Exploiting Twitter for Spiking Query Classification

Mitsuo Yoshida[1] ⋆ and Yuki Arase[2]

[1] Graduate School of Systems and Information Engineering, University of Tsukuba,
1-1-1 Tennodai, Tsukuba, Ibaraki, Japan
ceekz@mibel.cs.tsukuba.ac.jp
[2] Microsoft Research Asia,
Building 2, No.5 Dan Ling Street, Haidian District, Beijing, P.R. China
yukiar@microsoft.com

Abstract. We propose a method for classifying queries whose frequency spikes in a search engine into their topical categories such as celebrities and sports. Unlike previous methods using Web search results and query logs that take a certain period of time to follow spiking queries, we exploit Twitter to *timely* classify spiking queries by focusing on its massive amount of super-fresh content. The proposed method leverages unique information in Twitter—not only tweets but also users and hashtags. We integrate such heterogeneous information in a graph and classify queries using a graph-based semi-supervised classification method. We design an experiment to replicate a situation when queries spike. The results indicate that the proposed method functions effectively and also demonstrate that accuracy improves by combining the heterogeneous information in Twitter.

Keywords: Query Classification, Spiking Query, Twitter

1 Introduction

The frequency of a Web search query naturally reflects the degree of people's interest in it. Therefore, queries that suddenly spike in a search engine can be regarded as gaining more attention from people. We propose a method for *timely* classifying such spiking queries into their topical categories, for example, celebrities and sports. Having categories of spiking queries is useful for search engines in various ways. They may help to immediately improve the relevance of search results and enable to trigger an appropriate vertical search when a query becomes popular. They also benefit search advertisers in presenting relevant advertisements at the time when more people are interested in related topics. Although many query classification methods have been proposed (*e.g.*, [11]), little attention has been paid to spiking queries.

The biggest challenge in spiking query classification is a lack of resources for characterizing them. Although previous methods of query classification have used Web search results and query logs[3], they are not always available in a timely fashion for spiking queries due to the queries' sudden emergence. Broder *et al*. [4] show that about 5% of queries in their experimental dataset are too recent to obtain search results for, and

⋆ This project was conducted while the first author was visiting Microsoft Research Asia.

[3] In this paper, *query logs* include all information associated with a query, even click-through.

that affects the classification accuracy. Apart from this challenge in resource unavailability, the expensive cost of manually labeling queries to train a classifier is a common challenge in query classification tasks, regardless of whether focusing on spiking or general queries.

To tackle these challenges, we propose a method for timely classifying spiking queries by exploiting Twitter[4], which provides a huge amount of super-fresh content on a broad range of topics in real-time. We leverage unique information that Twitter provides: tweets, users, and hashtags. We use tweets that contain spiking queries to evaluate the similarity between queries, assuming that similar queries belong to the same category. In addition, we use information on users and hashtags and evaluate their correlation with queries. We assume that if a Twitter user follows, or in other words, "belongs to" a category, queries that appear in his/her tweets also belong to that category. Likewise, if a query belongs to a category, users who posted tweets containing the query also belong to the same category. The same relationship is also applicable to hashtags. If a hashtag belongs to a category, queries that appear in tweets with the same hashtag also belong to that category, and vice versa. We use a graph-model to integrate such heterogeneous information and adapt a graph-based semi-supervised learning method for classification that requires a smaller amount of training data.

The contribution of our method is twofold. First, we propose a novel method for timely classifying spiking queries by capitalizing on its correlation with Twitter. Second, we leverage unique information that Twitter provides to improve the classification accuracy. We consider not only tweets, but also users and hashtags, to characterize spiking queries and we combine these into a graph-model.

We carefully design an experiment to evaluate the proposed method for classifying spiking queries on the day the queries become popular in a search engine. The results show that the method is effective and that accuracy improves by combining the query similarity and correlation with users/hashtags.

2 Related Work

Many studies have investigated query classification of their topical categories based on Web search results and query logs. They use supervised, semi-supervised, or unsupervised classification methods.

