large volume of navigational queries containing URL substrings: `.com`, `.net`, `.org`, `http`, `.edu` and `www.` from the dataset.

Comparing Datasets. Although the exact sampling and construction of each query log is unknown, together these three datasets have differing characteristics that allow us to make in-depth comparisons between QAC approaches. AOL seems much more raw compared to Sogou and MSN, which have both been filtered to remove identifying information and adult queries. Spam issues we highlighted for AOL may prove problematic for any results based on it alone. As MSN and Sogou provide one month datasets, they offer greater daily depth in comparison to AOL's breadth over three months.

As AOL and MSN contain queries in the 26-character Latin (a-z) alphabet, queries are on average around 17 characters in length. In contrast, Sogou queries are almost all in the simplified Chinese alphabet – which has around 3,500 commonly used characters (covering 99.5% of all characters used) [11], although search query language distributions often differ [8]. Accordingly, Sogou queries are on average only 6.5 characters in length, as each character carries greater information than the Latin alphabet. As our proposed QAC approaches are blindly based on single characters and the temporal dimension, the alphabet/language of the queries is irrelevant.

Temporal Trends. Many studies have offered extensive insight into the temporal trends of longer-term but proprietary query log data [20, 5, 2, 24]. In our datasets we identified many popular queries with the majority of their occurrences concentrated within short periods - that is, queries related to unpredictable ongoing events and phenomena which need to be included in completion suggestions. In AOL, popular short-term temporal queries include: '`amelia earhart pictures`', '`karl der grosse`', '`the simpsons live action`' and '`leisure suit larry`'. Likewise, in MSN among the most popular are: '`stephen colbert`', '`poison milk`', '`ohio bear attack`' and '`kimberley dozier`'. We expect that the proportion of unpredictable recent queries will have substantially increased in more recent query logs, given the rise in real-time internet media.

4.2 Experimental Procedure

In this section we outline the experimental procedure we use to investigate the four research questions proposed in Section 1.

Real-time QAC Simulation. As QAC operates in a real-time scenario, our experimental methodology simulates a real-time user search scenario: when the user types a prefix, they receive completion suggestions based only on evidence prior to the time of their query. QAC effectiveness is measured by the presence and rank of a ground-truth match in each set of completion suggestions.

The chronologically ordered query log datasets (discussed in the previous section) provide a stream of ground-truth user-typed queries. We assume that the user would have selected their completed query if it had been presented to them. We discuss matching ground-truth queries and completion suggestions in the following section.

We run each approach for all 2- to 5-character query prefixes. Experiments for each prefix length were run independently, hence a successful completion suggestion for a 2-character prefix has no bearing on the later evaluation of a 3-character prefix.

To emulate a real user interface scenario (such as that shown in Figure 1), experiments are based on the user seeing at most the four highest-ranking completion suggestions for each prefix they input.

Evaluation Metric. As in past QAC work [3, 26, 25], we rely on Mean Reciprocal Rank (MRR) to measure the effectiveness of each QAC approach. Reciprocal Rank (RR) has typically been used for evaluation in retrieval situations in which there is a single relevant ranked item [10]. MRR reflects the user interaction model of QAC: a higher-ranked completion suggestion is more beneficial, but the difference in ordering of lower-ranked completion candidates is less significant. That is to say, there is less noticeable difference between a correct completion suggestion ranked at either the 3rd or 4th position, compared to the 1st or 2nd position. For a set of completion suggestions S and the user's intended (i.e. completed) query q', RR is computed as:

$$RR = \frac{1}{Rank(q' : S)} \tag{2}$$

If no match for q' is present, $RR = 0$ (avoiding divide by zero errors). MRR is computed as the arithmetic average of RR for all queries (thus it is in the range $[0,1]$). Because of the large number of queries over which we compute the MRR (i.e. 12-25M), it is relatively insensitive, so we round MRR values to 4 decimal places.

Because of the non-linear discounting of RR, a small percentage increase in MRR can be the result of a much larger percentage increase in completions with an RR > 0 (depending on the distribution of RRs obtained for each completion rank). In the most extreme case, an increase in MRR could correspond with a four-fold increase in completion suggestions with a RR > 0. In terms of reputation for a real-world QAC system, we argue that even showing an appropriate completion suggestion at a lower rank will be valuable, so even small improvements in RR are noteworthy.

