Tracking Millions of Query Intents with StarrySky and Its Applications

Qi Ye *, Feng Wang, Bo Li, Zhimin Liu and Gang Li
Sogou Inc., Sohu cyber building the 12th floor, Yard No.1, East Zhongguancun Road, Beijing, China
E-mail: yeqi@sogou-inc.com, wangfeng@sogou-inc.com, libo202442@sogou-inc.com, liuzhimin@sogou-inc.com, ligang@sogou-inc.com

Abstract. Query intent mining is a critical problem in various real-world search applications. In the past few years, dramatic advances have been witnessed in the field of query intent mining. This article presents a practical system—StarrySky for identifying and inferring millions of fine-grained query intents in daily sponsored search with high precision, and shows how real-world applications can benefit from query intent tracking results using StarrySky. Great advantages have already been achieved by deploying this system in Sogou sponsored search engine. The general architecture of StarrySky consists of three stages. First, millions of fine-grained query clusters are detected from two years' web search click logs by using a well-defined community detection algorithm. Second, query intents named as concepts are found by refining the query clusters with a series of well-designed operations. Third and foremost, a real-time inference algorithm is built for precisely assigning query intents to the detected concepts. Aside from the description of the system, several experiments are conducted to evaluate its performance. The inference algorithm achieves up to 96% precision and 68% coverage on daily search requests. It is also demonstrated that StarrySky is a practical and valuable system for tracking query intents in several real-world applications.

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 query suggestion, sponsored search and other vertical search applications. In these tasks, search queries might be only partially matched with search results lexically, and query intents possibly be misunderstood, which would cause the so-called vocabulary mismatch problem [17] in search applications and hurt user experience. This situation can be greatly improved by introducing a well-developed query intent tracking system which can give a deep understanding of query intents.

In this article, query intent tracking problem is addressed as a fine-grained query intent detection and inference task. A practical system StarrySky [43] is proposed for identifying and inferring millions of query intents in daily sponsored search with high precision and acceptable coverage.

In our system, concepts are found to represent different query intents. The concepts are just like stars shining in the query intent sky, which is where the name StarrySky comes from. As traditional clustering methods tend to find coarse-grained clusters which may not easily capture various fine-grained query intents, our system first detect millions of fine-grained query intent clusters from two years' click logs. Then, a series of post-processing operations are employed to refine the clusters and get high-quality concepts each of which contains queries with similar intent. Finally, an inference algorithm is built to assign head and tail queries to the identified concepts with high precision and acceptable coverage. Although the system mainly focuses on how to track query intents in sponsored search, its approach can also be commonly used to en-

*Corresponding author. E-mail: yeqi@sogou-inc.com hance user experience in other search tasks and recommendation systems.

The main contributions of this article are as follows:

- Detecting millions of fine-grained concepts in a massive query co-click graph created from 2 years' real-world web search click logs.
- Developing an efficient and scalable inference algorithm to track query intents with high precision.
- Demonstrating how real-world applications, such as query classification and commercial intent tracking, can benefit from query intent tracking results using StarrySky.

To the best of our knowledge, this is the first published documentation of a practical deployed system that infers millions of query intents in real-world web search applications. Researchers working on web intelligence and people skilled in the art would be able to create a similar system with the given details in this article. In the rest of the article, related work is first reviewed in Section 2. Section 3 describes an overview of our system and methods in detail. Section 4 presents applications and Section 5 shows experiments. Section 6 concludes the paper and discusses the future directions.

2. Related Work

Search query intent detecting can be characterized in different ways, and many algorithms have been proposed in the last few years. Intent of a certain query can be represented by its search goals [6], topic distribution [41], coarse-grained semantic categories [30], or fine-grained clusters [8,33] due to different usages. This work explores issues of query clustering, topic modeling, large-scale multi-class classification and search log mining.

