StarrySky: A Practical System to Track Millions of High-Precision Query Intents

Qi Ye
yeqi@sogou-inc.com

Feng Wang
wangfeng@sogou-inc.com

Bo Li
libo202442@sogou-inc.com

Sogou Inc.
Beijing, China

ABSTRACT

Query intent mining is a critical problem in various real-world search applications. In the past few years we have witnessed dramatic advances in the field of query intent mining area. In this paper, we present a practical system | **StarrySky** for identifying and inferring millions of query intents in daily sponsored search with high precision and acceptable coverage. We have already achieved great advantages by deploying this system in Sogou sponsored search engine[1]. The general architecture of **StarrySky** consists of three stages. First, we detect millions of fine-grained query clusters from two years of click logs which can represent different query intents. Second, we refine the qualities of query clusters with a series of well-designed operations, and call the final refined clusters as concepts. Third and foremost, we build a flexible real-time inference algorithm for assigning query intents to the detected concepts with high precision. Beyond the description of the system, we employ several experiments to evaluate its performance and flexibility. Our inference algorithm achieves up to 96% precision and 68% coverage on daily search requests. We believe **StarrySky** is a practical and valuable system for tracking query intents.

Keywords

Query Intent; Search Log Mining; Community Detection; Large-scale Multi-class Query Classification

1. INTRODUCTION

Understanding search query intent is critically important for satisfying users' search needs, especially in tail query understanding, query suggestion, sponsored search and other vertical search applications. In these tasks, search queries might be only partially matched with search results based on terms, and query intents may possibly be misunderstood, which would cause major relevance issues in search applications and hurt user experience. The situation can be highly improved by introducing a well-developed query intent tracking system which can precisely infer the intents of queries.

In this paper, we propose a practical system **StarrySky** for identifying and inferring millions of query intents in daily sponsored search with high precision and acceptable coverage. We use concepts to present di erent query intents found in our system. The concepts are just like stars shining in the sky of query intents, which is where the name **StarrySky** comes from. As traditional clustering methods tend to find coarse-grained clusters which are not easily capture various fine-grained query intents, in our system we first detect millions of fine-grained query intent clusters from two years click logs. Then, we employ a series of post-processing operations to refine the detected clusters and get the highly consistent concepts each of which contains the same intent queries in one concept. Finally, we build an inference algorithm to track the intents of head and tail queries, and assign them to the identified concepts with high precision and acceptable coverage. Although we mainly focus on how to track query intents in sponsored search, our approach can also be commonly used to enhance user experience in other search tasks and recommendation systems.

The main contributions of **StarrySky** are as follows:

- Detecting millions of fine-grained and highly consistent intent concepts in a massive query co-click graph created from 2 years real-world web search click logs.
- Developing an e cient and scalable inference algorithm to track query intents with high precision.

To the best of our knowledge, this is the first published documentation of a real-time deployed system that infers millions of query intents in real-world web search applications with high precision and acceptable recall. For researchers working on web intelligence, people skilled in the art would be able to create a similar system with the given details in this paper. We address the query intent finding problem as a fine-grained query intent detection and inference task, and propose an e ective and practical system to handle it. In the rest of the paper, we first review related work in Section 2. Section 3 describes the overview of our system and methods in detail. Section 4 presents the experiments based on our system. Section 5 concludes the paper and discusses the future directions.

[1]http://www.sogou.com

WWW'16 Companion, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2890588 .

2. RELATED WORK

To identify major intents of queries, various query clustering algorithms have been proposed. Sadikov et al. [14] model user search behavior as a graph by combining document click and session co-occurrence information, and perform multiple random walks on the graph to cluster queries. Hu et al. [9] present an unsupervised method for clustering queries with similar intent based on the clicked URL features. Radlinski et al. [13] use the BGLL community detection algorithm [2] to find query clusters which present query intents in a query graph. However, these clustering methods cannot be applied to tail queries which are not in the clusters, due to the lack of inference methods.