In supervised-learning based approaches, the KDD-Cup 2005 competition featured query classification, where attendees used search result pages and their attributes such as titles and snippets, as well as search engine directories to extract features of queries [11]. Baeza-Yates *et al*. [2] use click-through data to expand an input query and generate a feature based on terms that appear in accessed Web pages. Broder *et al*. [4] focus on rare queries with low frequency that are therefore difficult to classify. The difference between spiking queries and rare queries is the number of relevant Web pages. Broder *et al*. assume that there are still a sufficient number of Web pages to characterize the rare queries, even though their frequency in a search engine is small. Their method classifies search result pages of an input rare query and Web pages linked from them instead of

[4] `http://twitter.com/`

classifying the query itself. Nevertheless, we cannot always assume such resources are available for spiking queries when they spike in a search engine.

These methods use supervised-learning methods and thus have a drawback in that they require a large amount of training data. To relax this requirement, researchers also use semi-supervised or unsupervised learning methods. Shen *et al*. [13] propose building a fine-grained intermediate classifier through which an input query is first classified into intermediate categories based on maximum likelihood estimation, and then the intermediate categories are further classified into target categories with a coarse structure. Beitzel *et al*. [3] generate rules to classify queries based on linguistic knowledge combined with a classifier trained by a supervised-learning method. Xiao *et al*. [10] conduct binary classification that decides whether a query has a predetermined intent, such as job or product intents, using click-through data. Diemert and Vandelle [5] construct a concept graph using their search result pages and query logs, in which they inject target categories for classification. They expand an input query with its search result pages and extract salient categories by matching the expanded query and concepts by random-walk in the graph. Hu *et al*. [7] construct such a concept graph using Wikipedia.

These previous studies depend on relevant Web pages and query logs and they are therefore not applicable for timely classifying spiking queries for which these resources are not always available.

Another stream of related work exploits Twitter for extrinsic tasks such as detecting earthquakes [12], identifying and ranking URLs of trendy Web pages [6], and determining high-quality content from a QA portal [1]. These studies have different goals; however, they show that Twitter is a valuable resource that produces super-fresh content and reflects the trends of the general public. These studies also demonstrate the usefulness of various content in Twitter, *i.e*., URLs, users, and their social relationships.

3 Problem Statement

We start by formally defining the spiking query classification problem. In this study, we use Japanese queries and tweets, since Twitter has millions of users in Japan and is therefore popular, and we can obtain a sufficient number of tweets to conduct query classification. Although we use Japanese data, our approach is language-independent and is easily applicable to other languages.

3.1 Definition of Spiking Query

We first define a spiking query as a query whose frequency in a search engine spikes; a spike occurs when there is a massive increase followed by a corresponding decrease in the query's frequency [9]. We regard queries spiking once or multiple times as spiking queries and aim to classify them when they become popular. In this study, we choose queries showing spikes based on their frequency history from all queries that we obtain through a toolbar (Bing Bar[5]) installed on Windows Internet Explorer[6]. Our method can handle queries with either single-term or multiple-term structures.

[5] `http://toolbar.discoverbing.com/`

[6] `http://windows.microsoft.com/en-US/internet-explorer/products/ie/home`

We observe trends in spiking queries on Twitter in comparison with their trends on news pages to investigate whether tweets containing the spiking queries are available in a timely fashion. News pages serve as a good baseline since they are highly responsive to fresh topics, *i.e.*, spiking queries, since they are intended to convey timely information to people. We obtain news data by crawling the main content of Web pages from a news portal (Ceek.jp News[7]) covering local and nationwide news agencies throughout Japan.

A typical example is in Fig. 1. A trend in the spiking query `karelog` is evident from August 24 to September 15, 2011. `karelog` is the name of a mobile application released on August 28 in Japan. The frequencies of the query and tweets are shown in a logarithmic scale (for intuitive representation, we add 1 to all frequencies to avoid having a missing value when the raw frequency is 0), while the frequencies of the news pages are raw values. It is clear that the query created a buzz on Twitter with more than 13K tweets simultaneously on the day it spiked in the search engine for the first time. On the contrary, there was only one news page that featured this query on the same day. The spike in news pages did not occur until two weeks later, and on that day, the second spike occurred on Twitter. This shows that Twitter users are amazingly reactive to trendy topics. These observations reveal that Twitter enables us to classify spiking queries in a timely manner.

3.2 Problem Definition of Spiking Query Classification with Twitter

When queries $Q = \{q_1, \ldots, q_n\}$ that are input into a search engine spike, we aim to timely classify a spiking query q_i into a predetermined category in $C = \{c_1, \ldots, c_m\}$ by leveraging unique information that Twitter provides, *i.e.*, tweets, users, and hashtags.