We consider a literal lower-case exact string match between completion suggestion and ground-truth as a successful match. Although there often may not be an exact literal match between the user's intended (ground-truth) query and an appropriate completion suggestion (e.g. '`A.A.`', '`AA`' and '`American Airlines`' are semantically equivalent). Similarly, the ground-truth query might be a spelling mistake which would not have occurred if the user had been given the correct completion. We leave this advanced evaluation to future work.

4.2.1 Experiment Settings

Baselines. As in recent past work [3, 26], we consider MLE-ALL (i.e. '*MostPopularCompletion*') as our soft baseline. In exploratory work [29], we found MLE-W usually outperforms MLE-ALL so we consider it as our hard baseline – with which we com-

pare all our approaches in Section 5. Results we present for MLE-W, particularly for shorter prefixes, are different to those reported previously [29] due to fixed data issues.

We are unable to reliably re-implement past long-term temporal modelling as a QAC baseline [26], as the query logs we have available are all very short-term compared to the proprietary multi-year datasets the paper authors utilised. As we would need to separate the datasets into train and test periods (and only test on periods after training for temporal validity), over-fitting to a small training set, or alternatively, relying on a very small and potentially unrepresentative test set would be inevitable. In any case, we consider work relying on short-term query trends to be complementary to long-term query trend modelling, and expect future work to investigate combining the techniques.

Query Log Training Periods. In most real-world scenarios, a pre-existing query log would be used to train (or, *bootstrap*) a QAC system. We dedicate the first 14 days of Sogou and MSN, and 28 days of AOL to this purpose. Accordingly, all MRR results presented are computed from 00:00 on 2006-03-29, 2006-05-15 and 2008-06-15 onwards for the remaining duration of AOL, MSN and Sogou, respectively.

Since QAC performance can fluctuate over time, it is unfair to compare two approaches using the same query log but over different time periods. As such, all MRR comparisons we make (e.g. relative percentage change) are computed over the same time period (and therefore, the same set of queries).

5. RESULTS

We conducted a large number of experiments to test QAC approaches with several parameters for three query logs and prefixes of 2-5 characters. In this section we present the most interesting positive and negative results that experimentally validate the approaches we propose, and offer insight for real-world QAC system design.

The aggregate statistical power of MRR measured over tens of millions of queries in the datasets with which we experiment means that all the results we report are statistically significant, at least according to common *t*-tests [13]. Our analysis therefore concentrates on the effect size over the baseline of each approach – that is, relative MRR change.

5.1 Soft Baseline: MLE-ALL

Table 2: **MLE-ALL** Approach (Soft Baseline) - QAC MRR observed after initial training periods of 14 days (MSN/Sogou) and 28 days (AOL), with query prefix ρ lengths of 2-5 characters.

	Query Log / MRR		
ρ	**AOL**	**MSN**	**Sogou**
2	0.0962	0.1124	0.4117
3	0.1527	0.1702	0.5487
4	0.1969	0.2106	0.5970
5	0.2304	0.2340	0.6129

In Table 2 we report the absolute MRR achieved by MLE-ALL QAC for prefixes of 2-5 characters for each query log.

Most apparent, for all query logs, is that QAC is considerably more effective with a longer (i.e. more specific) prefix. This is unsurprising, given that each extra character in the prefix vastly reduces the space of possible completion suggestions, therefore increasing the chance of a completion suggestion match [3].

QAC is always more effective for MSN than for AOL, although at 4-5 characters their performance begins to converge. Expectedly, QAC is much more effective for Sogou – attributable to the greater specificity of each character in the larger Chinese alphabet (and hence, shorter Chinese queries).

5.2 Hard Baseline: MLE-W

In Table 3 we report the MRR change achieved by MLE-W QAC relative to the MLE-ALL soft baseline, with a 2- to 28-day sliding window period for prefixes of 2-5 characters. A 28-day sliding window period MRR is not reported for MSN and Sogou because they do not cover enough time to reliably compute it.