Another related work to intent finding in short text also includes topic modeling such as Latent Dirichlet Allocation (LDA) [1] and other weakly supervised algorithms [19]. However, the traditional topic models are automatically learned by unsupervised algorithms, and there is no guarantee that the hidden topics will be aligned with predefined taxonomy [8]. The topics will change if the models are retrained over a di erent dataset. Guo et al. [7] argue that query similarity should be defined upon query intents, and they use a regularized PLSI topic model to learn the potential intents of queries by using both the words from search snippets and the regularization from query co-clicks. However, recently it has also been found that topic modeling techniques such as LDA and PLSI do not work well with the short text like web search queries [17]. Jiang et al. [10] try to capture the latent relations between search queries via the URL, session and term dimensions to improves the performance of query intent mining. In this paper, as we regard that co-click relations reveal query intents more precisely, we only focus on using click logs to extract the intents.

There are also many proposed algorithms for intent detection based on semantic classification of search queries. Motivated by the needs of search advertising, Broder et al. [3] use a blind feedback technique and present a two-stage classifier to identify 6000 class labels with reasonable accuracy. Simonet [16] presents a framework based on annotated semantic entities for categorizing channels of videos in a thematic taxonomy with high precision and recall. Phan et al. [12] present a framework to build classifiers for short and sparse text by making use of the hidden topics discovered by the LDA algorithm from huge web text data collections. For short snippet sentiment and topic classification tasks, Wang and Manning [18] show that the Naive Bayes classifier actually does better than SVMs, and we adopt the Naive Bayes algorithm in our inference framework.

3. METHODOLOGY

We define the query intent tracking task as the following: given a search query, we infer its intent and try to assign it to a proper pre-detected fine-grained query concept or make a rejection. In this paper, the fine-grained query concepts are found by an unsupervised algorithm and refined by a series of refinement operations.

The main architecture of our idea of query concepts is shown in Fig. 1. In simple terms, we divide the frequent search queries into di erent fine-grained query clusters called concepts to indicate the query intents, and the belonging relations between queries and clusters and their words are shown in plain links. The weighted edges between concepts reveal their relevant relations which are shown in doted ones. After the intent concept graph has been constructed, we build an inference system to assign query intent to them. To make the intent inference process fast, robust and reliable, for the head queries found in the concept detection stage, we directly return their final concept labels. Otherwise, for the unseen tail queries, we build a large-scale inference algorithm based on n-gram features to assign them to appropriate intent concepts. We will discuss the algorithms of our system in details in this Section.

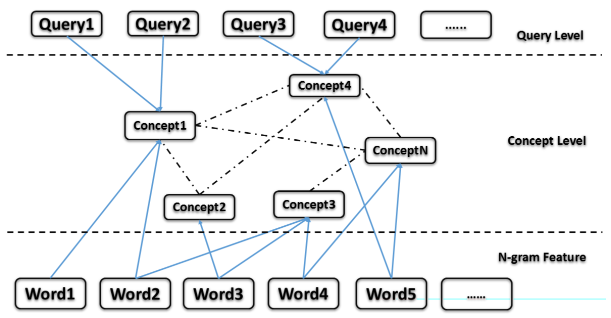


Figure 1: The architecture of StarrySky.

The system is divided into o ine training and online inference stages. In the o ine training stage, **StarrySky** starts from extracting the query co-click relations from web search logs and form a query co-click graph. After that, to capture di erent query intents, we implement a well developed community detection algorithm to find the fine-grained clusters and employ a series of well-designed operations to refine them to construct the final intent concepts. In the online inference stage, we build a large-scale multi-class classifier to infer the intent concept for a given query with two rejection options. Each of these two options refuses to assign labels to uncertain cases by fine tuned strategies.

3.1 Query Cluster Detection

We use web search click logs from the Sogou commercial search engine to build a co-click graph for query intent mining. The basic assumption is that if users click the same web search results, the intents of queries they used should probably be similar. We extract query-URL click information from logs and transform the query-URL bipartite graph into a query co-click graph. It is worth to point out that the co-click query graph is formed with weighted edges based on click co-occurrence counts. To leave reliable co-click relations for the following intent mining algorithms, we set a threshold to remove the low weighted edges. Finally, we get a sparse and trustable query graph with query intent relations. Similar approaches can also be found in other query intent detection algorithms [13, 14]. Without loss of generality, we use a graph clustering algorithm to discover groups of queries with similar intents in the query graph. We implement a well developed community detection algorithm named MMO [20] which has nearly linear time complexity and linear space complexity.

3.2 Concept Refinement

Due to the well known resolution limit issue of modularity [6] and the sparseness of clicks, there might still be some giant inconsistent clusters with multiple intents and small isolated ones with the same intents. To make the concepts more semantically consistent, we need to refine the clusters with considering query similarity measurements.