In this study, we assume that one query belongs to one category, because when a query spikes, it is generally triggered by a specific topic such as the release of a new product or new film, which makes the corresponding category dominant. Therefore, we decide the most likely category to be the query's category.

4 Proposed Method

The input used with our method consists of spiking queries, predefined categories, and prior labels of queries $\hat{Y} = \{y'_1, \ldots, y'_n | y'_i \in C \cup \nu\}$ where $y'_j \in C$ if the query q_j is labeled; otherwise, y'_j has a default label ν. Prior labels represent categories of labeled queries and serve as training data in semi-supervised classification. The method finally outputs categories $Y = \{y_1, \ldots, y_n | y_i \in C\}$ assigned to the input queries.

Our first step is to construct a graph. We extract users $U_Q = \{u_1, \ldots, u_l\}$ who posted tweets containing the queries Q, and hashtags $H_Q = \{h_1, \ldots, h_k\}$ that are assigned to tweets containing the queries Q. We match a query and tweet by substring matching. With these queries, users, and hashtags, we construct the query-Twitter graph $G = \{V, E, W\}$. Here, V consists of n query nodes V_q, l user nodes V_u, and k hashtag nodes V_h ($N = n + l + k$), in which we generate the query nodes V_q using the input

[7] `http://news.ceek.jp/` (This news portal is provided by the first author.)

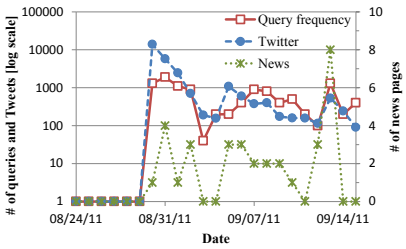

Fig. 1. Trend of spiking query `karelog` in search engine, Twitter, and news pages.

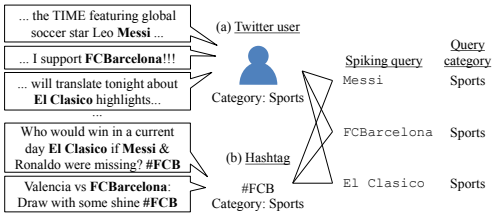

Fig. 2. Correlation between query and user (a) and query and hashtag (b); categories of query and user/hashtag affect each other.

queries Q. The variable E represents the edges between nodes, and W is an $N \times N$ edge weight matrix. If the $edge(v_i, v_j) \notin E$ between nodes v_i and v_j, then $W_{ij} = 0$. With this query-Twitter graph G, we conduct classification using a graph-based semi-supervised classification method, in which we propagate categories of labeled queries $\hat{Y}$ to the entire graph.

4.1 Graph Construction

Correlation between Query and User Twitter provides information on who posts tweets, which enables to characterize queries from a novel aspect, *i.e.*, correlation between a query and a user. Fig. 2 (a) illustrates this correlation; we can infer that the category of the user is "sports" if he/she posts many tweets containing queries in a sports category. On the contrary, we can infer that the category of query `Messi` is probably sports if users who belong to the sports category frequently post tweets containing that query.

To employ the correlation, we extract user nodes V_u from Twitter data. For each query q_i, we extract a set of tweets T_i containing the query q_i. Based on T_i, we find a set of users U_i who posted tweets containing the query q_i. Finally, we obtain a unique set of users U_Q from $U_{all} = \{U_1, \ldots, U_n\}$ and use them as user nodes V_u.

We join nodes V_q and V_u by edges E and compute the edge weight matrix W. To introduce the correlation between a query q and user u, we set an edge with weight W_{qu} between them if the user has posted tweets containing the query. Naturally, the more the user posts tweets containing the query, the more likely it is that the user and query belong to the same category. Therefore, we define the edge weight to represent the strength of their correlation. When considering this from the user side, the correlation $\psi(u \to q)$ represents the probability that the user posts tweets containing the query. On the other hand, when considering this from the query side, the correlation $\psi(q \to u)$ represents the probability of the query being tweeted by this specific user. These are computed as

$$\psi(u \to q) = \frac{count(u,q)}{\sum_{q' \in Q} count(u,q')}, \psi(q \to u) = \frac{count(u,q)}{\sum_{u' \in U_Q} count(u',q)},$$

where $count(u,q)$ represents the number of tweets that are posted by the user u and that contain the query q. We then compute the edge weight as $W_{qu} = (\psi(u \to q) + \psi(q \to u))/2$, which represents the strength of the correlation between the user and query.