In all cases, using a sliding window period of past evidence has a considerable effect on overall QAC performance compared to the soft baseline. For AOL there is a sliding window period which can improve QAC performance over MLE-ALL for prefix lengths of 2-3 characters, albeit only marginally for 3 character prefixes. A 7 day sliding window period improves QAC performance by up to 1.1% for shorter prefixes of 2 characters.

Meanwhile for MSN, QAC for 2- or again, marginally, 3-character prefixes outperforms the soft baseline when using a small sliding window period. Specifically, we see the best performance improvement, of almost 3%, when using a 4-day sliding window.

Finally, for Sogou, although QAC performance increased by up to 3.6% for 2-character prefixes with a 2-day sliding window period, it is always harmed for longer prefixes.

For all query logs, however, using a sliding window period between 2- and 28-days always impairs QAC performance compared to the soft baseline for 4- or 5-character prefixes.

All further results of our proposed approaches reported in this section are for 2- and 3-character prefixes, and reported relative to the highlighted hard baseline findings.

Table 3: **MLE-W** Approach (Hard Baseline) - QAC MRR change (relative to the MLE-ALL soft baseline) observed for past 2- to 28-day sliding window periods of query log evidence, with query prefix ρ lengths of 2-5 characters for AOL, MSN and Sogou query logs. Best performing sliding window periods are highlighted.

	Query Log / Sliding Window Period (days)				
ρ	2 days	4 days	7 days	14 days	28 days
			AOL		
2	-1.12%	+0.47%	**+1.12%**	+1.03%	+0.69%
3	-11.18%	-4.91%	-2.01%	-0.45%	**+0.14%**
4	-19.58%	-11.46%	-6.71%	-3.14%	**-1.04%**
5	-26.00%	-17.01%	-11.16%	-5.97%	**-2.39%**
			MSN		
2	+2.60%	**+2.95%**	+2.28%	+0.68%	-
3	-5.13%	-1.50%	-0.11%	**+0.27%**	-
4	-11.05%	-5.77%	-2.56%	**-0.58%**	-
5	-15.84%	-9.45%	-4.97%	**-1.47%**	-
			Sogou		
2	**+3.59%**	+3.36%	+3.03%	+2.47%	-
3	-5.46%	-2.96%	-1.52%	**-0.19%**	-
4	-7.81%	-4.86%	-2.90%	**-0.87%**	-
5	-10.22%	-6.60%	-4.11%	**-1.37%**	-

5.3 Approach: LNQ

In Table 4 we report the MRR change achieved by LNQ QAC relative to the hard baseline (highlighted in Table 3) for prefixes of 2-3 characters. We experiment with an LNQ capacity parameter of $N = 100, 200, 400, 800, 1200$. In preliminary experiments we found that n (i.e. the flood limit parameter) had a negligible effect on our results (at least when n was $\geqslant 0.5N$), so experiment with N and n as a single parameter (thus disregarding n). Future work will study the effect of n as a query flood limit in real-world scenarios.

LNQ with one or more parameter settings is able to significantly outperform the hard baseline for all query logs for prefix lengths of 2-3 characters. The greatest LNQ QAC performance improvements are observed for Sogou, where any N/n setting is able to exceed the hard baseline for all prefix lengths. In particular, for 2-character prefixes LNQ exceeds hard baseline performance by 4.9% and 3.9% for Sogou and MSN, respectively.

Optimal N and n parameter settings vary between query logs. Sogou is less sensitive to N, whereas MSN and AOL performance fluctuates more with different settings. This may in part be because Sogou has stronger temporal query distributions since it has the most queries concentrated in the shortest time, along with very specific prefixes. Interestingly, a clear pattern emerges whereby the optimal N and n parameters for LNQ are near constant for each query log, regardless of the query prefix length, indicating that parameters are dependent on the query log and its intrinsic temporal characteristics rather than prefix length.

Table 4: **LNQ** Approach - QAC MRR change (relative to the MLE-W Hard Baseline) observed for each N/n parameter, with query prefix ρ lengths of 2-3 characters for AOL, MSN and Sogou query logs. Best performing N/n are highlighted.