3.2.1 Quality Evaluation

The first critical problem is that there are some giant clusters which contain many small coherent sub-clusters. The phenomenon indicates that the large cluster may contain multiple intents. To reveal the problem in detected intent clusters, we present a formal quality score $r(c)$ for evaluating the quality of a given intent cluster c, which is defined as:

$$r(c) = \frac{\sum_{q,s \in c, q \neq s} f(q,s)}{|c| \times (|c|-1)}, \tag{1}$$

where $f(q,s) \in [0,1]$ is a pre-defined similarity score of a query q and query s from the same cluster c. We calculate the quality score $r(c)$ in Eq. 1 for each cluster, and then set an empirical threshold $t_s = 0.1$ to find low quality ones. If the relevance score $r(c)$ of a cluster c is less than t_s, i.e., $r(c) < t_s$, the cluster c should be divided into smaller cohesive sub-clusters. After the low quality cluster c_i has been found, the subgraphs G_{c_i} induced by c_i will be split into sub-clusters by the MMO algorithm. After the splitting operation, we recursively check the qualities of sub-clusters and divide them with the same splitting method if necessary.

Another critical problem of the intent clusters is caused by the local nature of the clustering algorithm and the sparse nature of the query co-click information. Thus, clusters extracted from local community detection algorithm may contain many relevant small ones with the same intents. To reveal the small cluster problem, we also define a relevance score $r(c_i, c_j)$ between two clusters c_i and c_j, which is defined as:

$$r(c_i, c_j) = \frac{\sum_{q \in c_i, s \in c_j} f(q,s)}{|c_i| \times |c_j|}, \tag{2}$$

where $f(q,s)$ is also the similarity score of two queries in these clusters.

To combine these small clusters with the same intents and speed up this process, we iteratively calculate $r(c_i, c_j)$ for each cluster pair c_i and c_j, if they are connected in the co-click graph. We merge the cluster pair when its relevance score is larger than a certain threshold t_d, i.e., $r(c_i, c_j) > t_d$, where t_d equals 0.9 in our system. After merging high relevant clusters, we drop small clusters which contain less than 3 queries to make sure that clusters are nontrivial.

After the above refinement steps, we get qualified and consistent query clusters called concepts in our system which can represent di erent fine-grained query intents. Furthermore, we can naturally form a concept graph to reveal the topological relations between di erent concepts. An edge between two concepts c_i and c_j is weighted by $r(c_i, c_j)$ and is kept in the graph if its relevance score is larger than certain empirical threshold $t_r = 0.75$.

3.2.2 Query Similarity

To define the above measures $f(q,s)$ in Eq. 1 and Eq. 2, we need to measure the similarity between queries first. The query similarity task can be formalized as: given a pair of queries q and s, the goal is to learn a probability binary classifier $f(q,s) \rightarrow [0,1]$. $f(q_i, s_i) > f(q_j, s_j)$ indicates that the pair (q_i, s_i) is more relevant comparing with (q_j, s_j). In this part, we present a practical and more precise ensemble method to measure the similarity between queries. The base classifiers of ensemble classifier are formed by the following methods. First, for the popular queries, we calculate the point-wise mutual information (PMI) scores of the co-occurrence queries in query click URL logs and sessions. Second, we calculate the Jaccard sore and the Cosine score of each query pair after removing the stop words in them. Third, to add semantic based methods, we use the web-based query expansion method [15] to get the Cosine similarity of each query pair, and the LDA topic model based algorithm to calculate the topic divergence similarity between each pair [1]. After building these base classifiers, we take the outputs of these base classifiers as features of a linear logistic regression model and use the liblinear library [4] to train the model.

3.3 Large-Scale Query Intent Inference

Based on the detected large-scale intent concepts C, we try to build an inference system to infer intents of online queries with high precision and acceptable coverage. The goal of query intent inference algorithm in **StarrySky** is to find the most relevant concept label c for a given query q in the candidate finding step, and to reject the uncertain case in the rejection step.