Correlation between Query and Hashtag A unique functionality in Twitter is the hashtag. The hashtag starts with the indicator symbol "#," as in "#FCB," and is occasionally assigned to mark the topic of a tweet. Users share the same hashtag and freely assign it to their tweets, which enables tweets about the same topic to be aggregated. In our data, about 10% of tweets contain hashtags. The hashtag is another useful resource for query classification because it is highly likely that queries contained in tweets with the same hashtag belong to the same category, as Fig. 2(b) shows. This relationship is identical to the correlation between a query and a user.

For each query q_i and tweets T_i containing the query, we find a set of hashtags H_i assigned to T_i. We then obtain a unique set of hashtags H_Q from $H_{all} = \{H_1, \ldots, H_n\}$ and use them as hashtag nodes V_h. Similar to the edge between a query and user, we set an edge with weight W_{qh} between a query q and hashtag h based on their correlation:

$$\psi'(h \to q) = \frac{count'(h,q)}{\sum_{q' \in Q} count'(h,q')}, \psi'(q \to h) = \frac{count'(h,q)}{\sum_{h' \in H_Q} count'(h',q)},$$

where $count'(h,q)$ represents the number of tweets that contain both the hashtag h and the query q. Finally, the edge weight is computed as $W_{qh} = (\psi'(h \to q) + \psi'(q \to h))/2$.

Context Similarity between Queries In addition to the correlation between a query and user/hashtag, we consider the similarity between queries using tweets in which the queries appear. This common approach has been used in previous studies, as discuss in Sec. 2. The underlying assumption is that queries of the same category are tweeted in a common context; for example, queries belonging to the sports category would be tweeted with the terms "football" and "tournament." We set an edge between query q and q' with weight $W_{qq'}$ based on their similarity $sim(q,q')$.

We generate a vector to represent q using the bag-of-words model with terms extracted from tweets $T_Q = \{T_1, \ldots, T_n\}$ that contain the queries of the classification target. We use tf-idf to compute an element of the vector. Then we compute the similarity $sim(q,q')$ of q and q' using the cosine similarity; this is a conventional method to measure the similarity of two vectors. To prune an edge between dissimilar queries, we introduce a threshold ρ. The edge is set only if $sim(q,q') > \rho$; otherwise, the edge weight is $W_{qq'} = 0$.

When the categories the queries' belonging to are semantically related, the queries may have a similar context even though they belong to different categories. For example, the name of an actress belonging to the celebrities category may appear together with the title of a film starring the actress, even though the title belongs to the movie category. In such a case, queries have similarity with various categories and may confuse the classifier. Therefore, we need to introduce a normalizing factor in the similarity measure. When we focus on q, the similarity is updated as

$$sim'(q \to q') = \frac{sim(q,q')}{\sum_{q^* \in \{q^* | edge(q,q^*) \in E\}} sim(q,q^*)}.$$

This normalization factor makes the similarity directional, and thus, the final edge weight is computed by considering the similarity both from q to q' and from q' to q: $W_{qq'} = (sim'(q \to q') + sim'(q' \to q))/2$.

Balance Edge Weights We define edge weights between a query and user/hashtag based on their correlation, and an edge weight between queries based on their context similarity. Since these correlations and context similarities are based on different statistical evidence, we may not handle their weights equally. Therefore, we balance their influence by introducing a weighting parameter[8] α.

$$W'_{qu} = \alpha W_{qu}, W'_{qh} = \alpha W_{qh}, W'_{qq'} = (1-\alpha) W_{qq'}. \quad (1)$$

All of these weights are normalized to range from 0 to 1 as describe previously. The parameter α ranges from $0 \leq \alpha \leq 1$. We evaluate the effect of the parameters in our method, *i.e.*, ρ and α, in Sec. 5.4.