	Query Log / LNQ N (and n) Parameter				
ρ	100	200	400	800	1200
	AOL				
2	-8.26%	-3.28%	-0.53%	+0.46%	**+0.83%**
3	-4.04%	-0.95%	+0.44%	+0.78%	**+0.81%**
	MSN				
2	-2.98%	+1.24%	+3.27%	+3.82%	**+3.92%**
3	-1.78%	+0.81%	+1.92%	**+2.16%**	+2.05%
	Sogou				
2	+4.58%	**+4.88%**	+4.65%	+4.09%	+3.66%
3	+3.17%	**+3.21%**	+3.03%	+2.75%	+2.55%

5.4 Approach: PNQ

In Table 5 we present the MRR change achieved by PNQ QAC, relative to the MLE-W hard baseline, for prefixes of 2-3 characters. We experiment with PNQ linear regression models of 10×50, 5×100, 20×50, 10×100 and 5×200 (this $M \times N$ nomenclature is explained in Section 3.4).

With the appropriate regression model, PNQ is able to marginally outperform LNQ for all query logs and prefix lengths, except for MSN with 3-character prefixes. However, despite the additional overhead of making predictions, there is relatively little improvement over the optimal LNQ parameter.

In contrast with MSN and Sogou, PNQ for AOL is very sensitive to the parameters, although AOL on the whole does have much lower QAC effectiveness than both MSN and Sogou. There is no optimal parameter for all query logs, although the same parameter is optimal for different prefix lengths in the same query log.

Predictive Model Accuracy. Although we focus on the final outcome (i.e. the QAC MRR) in our scenario, to illustrate the accuracy of the PNQ multiple linear regression model in predicting upcoming query counts, we report some further analysis for a 5×200 linear regression model.

In the first two days of AOL, MSN and Sogou, there were 160K, 287K and 170K training instances generated, respectively. For each query log, the regression model co-efficients were as follows: AOL - $\{0.19, 0.1, 0.07, 0.07, 0.07\}$, MSN - $\{0.26, 0.11, 0.05, 0.04, 0.03\}$ and Sogou - $\{0.4, 0.05, 0.03, 0.017, 0.01\}$ (the left-most co-efficient is for the query count in the most recent 200 queries in this model). Thus, a varying temporal dynamic between query logs is apparent.

Predicting query counts in the next 100 queries with a given prefix, these models achieved R^2 (i.e. co-efficient of determination) of 0.9, 0.92 and 0.98, and a RMSE (i.e. Root Mean Squared Error) of 0.67, 0.82 and 1.5 for AOL, MSN and Sogou, respectively.

Table 5: **PNQ** Approach - QAC MRR change (relative to the MLE-W Hard Baseline) observed for each $M \times N$ regression model, with query prefix ρ lengths of 2-3 characters for AOL, MSN and Sogou query logs. Best performing regression models are highlighted.

	Query Log / $M \times N$ Regression Model				
ρ	10×50	5×100	20×50	10×100	5×200
	AOL				
2	-0.25%	-0.35%	+0.82%	**+0.85%**	+0.80%
3	+0.68%	+0.61%	**+1.12%**	**+1.12%**	+1.07%
	MSN				
2	+3.29%	+3.30%	+3.86%	+3.95%	**+4.06%**
3	+1.60%	+1.70%	+1.96%	+2.02%	**+2.11%**
	Sogou				
2	**+5.14%**	+5.13%	+5.11%	+5.11%	+5.04%
3	**+3.38%**	+3.36%	+3.35%	+3.34%	+3.29%

5.5 Approach: O-MLE-W, O-LNQ & O-PNQ

In Table 6 we report MRR change, relative to the hard baseline, achieved by the online parameter learning meta-approaches: O-MLE-W, O-LNQ and O-PNQ, with a learning horizon $\Delta = 100, 300, 600$ queries. For each QAC approach (i.e. MLE-W, LNQ and PNQ), to rank completion suggestions for each prefix at q_t, we identify the highest performing approach parameter for each prefix based on past queries with the same prefix observed in the learning horizon Δ. Each QAC approach may use the parameters from the previously reported experiments (e.g. $N/n = 100, 200, 400, 800, 1200$ for LNQ).

The results obtained by the online parameter learning meta-approach represent the highest QAC effectiveness achieved by any of the approaches we propose, even if only marginally for MSN and Sogou. O-LNQ proves optimal for MSN and Sogou, while O-MLE-W is the best approach for AOL. For MSN and Sogou there is typically little difference between O-LNQ and O-PNQ performance. Relying on a greater number of past queries in the training horizon (i.e. $\Delta = 300, 600$) yields greatest QAC performance; however, performance is often insensitive to Δ, especially for Sogou.