3.3.1 Candidate Finding

In this step, we focus on finding the most relevant concept for a given query q. We use the frequent n-grams to represent the features of a query. A query q with n n-grams can be represented as a feature vector $\mathbf{x}_q = \langle x_1, x_2, \cdots, x_n \rangle$. For the sake of precision and e ciency, we only use 2-gram and 3-gram here as features of each query, and use the sophisticated Laplace smoothing technique to estimate the conditional probability $p(x|c)$ of certain n-gram x given the concept label c. Each feature vector $\mathbf{x}_q$ is defined to be an one-hot encoding sparse vector, that is the value of x_i is one if query q hits n-gram x_i otherwise it is zero. These n-gram features live in a very high dimension space (i.e., about 20 million features).

Given the n-gram feature vector $\mathbf{x}_q$ of a query q, the most likely concept c^* can be inferred by the following equation by taking a Bayes-optimal approach:

$$\begin{aligned} c^* &= \operatorname*{argmax}_{c \in C} p(c|\mathbf{x}_q) \propto \operatorname*{argmax}_{c \in C} p(\mathbf{x}_q|c) \times p(c) \\ &\propto \operatorname*{argmax}_{c \in C} \sum_{i=1}^{n} \log p(x_i|c) + \log p(c). \end{aligned} \tag{3}$$

The process of Eq. 3 can be greatly sped-up by building an inverted index of the n-gram set as the non-zero features of each query are very sparse. We also use the following equation:

$$y(c|\mathbf{x}_q) = \sum_{i=1}^{n} \log p(x_i|c) + \log p(c), \tag{4}$$

to indicate the likelihood score of between the concept c and the given query q.

3.3.2 Rejection Options

After the candidate finding step, we get a list of candidate concepts $C_{cand} = \{c_i | 1 \leq i \leq k\}$ sorted by their likelihood scores with Eq. 4 in descending order, and choose the most likely one $c_1 = c^*$ with the largest likelihood score to be the predicted concept. However, even the concept with the largest likelihood score may not satisfy the requirement of relevance. A simple global threshold may help but will hurt the coverage of the algorithm. To improve the accuracy and robustness of the inference algorithm, we take two rejection options after the candidate finding step.

The first rejection option (Option 1) is straight-forward: we propose a significance test strategy based on the likelihood scores to reject the un-trustful c_1. The significance score $r(c_1, c_2|\mathbf{x}_q)$ is defined as the ratio $r(c_1, c_2|\mathbf{x}_q) = \frac{y(c_1|\mathbf{x}_q)}{y(c_2|\mathbf{x}_q)}$ between the maximal likelihood score $y(c_1|\mathbf{x}_q)$ and the second biggest one $y(c_2|\mathbf{x}_q)$. As these likelihood scores are both negative, the smaller the ratio $r(c_1, c_2|\mathbf{x}_q)$ is, the more trustful the candidate label c_1 is. In our system, we set an empirical threshold $\lambda_r = 0.80$ to reject the uncertain cases. Another acceptable choice is that when c_1 and c_2 are highly relevant in the concept graph, i.e., there is an edge between c_1 and c_2 in the concept graph, we will return the concept label c_1 directly to avoid being falsely rejected by the λ_r.

In the second rejection option (Option 2), we cast it as a relevance problem in traditional IR. By considering each concept as a document, both queries and concepts are presented by TF-IDF weighted vectors here. The rejection strategy is based on the following two similarity scores: the query side similarity score $s_q(\mathbf{v}_q, c)$ and the concept side similarity score $s_c(\mathbf{v}_q, c)$.

Let us begin from the notations. $\mathbf{v}_q$ is the feature vector of the query q, that is $\mathbf{v}_q = \langle \text{TF-IDF}(x_1), \cdots, \text{TF-IDF}(x_n) \rangle$, where TF-IDF$(x_i)$ is the TF-IDF weight of n-gram x_i. We use $\mathbf{v}_q(x_i)$ to denote the weighted feature of x_i in $\mathbf{v}_q$. For a concept c which has k n-gram features can also be represented in the same way as a TF-IDF weighted vector $\mathbf{v}_c = \langle \text{TF-IDF}(x_1), \cdots, \text{TF-IDF}(x_k) \rangle$.

The value $s_q(\mathbf{v}_q, \mathbf{c})$ of the query q and the concept c is defined as query side similarity score:

$$s_q(\mathbf{v}_q, c) = \frac{\sum_{x_i \in \mathbf{x}_q, \mathbf{x}_c} \mathbf{v}_q(x_i)}{\sum_{x_i \in \mathbf{x}_q} \mathbf{v}_q(x_i)}, \tag{5}$$

which shows the ratio of common feature weights in query. To keep a reliable precision, we set an empirical threshold $\lambda_q = 0.6$ to reject the cases whose similarity scores are less than the threshold λ_q.