4.2 Graph-based Semi-supervised Classification

Now that we have the query-Twitter graph, we conduct classification using the graph. We cast the classification problem as a semi-supervised graph labeling problem to achieve a cost-effective classifier. The query-Twitter graph G has N nodes consisting of n query nodes, l user nodes, and k hashtag nodes ($n < k \ll l$). Under the framework of the semi-supervised approach, we assign category labels C to the small number of n_0 query nodes V_{q0}. Other n_1 query nodes are unlabeled ($n = n_0 + n_1$). Our aim is to propagate the labels to unlabeled n_1 query nodes via user/hashtag nodes. In the label propagation, we want nodes connected by a highly weighted edge to have the same label. When a node v_i receives a propagated label y_i and its neighboring node $v_j \in Neighbor(v_i)$ receives a propagated label y_j, our objective function is

$$E(C) = \sum_{i,j} W_{ij}(y_i - y_j)^2 \ \text{ s.t. } \ y_k = \tilde{c},$$

where y_k is the label of a labeled node $v_k \in V_{q0}$ that is assigned a category $\tilde{c} \in C$. The final label assignment Y is obtained by minimizing the objective function

$$Y = \underset{Y}{\operatorname{argmin}} E(C).$$

Solutions for this optimization problem have been proposed as graph-based semi-supervised classification algorithms [15, 14]. We adapt the modified adsorption algorithm [14] because it is suitable for handling a highly connected graph like ours and prevents densely connected nodes from excessively affecting the lesser connected nodes. In addition, it achieves state-of-the-art performance. We use the implementation distributed by the authors[9].

5 Evaluation

We evaluate the classification accuracy of the proposed method with a realistic setting that replicates the situation when a query spikes.

[8] Since the principle of the correlations regarding users and hashtags is the same, we use the same weighting parameter for simplicity.

[9] `https://github.com/parthatalukdar/junto` (Junto v1.2.2)

5.1 Dataset

For this evaluation, we need queries with their frequency history. We use a 1% sample of queries in Japanese that ware input to Bing Bar during three months; from July 1 to September 30, 2011. Our query data consist of a query string, issued date, and frequency on the day. We extract spiking queries and discard those whose frequency is less than 10 since the magnitude of spikes is too small to determine whether the spikes are meaningful or just by chance. The result produced 5,721 unique queries, which we then labeled categories to.

For labeling, we use an on-line dictionary service called Hatena Keyword[10], where only approved users can edit entries and assign categories to them. This is a popular service in Japan and is accessed by more than 5 million people per month. An advantage of using this service is that it has a simple and clear category structure, unlike Wikipedia, which has complex and unstructured categories. As a result, we assign 17 categories to 2,923 queries, as shown in Fig. 3. We sample queries and manually examine the assigned categories, and confirm that their quality is reliable. In this evaluation, we only label queries, not users or hashtags. We plan to label them in a future study.

The Twitter data we use consist of tweets in Japanese posted during the same period. In total, we collect 251M tweets through Twitter's official API[11]. Of these, we use about 45M tweets that contain the spiking queries necessary for the experiments. We carry out preprocessing using MeCab[12] [8], with which we segment Japanese sentences into words in order to compute the similarity (*i.e.*, edge weight) between queries.

5.2 Comparison Method

To evaluate our method in comparison to another method, we need a method that uses a semi-supervised learning based classification approach. In addition, to evaluate accuracy on days when a query spikes, the comparison method should use resources with date information for feature extraction. The latter requirement makes it difficult to compare our method with those used in the previous studies we discuss in Sec. 2 because they depend on resources that do not allow us to replicate their situation in the past, for example, search engine results.

Therefore, we decide to compare our method with a method using a graph constructed with news pages that contain published date information instead of tweets. Since news pages are one of the most responsive resources to fresh topics, this method serves as a baseline to evaluate how timely the proposed method can classify spiking queries. We also compare our method with subgraphs of the query-Twitter graph to analyze the effect of the correlation and context similarity as follows.

1. NewsGraph (Baseline): We use the news archive described in Sec. 3.1. During the three-month experiment period, we collect 402K news pages consisting of 103 sites in total. We construct a graph consisting of query nodes and compute an edge weight based on their context similarity when they appear in news content.