Table 6: **O-** Online Parameter Learning Approaches - QAC MRR change (relative to the MLE-W Hard Baseline) observed for each approach and training horizon (Δ queries), with a query prefix ρ length of 2 characters for AOL, MSN and Sogou query logs. Overall best performing approach and Δ are highlighted.

	Query Log / Δ		
Approach	100	300	600
		AOL	
O-MLE-W	+1.45%	**+1.57%**	+1.54%
O-LNQ	-0.36%	+0.41%	+0.72%
O-PNQ	+0.54%	+0.63%	+0.70%
		MSN	
O-MLE-W	+3.18%	+3.45%	+3.40%
O-LNQ	+3.32%	+3.90%	**+4.11%**
O-PNQ	+3.80%	+3.90%	+3.94%
		Sogou	
O-MLE-W	+3.83%	+3.75%	+3.67%
O-LNQ	+5.42%	**+5.43%**	+5.42%
O-PNQ	+5.27%	+5.28%	+5.28%

5.6 Time-based Approach Performance

To characterise QAC approach performance over time, in Figure 3 we present the 6-hourly MRR of four QAC approaches. Each approach uses the best performing parameters for Sogou with a 2-character prefix: MLE-ALL (soft baseline), MLE-W (2-day sliding window - hard baseline), LNQ ($N/n = 200$) and PNQ (10×50).

Very apparent is the natural QAC performance fluctuation for all approaches over time. The 'cold-start' period of low MRR (i.e. when there is little past query popularity evidence to rank completion suggestions) is prominent in the first four 6-hour periods which exhibit comparatively low MRR.

While there is some degree of daily cyclical pattern (note the repeated spikes, likely due to underlying day/night querying behaviour changes, since MRR is normalised), there is no obvious long-term pattern, suggesting a random degree of query popularity distribution over time with which QAC struggles to assist reliably.

Although the performance of all approaches is largely similar to begin with, beyond this (i.e. from twenty 6-hourly periods onwards), substantial variance between approaches is observed. MLE-W starts by under-performing compared to MLE-ALL. However, in the latter half of the month (i.e., sixty 6-hourly periods/15 days onwards), MLE-ALL gets progressively and consistently worse compared to MLE-W, probably because it becomes increasingly insensitive to changing query distributions because of the accumulation of overbearing historic query popularity evidence.

LNQ and PNQ mostly reflect one another in performance, with occasional minor deviations. Both approaches consistently outperform MLE-ALL and MLE-W approaches at almost all times in the latter half of the month, and increasingly so towards the end of the month.

6. DISCUSSION

In the following section we discuss the results we presented in the previous section in relation to the four research questions outlined in Section 1.

RQ1: How effective is QAC in different scenarios? The results outlined in Section 5.1 demonstrate that QAC effectiveness

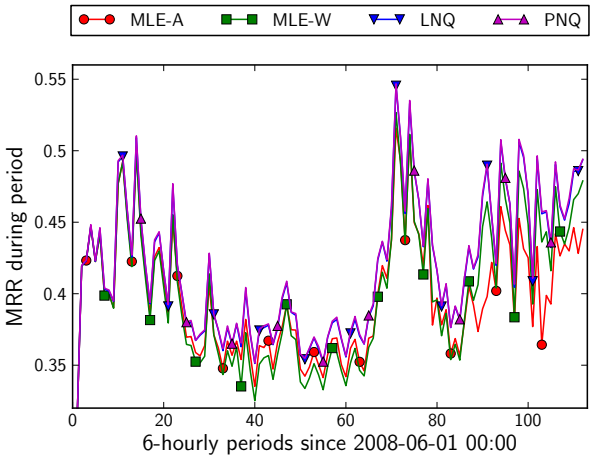

Figure 3: QAC performance every 6-hourly period in June 2008, for 4 QAC approaches with Sogou and a 2-character prefix.

varies considerably in different scenarios, that is, for different query logs and prefix lengths in our experiments. QAC performance for shorter, non-specific prefix lengths (i.e. 2-3 characters) is always relatively low, as there is a large space of possible completions and much greater uncertainty regarding the user's intended query [3].