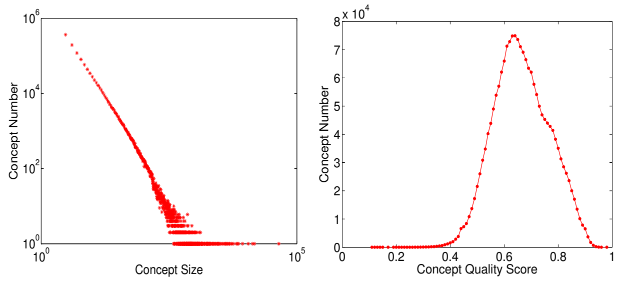
We also define the concept similarity score $s_c(\mathbf{v}_q, c)$ in the same manner:

$$s_c(\mathbf{v}_q, c) = \frac{\sum_{x_i \in \mathbf{x}_q, \mathbf{x}_c} \mathbf{v}_c(x_i)}{\sum_{x_i \in \mathbf{x}_c} \mathbf{v}_c(x_i)} \propto \sum_{x_i \in \mathbf{x}_q, \mathbf{x}_c} \mathbf{v}_c(x_i). \tag{6}$$

As it is hard to set a common threshold for all the concepts, we use the following method to calculate a local threshold λ_c for each concept c, which is defined as:

$$\lambda_c = \frac{\sum_{q \in c} s_c(\mathbf{v}_q, c)}{|c|} \propto \frac{\sum_{q \in c} \sum_{x_i \in \mathbf{x}_q, \mathbf{x}_c} \mathbf{v}_c(x_i)}{|c|}, \tag{7}$$

which indicates the average query similarity of concept c. As the threshold λ_c is only based on concept c, to speed up the rejection process in practise, we omit the identical denominator $\sum_{x_i \in \mathbf{x}_c} \mathbf{v}_c(x_i)$ of the similarity score $s_c(\mathbf{v}_q, c)$ in Eq. 6 and the local threshold λ_c in Eq. 7 during the calculation. If the concept side similarity score $s_c(\mathbf{v}_q, c)$ is less than the local threshold λ_c, the concept class c will be rejected in the inference stage.



(a) Concept Size (b) Concept Quality

Figure 2: The size and quality distributions of concepts.

4. EXPERIMENTS

In this section, we conduct experiments to verify the performance of **StarrySky** from di erent perspectives. We show the precision of concepts in the system, and demonstrate the e ectiveness of our inference system.

4.1 Dataset

In the o ine training stage, we extract web search query-URL pairs from 2 years of anonymized click logs of the Sogou search engine, and form a click-though bipartite graph. The edges in graph are weighted by click counts. To eliminate unstable clicks, we remove the noisy edges with the smallest weights. After that, there are 13 million queries and 16.5 million URLs left in the graph. We get an one-mode query co-click graph by connecting two queries with the same URL in the bipartite graph by omitting the unstable links between queries. Finally, we build a large-scale trusted query co-click graph which contains about 13 million nodes and 800 million edges.

4.2 Query Clustering Precision

First, we apply the MMO algorithm to find query clusters in the co-click graph, and the trusted large-scale co-click graph is un-weighted in the algorithm. To evaluate the cluster qualities, we need to learn the query similarity model $f(q, s)$ in Eq. 1. We keep a team of annotators to label query bid-words pairs for several years to track the qualities of our search advertisements, and there are about several millions of labeled query bid-word pairs accumulated in our data base. We sample about 1 million query bid-word pairs with balanced binary class labels (i.e., relevant or not relevant) as the training data of our ensemble classifier. We compare our ensemble model with the query expansion [15] algorithm and the LDA method [15] in a sampled test data set which contains about 165 thousand query pairs with 110767 positive instances and 54236 negative instances. Moreover, we also compare the model with some other well-known algorithm, i.e., Word2Vec [11] in the test dataset. We train the Word2Vec algorithm[2] with the Skip-gram implementation on a large search query corpus which contains 160 million queries with the window size of 5. The learned vector rep-

[2]https://code.google.com/p/word2vec/

Table 1: Examples of query intent concepts in StarrySky.