[10] `http://d.hatena.ne.jp/keyword/`

[11] `https://dev.twitter.com/docs/streaming-apis/streams/public`

[12] `http://code.google.com/p/mecab/` (MeCab v0.98 for MS-Windows)

2. QueryGraph: To evaluate the effect of similarity between queries only, we construct a graph consisting of query nodes. This is a sub-graph of the query-Twitter graph, *i.e.*, $G_q = \{V_q, E_q, W_q\}$ where V_q represents query nodes, E_q represents edges among query nodes, and W_q is the edge weight matrix among query nodes.
3. UserGraph: To evaluate the effect of correlation between queries and users/hashtags only, we construct a query-Twitter graph with only edges between query and user/hashtag nodes. This is a sub-graph $G_u = \{V, E_u, W_u\}$, where V includes all nodes, E_u represents edges between query and user/hashtag nodes, and W_u is the edge weight matrix among query and user/hashtag nodes, *i.e.*, $W_{ij} = 0$ if $v_i, v_j \in V_q$.
4. QueryTwitterGraph (Proposed method): This is the proposed query-Twitter graph G that has full features, and is the superimposed graph of QueryGraph and UserGraph.

5.3 Procedure

To replicate the situation when a query spikes in a search engine, we segment queries, tweets, and news pages according to their date information. We set a sliding window of size d-day and use queries spiking in the window to evaluate classification accuracy. As Fig. 4 shows, we use queries spiking in the window to construct a graph with tweets or news pages published in the same window, and then label the queries spiking in the first $d-1$ days with their categories and then propagate their labels to queries spiking *on* the d-th day (we call the d-th day a test day). For queries that spike multiple times, we regard the day when each query's frequency is maximum as its spike. In this manner, we can evaluate the classification accuracy on the day a query spikes.

We set $d = 28$ ($= 4$ weeks) to avoid any differences in frequency between queries and tweets due to the effect of the day of week. We slide the window from the beginning of the experimental period to the end in one day intervals, *i.e.*, the first test day is July 28. In this setting, we have 65 windows (with test days from July 28 to September 30). Of these, we use the first 28 windows to tune the parameters for graph construction as the development dataset, and the remaining 37 windows to evaluate the accuracy. Overall, each day has about 31.7 queries on average with a standard deviation of 14.3, and each test day has 30.1 queries on average with a standard deviation of 17.7.

We follow the standard evaluation metrics for query classification that were used in the KDD-Cup 2005 competition [11], namely, precision (P) and recall (R), which are defined as:

$$P = \frac{\sum_i \textit{\# of queries are correctly labeled as } c_i}{\sum_i \textit{\# of queries are labeled as } c_i},$$
$$R = \frac{\sum_i \textit{\# of queries are correctly labeled as } c_i}{\sum_i \textit{\# of queries whose category is } c_i},$$

as well as F-score (F1): $F1 = 2PR/(P+R)$.

5.4 Results and Discussion

We first describe the overall classification accuracy, and then show how the different parameters affect the evaluation metrics. Finally, we compare our method with a graph constructed using click-through data.

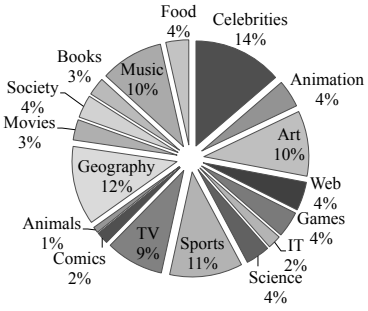

Fig. 3. Distribution of query categories

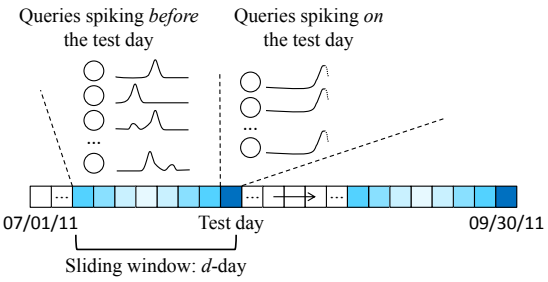

Fig. 4. Experimental procedure

Table 1. Classification accuracy

Method	Precision	Recall	F-score
NewsGraph	45.5	30.3	36.4
QueryGraph	46.1	42.1	44.0
UserGraph	48.9	45.7	47.3
QueryTwitterGraph	**50.9**	**50.1**	**50.5**

Table 2. Comparison with click-through data

Method	Precision	Recall	F-score
ClickGraph	35.4	32.1	33.7
QueryGraph	38.5	25.8	30.3
UserGraph	23.3	17.6	20.1
QueryTwitterGraph	36.8	33.3	35.0

Classification Accuracy We set the parameters on graph construction (ρ and α) to have the best F-score, as described in the next paragraph. Specifically, $\rho = 0.4$ for NewsGraph, $\rho = 0.2$ for QueryGraph, and $\rho = 0.2$ and $\alpha = 0.8$ for QueryTwitterGraph.