Chinese users have far more effective QAC since the Simplified Chinese alphabet is much larger than the Latin alphabet, reflected by a much shorter average Chinese query length (7 characters, compared to 17 for the Latin alphabet).

Even for the Latin alphabet, QAC effectiveness is not equal between scenarios – at least for shorter prefixes, e.g. 2-4 characters, in which MSN achieves up to 16.8% greater baseline QAC performance than AOL. Indeed, AOL performs relatively poorly in all our experiments.

In part, this may be caused by a number of factors. Firstly, although AOL contains more queries, these are spread sparsely over a three month period, whereas MSN queries are concentrated in a one month period. This could suggest that a search engine serving more queries is able to make better completion suggestions since it has a larger sample of changing behaviour. Additionally, AOL's day of missing data [2] may to some degree harm QAC effectiveness following the affected period. Furthermore, there may be underlying demographic differences between users of the two search engines that lead to changes in query distributions, for instance, MSN and Sogou users may be more responsive to ongoing events. Finally, bot spamming activity in AOL may degrade QAC performance for real users by distorting query popularity distributions. Regardless of AOL's possible issues, we still consider AOL results indicative if the behaviour is echoed by Sogou and MSN results.

Effectiveness converges for 5-character prefixes, probably because QAC is simply constrained by how much past evidence is available, and a 5-character prefix hugely narrows possible completion suggestions in any case.

RQ2: Can relying only on recent query popularity evidence improve QAC effectiveness? In Section 5.2 we found that QAC effectiveness can be improved by using a recent sliding window period of query log evidence to rank completion suggestions. Although MRR gains are relatively small (1.1-3.6% above the soft baseline), they are consistent for 2-character prefixes over all query logs.

Although performance improvements for each prefix length and sliding window period are different for each query log, there is a clear overall relationship between prefix length and optimal slid-

ing window period emerging from the results. QAC for shorter prefixes performs optimally with a shorter sliding window of evidence, and conversely, QAC for longer prefixes performs best with a longer sliding window of evidence. However, for prefixes of 3 or more characters, the 14-28 day sliding window period upper limit we experiment with is not enough to begin observing notable improvements consistent with this trend (we cannot reliably increase the sliding window period for our short-term experimental query logs). We would expect experiments on longer-term query logs to begin seeing improvements based on this trend.

Even though relying on recent evidence (i.e. using a short sliding window) increases the time-sensitivity of QAC, it is apparent that it can also harm the robustness of QAC, reducing performance overall. Less common prefixes (e.g. for rarer queries) and longer prefixes (e.g. 4-5 characters) rely on a longer sliding window period to include the evidence necessary to rank them effectively, since they are less likely to have been used recently. In these cases, imposing a sliding window period is an inherently flawed approach as it excludes potentially valuable past query log evidence.

Regardless of language, time is a common characteristic that affects all QAC scenarios. QAC effectiveness is not stationary over time (evidenced by Figure 3), however, the impact of time is very different for each query log, suggesting that each query log scenario has differing temporal characteristics.

RQ3: Can an optimal trade-off between recency and robustness be achieved for QAC? In Section 5.3 we presented the results of our proposed LNQ approach, which addresses the flaw of imposing a sliding window period of evidence for all prefixes. Rather than imposing a strict past evidence time period, LNQ tracks the past N queries observed with each prefix and ranks completion suggestions by their distribution. The N parameter allows the model to be adjusted for robustness (greater N) or recency (lower N), while accommodating the varying popularity of different prefixes. n ensures no single query can flood the model if it becomes extremely popular, or abused (i.e. spammed).

LNQ results observed for MSN and Sogou are especially encouraging, with up to +3.9% and +4.9% QAC performance increases over the hard baseline for many N parameters, demonstrating that an optimal trade-off between recency and robustness can be reached. Improvement is marginal for AOL, but exploring further N parameters may yield better results. What is in essence a very straight-forward and practical technique is capable of making considerable gains over the hard baseline.