ID	Sampled Queries	Head Query
265211	苹果批发价(wholesale price of apples), 苹果批发价格(the wholesale price of apple fruit), 红富士批发价格(the wholesale price of red Fuji apples)	苹果批发价(wholesale price of apples)
195748	苹果配件批发(wholesales of apple accessories), 苹果手机配件批发(wholesales of the accessories of iphones), 苹果手机配件批发网(wholesale web sites of apple accessories)	苹果手机配件批发网(wholesale web sites of apple accessories)
403304	减肥抽脂(slimming liposuction), 抽脂手术(liposuction surgery), 吸脂减肥的价格(liposuction prices), 吸脂整形(liposuction plastic surgery)	吸脂(liposuction)
1399473	1111购物狂欢节(1111 online shopping day), 双11天猫(double 11 Tmall), 11.11淘宝(11.11 Taobao)	双十一(double 11)

resentations have 200 dimensions for words. We use two methods to represent queries base on the vector representations of word vectors. One is simply averaging the word vectors (Word2Vec-Avg) in a bag of words fashion. The other one is combined the word vectors by their TF-IDF weights (Word2Vec-Weighted). The AUC scores (Area Under **ROC** Curve) are employed to evaluate the quality of di erent query similarity algorithms. The AUC score got by our method is 0.878, while the AUC scores got by the query expansion, LDA, Word2Vec-Avg and Word2Vec-Weighted algorithms are 0.785, 0.729, 0.693 and 0.731, respectively. By considering the metrics of the AUC scores, our model is substantially better than all the baseline methods.

We preform the MMO algorithm in the real-world co-click graph. There are about 1.9 million concepts and 1.14 million edges in the concept graph. Fig. 2(a) shows that the concept sizes follow the `power-law' distributions. This phenomenon has also been found in many real-world networks [5]. There still exist some huge concepts which contain more than 10000 queries, and all these largest concepts are related to porn intents with quality scores above 0.5. Fig. 2(b) shows the concept quality distributions. The concept quality scores centre at 0.64, which indicates that most concepts are highly qualified according to our concept quality metric. To improve the qualities of concepts in **StarrySky**, we filter out 4213 concepts whose scores are less than 0.4.

We also evaluate the precision of concepts by human annotation. For each concept, we randomly sample query pairs for labeling. We conduct a manual annotating on 10000 query pairs randomly selected from the concepts. For each query pair, annotators are asked to choose one of the following options, i.e., `highly relevant', `relevant' or `not relevant'. Each query pair is assigned to 3 annotators and finally be labeled by majority voting. The annotation result shows that the `highly relevant' rate is 83.7% and the `not relevant' rate is about 1.5%, while others are `relevant'. It clearly indicates that the query clusters we got are very accurate ($\sim$ 98.5%) regarding both the highly relevant and the relevant ones as correct in our system. Table 1 gives some examples of the concepts including IDs, sampled queries and head queries with the largest daily query counts. As most query clusters of the concepts are in Chinese, we also give their English explanations. As shown in Table 1, we can also distinguish the lexical ambiguous intents such as `the wholesale of apple fruit' and `the wholesale of apple accessories'.

4.3 Inference Precision & Coverage

To evaluate the performance of the inference algorithm, we infer the intents of queries in daily search tra c. Table 2 shows the experiment comparisons of di erent rejection options. The metrics in Table 2 are precision (Pre) and coverage (Cov). These metrics are either weighted by daily query counts (i.e., Pre_1 and Cov_1) to show their performance in real-world web search applications or without query counts (Pre_2 & Cov_2) to show their performance for unique queries. We randomly sample 30000 queries from the daily search tra c to evaluate the performance, and ask three annotators to judge the correctness.

Table 2: Performance of different rejection options in concept inference algorithm.