Table 1 lists the precision, recall, and F-score of the proposed and comparison methods. The proposed method achieves the best accuracy, with a precision value of 50.9%, recall of 50.1%, and F-score of 50.5%. We conduct a sign test and confirm that QueryTwitterGraph has significantly better classification power than NewsGraph, QueryGraph, and UserGraph ($p \ll 0.01$). In fact, it has about a 20% higher recall value than NewsGraph. This result shows that Twitter is useful not only for timely classification but also for widening the coverage of the classifier. Another surprising result is that our method achieves about 5% higher precision than NewsGraph, even though tweets are noisier than professionally edited news text. The results in Table 1 indicate that this is due to the effect of user/hashtag nodes; the fact that UserGraph achieves 3.4% higher precision than NewsGraph demonstrates it.

When comparing QueryGraph and UserGraph, it is impressive that UserGraph achieves about 3% higher precision and about 4% higher recall than QueryGraph ($p = 0.018$ by a sign test). Recall that UserGraph does not depend on any textual resources; it is based purely on the correlation between a query and user/hashtag in terms of their belonging category. This shows that the correlation between a query and user/hashtag is effective evidence of classification. The even better accuracy of QueryTwitterGraph shows that these two graphs complement each other to improve the accuracy.

Effect of Parameters Next, we evaluate the effect of the parameters in our method. We start with parameter ρ, which controls the number of edges between query nodes. Fig. 5 plots the precision, recall, and F-score on QueryGraph when the value of the parameter ρ is changed in the development dataset. The best F-score (41.6%) is achieved when $\rho = 0.2$, along with the best recall (39.7%) and the second-best precision (43.7%).

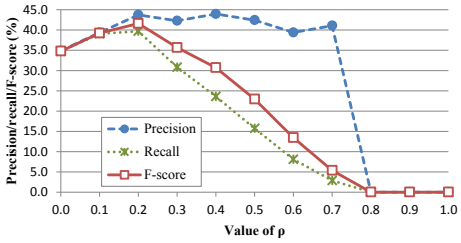

Fig. 5. Effect of parameter ρ (QueryGraph)

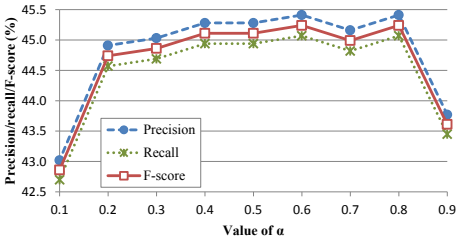

Fig. 6. Effect of parameter α ($\rho = 0.2$)

When we increase the value of ρ, the number of edges decreases. Larger ρ disrupts the propagation of labels, and thus both precision and recall drop.

Parameter α balances the effect of the correlation between a query and user/hashtag and that of context similarity between queries in Eq. (1). When α is larger than 0.5, our method places more edge weight on correlation. Fig. 6 plots the precision, recall, and F-score on QueryTwitterGraph when the value of the parameter α (we use $\rho = 0.2$) is changed in the development dataset. The accuracy improves when the value of α is increased, and the best precision (45.4%), recall (45.1%), and F-score (45.2%) are achieved when $\alpha = 0.8$. This result shows the effectiveness of introducing the correlation between a query and user/hashtag, and its contribution is significant.

Comparison with Click-Through Data Click-through data are a useful resource for characterizing associated queries, since they provide relevant Web pages accessed by people. We obtain click-through data of the experimental queries (four weeks from September 3 to September 30) and use the data to construct a graph, which we then compare with our method. In total, we have 809 queries with click-through data.

We construct a graph (ClickGraph) consisting of query nodes using the click-through data and conduct classification as described in Sec. 4.2. We extract the top-10 most frequently accessed Web pages for each query and compute an edge weight between query nodes based on their context similarity. Due to the limitation in available data, we set the size of the sliding window at two weeks ($d = 14$) to get 14 windows. We use the first 7 windows to tune the parameters and the remaining 7 windows for evaluation. For a fair comparison, we apply the same setting to QueryGraph, UserGraph, and QueryTwitterGraph.