Different prefixes benefit from different LNQ N parameters at different times. This is demonstrated by O-LNQ in Table 6, where online learning of the optimal LNQ N parameter per prefix for LNQ leads to the best performance we observe overall for 2-character prefixes in MSN and Sogou: +4.1% and + 5.4% over the hard baseline, and correspondingly a +7.2% and +9.2% QAC effectiveness improvement over the soft baseline.

RQ4: Can short-range query popularity prediction (based on recent trends) improve QAC effectiveness? Given the effectiveness of the stand-alone LNQ approach, we adapted it for short-range query popularity prediction based on recent trends. In Section 5.4 we presented results of the PNQ approach which uses an incrementally-learnt regression model to predict current query popularity for completion suggestion ranking.

In almost all cases, PNQ was only able to marginally improve the best performing LNQ approach for each query log (i.e. by a fraction of a percent). Despite the relatively high prediction accuracy of the query count regression model (especially for R^2, which is most pertinent given the rank-ordered nature of QAC), PNQ only offers small gains over LNQ. Future work will investigate more elaborate incrementally-learnt regression models to improve PNQ performance through increased temporal sensitivity.

7. CONCLUSION

In this work we considered the trade-off between recent (i.e. time-sensitive) and robust query auto-completion (QAC). Our objective was to develop effective QAC approaches which can provide both consistently popular, and recently popular query completion suggestions when necessary.

First we outlined two existing common approaches to QAC, which are based on the most popular queries observed previously, and in recent periods of time (i.e. MLE-ALL and MLE-W). We proposed a new QAC approach based on tracking the last N queries with a given prefix (i.e. LNQ). We extended this approach to anticipate upcoming query popularity changes using query count prediction based on an incrementally-learnt short-range linear regression model (i.e. PNQ). Finally, we proposed an online parameter learning meta-approach to determine the optimal parameters of the aforementioned QAC approaches in a typical online QAC setting.

Using three real query logs: AOL, MSN (both English) and Sogou (Chinese) – we simulate a realistic real-time QAC system and experiment with each QAC approach for query prefix lengths of 2 to 5 characters over tens of millions of queries. The diverse temporal, language and demographic characteristics of each query log allow us to compare approach performance in differing real-world scenarios. As all three query log datasets are publicly available, all results presented in this paper are reproducible. Through extensive empirical investigation we study the effectiveness of each approach with various parameter settings.

We find that using all past query log evidence to rank completion suggestions (i.e. MLE-ALL, our soft baseline) leads to underperforming QAC. However, using a sliding window period of past evidence (MLE-W, our hard baseline) can improve QAC performance up to 3.6% above the soft baseline.

Tracking the last N queries observed for a prefix to rank completion suggestions (i.e. LNQ) produces strong performance improvements in most cases, especially when the trade-off between robustness and recency, controlled by N, is optimal. Notably, QAC for MSN and Sogou is improved by +7% and +8.7%, respectively, for prefixes of 2 characters over the MLE-ALL soft baseline. LNQ is a highly practical and efficient approach which can be easily trained on- or off-line for effective recent and robust QAC performance. Notably, online learning to select the optimal LNQ N parameter for each prefix over time leads to the best QAC performance we observe for MSN and Sogou (i.e. +7.2% and +9.2% over the soft baseline for prefixes of 2 characters). This represents reproducible state-of-the-art QAC performance for these query logs.

Employing a relatively high-accuracy recent-trend linear regression model to rank completion suggestions based on short-range predicted query counts (i.e. PNQ) offers little improvement over an appropriately selected LNQ model.

Overall, building an effective QAC system requires careful selection of approach, parameters and query popularity predictive model to suit the intrinsic temporal, language and demographic characteristics of the scenario; we found no combination that is assured to be effective in all situations. We hope the insights and patterns observed for each approach in this work will inform later QAC system research and development.

Future Work. Further work will investigate improving short-range query popularity prediction approaches, as well as methods to incorporate long-term temporal query popularity modelling [26]. Experiments on longer, more recent query logs will further validate the findings presented in this work.

8. REFERENCES

[1] *Proceedings of the 2009 workshop on Web Search Click Data*. ACM, 2009.

[2] E. Adar, D. S. Weld, B. N. Bershad, and S. S. Gribble. Why we search: visualizing and predicting user behavior. In *WWW '07*, pages 161–170. ACM, 2007.

[3] Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *WWW '11*, pages 107–116, 2011.