Rejection	Pre_1	Cov_1	Pre_2	Cov_2
No Infer	99.1%	56.4%	98.8%	12.7%
Option 1	96.9%	68.2%	94.9%	31.3%
Option 2	97.4%	61.3%	96.6%	23.6%
No Reject	76.3%	87.9%	72.5%	65.5%

The evaluation results of di erent algorithms are shown in Table 2. The first row shows the performance of the algorithm without taking any inference step (i.e., No Infer), only considering the queries already existing in the concepts. This method is very accurate for the head queries, and it can cover about 56.4% query counts with little mistakes. However, it only covers 12.7% (Cov_2) unique queries without counts. The coverage (i.e., Cov_2) got by the method with rejection option 1 is larger than the score got by the method with option 2. With option 1, we can infer 33% more unique queries per day than with option 2. Finally, taking the common used nearest-neighbor approach, we choose the nearest concept induced by Eq. 4. The last row shows the result of the method taking the nearest-neighbor approach without any rejection option (No Reject). Although this method can achieve considerably coverage, its precision values are too low to meet the requirements of online applications. The whole inference process only takes about 97 microseconds for each query on average. Overall, by considering query counts, our inference algorithm can achieve up to 96% precision using either of these two rejection options with above 61% coverage. These results suggest that the online inference algorithm of **StarrySky** is potential useful and e ective, which can be used in real-time web-search applications.

5. CONCLUSION AND DISCUSSION

The ability to automatically detect query intents is a key step to enhance query understanding in modern search engines and to improve user experience. To track large-scale query intents with high precision, we introduce a practical system, **StarrySky**, which includes o ine training stage to discover fine-grained query intent concepts and online inference stage to assign queries to these found concepts. First, we cluster queries into fine-grained clusters from a query co-click graph with 13 million nodes and 800 million edges which is extracted from two years click logs of Sogou search engine and employ a set of operations to refine the intent concepts derived from clusters. Furthermore, after millions of intent concepts have been found, we build a high precision inference algorithm to assign daily queries to the concepts and try to infer their intents. To the best of our knowledge, this is the first published documentation of a real-time deployed system that infers millions of query intents in real-world web search applications with high precision and acceptable coverage. The inference algorithm can achieve the precision of 96% and the coverage of 68% considering query counts in daily web search. This system is currently supporting the analysis of query intents in Sogou sponsored search engine, and serving several real-world applications such as ad bid-word retrieval and bad case recognition tasks.

Acknowledgments

The authors would like to thank Prof. Jie Tang from Tsinghua University for his valuable comments and discussions, and all the members in ADRS (ADvertisement Research for Sponsered search) group in Sogou Inc. especially Ruining Wang for the help with parts of the data processing and experiments.

6. REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993{1022, Mar. 2003.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.*, page 10008, 9 October 2008.

[3] A. Z. Broder, M. Fontoura, and et al. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th Annual International ACM SIGIR*, pages 231{238, 2007.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871{1874, 2008.

[5] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75 { 174, 2010.

[6] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proc. Natl. Acad. Sci.*, 104:36{41, 2007.

[7] J. Guo, X. Cheng, G. Xu, and X. Zhu. Intent-aware query similarity. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 259{268, New York, NY, USA, 2011. ACM.

[8] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 267{274, New York, NY, USA, 2009. ACM.

[9] Y. Hu, Y. Qian, H. Li, D. Jiang, J. Pei, and Q. Zheng. Mining query subtopics from search log data. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 305{314, New York, NY, USA, 2012. ACM.

[10] D. Jiang, K. W.-T. Leung, and W. Ng. Query intent mining with multiple dimensions of web search data. *World Wide Web*, pages 1{23, 19 March 2015.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. E cient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[12] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 91{100, New York, NY, USA, 2008. ACM.

[13] F. Radlinski, M. Szummer, and N. Craswell. Inferring query intent from reformulations and clicks. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1171{1172, New York, NY, USA, 2010. ACM.

[14] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 841{850, New York, NY, USA, 2010. ACM.

[15] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 377{386, New York, NY, USA, 2006. ACM.

[16] V. Simonet. Classifying youtube channels: A practical system. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 1295{1304, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[17] J. Tang, Z. Meng, X. Nguyen, Q. Mei, and M. Zhang. Understanding the limiting factors of topic modeling via posterior contraction analysis. In *Proceedings of The 31st International Conference on Machine Learning*, pages 190{198, Beijing, 2014.

[18] S. I. Wang and C. D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the ACL*, pages 90{94, 2012.

[19] S. Yang, A. Kolcz, A. Schlaikjer, and P. Gupta. Large-scale high-precision topic modeling on twitter. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1907{1916, New York, NY, USA, 2014. ACM.

[20] Q. Ye, W. Bin, and W. Bai. *The Influence of Technology on Social Network Analysis and Mining*, volume 6, chapter 16 Detecting Communities in Massive Networks E ciently with Flexible Resolution, pages 373{392. Springer, 2013.