Table 2 lists the precision, recall, and F-score of ClickGraph ($\rho = 0.4$), QueryGraph ($\rho = 0.3$), UserGraph, and QueryTwitterGraph ($\rho = 0.3, \alpha = 0.5$). The results indicate that QueryTwitterGraph achieves accuracy comparable to ClickGraph (no significant difference is detected). This result shows that our approach is promising, as it is comparable to a method that uses a highly useful resource such as click-through.

6 Conclusion and Future Work

We propose a method for timely classifying spiking queries by exploiting Twitter. The proposed method achieves high accuracy by leveraging unique information that Twitter provides, *i.e.*, tweets, users, and hashtags.

In the future, we first plan to extend our method and include social network information in Twitter such as follower-followee relationships and behaviors of retweeting and mentioning. Then, we apply multi-label classification. We also plan to collect a large-scale human annotation to obtain categories for spiking queries and conduct a more detailed evaluation. Moreover, we combine our method with a previous approach that uses search result pages and query logs to further improve the classification accuracy.

Acknowledgments We sincerely thank Mikio Yamamoto, Takashi Inui, Takaaki Tsunoda, Ming Zhou, and Qing Ma for their valuable comments and feedback in this project. This project was supported by JSPS KAKENHI Grant Number 11J01016.

References

1. Agichtein, E., Castillo, C., Donato, D., Gionis, A., Mishne, G.: Finding high-quality content in social media. In: WSDM '08. pp. 183–194 (2008)
2. Baeza-Yates, R., Caldern-Benavides, L., Gonzlez-Caro, C.: The intention behind web queries. In: Crestani, F., Ferragina, P., Sanderson, M. (eds.) SPIRE '06. LNCS, vol. 4209, pp. 98–109. Springer, Heidelberg (2006)
3. Beitzel, S.M., Jensen, E.C., Frieder, O., Lewis, D.D., Chowdhury, A., Kolcz, A.: Improving automatic query classification via semi-supervised learning. In: ICDM '05. pp. 42–49 (2005)
4. Broder, A.Z., Fontoura, M., Gabrilovich, E., Joshi, A., Josifovski, V., Zhang, T.: Robust classification of rare queries using web knowledge. In: SIGIR '07. pp. 231–238 (2007)
5. Diemert, E., Vandelle, G.: Unsupervised query categorization using automatically-built concept graphs. In: WWW '09. pp. 461–461 (2009)
6. Dong, A., Zhang, R., Kolari, P., Bai, J., Diaz, F., Chang, Y., Zheng, Z., Zha, H.: Time is of the essence: improving recency ranking using twitter data. In: WWW '10. pp. 331–340 (2010)
7. Hu, J., Wang, G., Lochovsky, F., tao Sun, J., Chen, Z.: Understanding user's query intent with Wikipedia. In: WWW '09. pp. 471–480 (2009)
8. Kudo, T., Yamamoto, K., Matsumoto, Y.: Applying conditional random fields to Japanese morphological analysis. In: EMNLP '04. pp. 230–237 (2004)
9. Kulkarni, A., Teevan, J., Svore, K.M., Dumais, S.T.: Understanding temporal query dynamics. In: WSDM '11. pp. 167–176 (2011)
10. Li, X., Wang, Y.Y., Acero, A.: Learning query intent from regularized click graphs. In: SIGIR '08. pp. 339–346 (2008)
11. Li, Y., Zheng, Z., Dai, H.K.: KDD CUP-2005 report: facing a great challenge. SIGKDD Explor. Newsl. 7(2), 91–99 (2005)
12. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: WWW '10. pp. 851–860 (2010)
13. Shen, D., Sun, J.T., Yang, Q., Chen, Z.: Building bridges for web query classification. In: SIGIR '06. pp. 131–138 (2006)
14. Talukdar, P., Crammer, K.: New regularized algorithms for transductive learning. In: Buntine, W., Grobelnik, M., Mladenic, D., Shawe-Taylor, J. (eds.) ECML-PKDD '09. LNCS, vol. 5782, pp. 442–457. Springer, Heidelberg (2009)
15. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML '03. pp. 912–919 (2003)