[4] H. Bast and M. Celikik. Fast construction of the hyb index. *ACM Trans. Inf. Syst.*, 29(3):16:1–16:33, July 2011.

[5] S. M. Beitzel, E. C. Jensen, A. Chowdhury, O. Frieder, and D. Grossman. Temporal analysis of a very large topically categorized web query log. *J. Am. Soc. Inf. Sci. Technol.*, 58(2):166–178, Jan. 2007.

[6] S. Bhatia, D. Majumdar, and P. Mitra. Query suggestions in the absence of query logs. In *SIGIR '11*, pages 795–804. ACM, 2011.

[7] L. D. Catledge and J. E. Pitkow. Characterizing browsing strategies in the world-wide web. In *WWW '95*, pages 1065–1073. Elsevier North-Holland, Inc., 1995.

[8] M. Chau, Y. Lu, X. Fang, and C. C. Yang. Characteristics of character usage in chinese web searching. *Information Processing & Management*, 45(1):115–130, 2009.

[9] S. Chaudhuri and R. Kaushik. Extending autocompletion to tolerate errors. In *SIGMOD '09*, pages 707–718. ACM, 2009.

[10] W. Croft, D. Metzler, and T. Strohmann. *Search Engines: Information Retrieval in Practice*. Pearson, London, England, 2010.

[11] J. Da. A corpus-based study of character and bigram frequencies in chinese e-texts and its implications for chinese language instruction. In *4th International Conference on New Technologies in Teaching and Learning Chinese*, pages 501–511, Beijing, China, 2004. The Tsinghua University Press.

[12] H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *WWW '11*, pages 117–126. ACM, 2011.

[13] P. Ellis. *The Essential Guide to Effect Sizes: Statistical Power, Meta-Analysis, and the Interpretation of Research Results*. Cambridge University Press, 2010.

[14] P. Fafalios, I. Kitsos, and Y. Tzitzikas. Scalable, flexible and generic instant overview search. In *WWW '12*, pages 333–336. ACM, 2012.

[15] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, In Press, 2014.

[16] N. Golbandi, L. Katzir, Y. Koren, and R. Lempel. Expediting search trend detection via prediction of query counts. In *WSDM '13*, pages 295–304, 2013.

[17] S. R. Kairam, M. R. Morris, J. Teevan, D. J. Liebling, and S. T. Dumais. Towards supporting search over trending events with social media. In *ICWSM '13*, 2013.

[18] D. Kastrinakis and Y. Tzitzikas. Advancing search query autocompletion services with more and better suggestions. In *ICWE '10*, pages 35–49, Berlin, Heidelberg, 2010. Springer-Verlag.

[19] D. E. Knuth. *The art of computer programming, volume 1 (3rd ed.): fundamental algorithms*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.

[20] A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics. In *WSDM '11*, pages 167–176, New York, NY, USA, 2011. ACM.

[21] Y. Moshfeghi and J. M. Jose. On cognition, emotion, and interaction aspects of search tasks with different search intentions. In *WWW '13*, pages 931–942, ACM, 2013.

[22] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *InfoScale '06*. ACM, 2006.

[23] C. Sengstock and M. Gertz. Conquer: a system for efficient context-aware query suggestions. In *WWW '11*, pages 265–268. ACM, 2011.

[24] M. Shokouhi. Detecting seasonal queries by time-series analysis. In *SIGIR '11*, pages 1171–1172, 2011.

[25] M. Shokouhi. Learning to personalize query auto-completion. In *SIGIR '13*, pages 103–112. ACM, 2013.

[26] M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *SIGIR '12*, pages 601–610, 2012.

[27] A. Strizhevskaya, A. Baytin, I. Galinskaya, and P. Serdyukov. Actualization of query suggestions using query logs. In *WWW '12*, pages 611–612, 2012.

[28] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts. Information re-retrieval: repeat queries in yahoo's logs. In *SIGIR '07*, pages 151–158, New York, NY, USA, 2007. ACM.

[29] S. Whiting, J. McMinn, and J. Jose. Exploring real-time temporal query auto-completion. In *DIR Workshop '13*, pages 12–15.

[30] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML '04'*, pages 116–124. ACM, 2004.

[31] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(1):27–39, 2014.