To identify major intents of queries, various query clustering algorithms have been proposed. Sadikov et al. [33] 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. [21] present an unsupervised method for clustering queries with similar intent based on the clicked URL features. Radlinski et al. [30] use a community detection algorithm [5] to find query clusters which present query intents in a query graph. To find topics in YouTube, Gargi et al. [15] design a practical and efficient multi-stage clustering system to detect communities on a large-scale YouTube video graph based on co-watch statics. However, these clustering methods cannot be applied to tail queries which are not in the clusters, due to the lack of inference methods. Cheung and Li [8] propose an unsupervised method to find query clusters with similar intent, and classify new queries into the discovered cluster patterns. The domain coverage of the cluster patterns can reach up to only 20% of traffic in certain domains. Rephil is a huge directed graphical model developed by Google, which is trained on about 100 billion text snippets or search queries. The resulting model contains about 12 million word or compounds and about 1 million "concepts" which are clusters of co-occuring words. For a given query, it can get the most relevant concept in the model to inference its intent [27]. However, the details of Rephil have not been published.

Other related work on intent finding in short text also includes topic modeling such as Latent Dirichlet Allocation (LDA) [4] and other weakly supervised algorithms [41]. However, the traditional topic models are automatically learned by unsupervised algorithms, and there is no guarantee that the hidden topics will be aligned with pre-defined taxonomy [19]. The topics may change if the models are retrained over a different dataset. Guo et al. [18] 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 traditional topic modeling techniques such as LDA and PLSI do not work well with the short text like web search queries [17,38]. Jiang et al. [22] try to capture the latent relations between search queries via URL, session and term dimensions to improve the performance of query intent mining. In this article, we only focus on using click logs to extract the intents, as co-click relations reveal query intents more precisely.

There are also many algorithms for intent detection based on semantic classification of search queries. However, query classification is difficult due to the short, sparse and noisy nature of search queries [35,39, 40]. Simonet [36] presents a framework based on annotated semantic entities for categorizing video channels in a thematic taxonomy with high precision and recall. Phan et al. [29] 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 [39] show that Naive Bayes classifier actually does better than SVMs, so we adopt the Naive Bayes algorithm in our inference framework. Yu et al. [46] find some interesting differences between short title classification and general text classification. They find that stemming and stop-word removal are harmful, and bi-grams are very effective in title classification. Shen et al. [35] describe a hierarchical product classification system at eBay. They use the K-NN algorithm at the first level and use the SVM classifiers for the fine-grained levels. Bekkerman and Gavish [3] find that in short text classification a document's class label is mostly concentrated in a few n-grams. Yuan et al. [47] give a comprehensive survey on recent development of large-scale linear classification. They find that linear classifiers yield comparable accuracy in their text classification tasks. Joulin et al. [23] also show that linear models with the bag of n-gram features still achieve very good performance in real-world practice comparing with the deep learning neural networks.

Query intent mining shows a great advantage in enhancing the performance of real-world query classification task. A common way to enhance the precision of query classifier is to obtain extra data. Many pervious work has been proposed in this field. To classify rare queries, Broder et al. [7] use a blind feedback technique to classify web queries by using the web search results. Their method can give each query one of 6000 class labels with high accuracy. However, their method need to send the query to a search engine, and this manner makes it computationally infeasible in many online real-time tasks. Wang et al. [40] propose a framework for short text classification applications based on "bag of concepts", and the concepts are sets of entities learned from the Probase[1] knowledge base. However, there are many queries containing no entities and the coverage of this method is still limited for daily web search queries. As we mainly focus on Chinese search query classification, some widely used extra resources such as FreeBase[2] and ProBase cannot be easily applied to our tasks.

3. Methodology

This article defines query intent tracking task as follows: given a search query, the task tries to infer its intent and assigns it to a proper fine-grained query

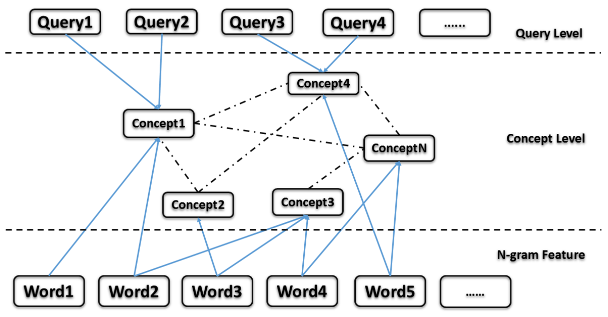


Fig. 1. The architecture of StarrySky.

concept. In StarrySky system, the fine-grained query concepts are found by an unsupervised algorithm and refined by a series of well-designed operations.

The main architecture of query concepts in StarrySky is shown in Fig. 1. In simple terms, search queries are divided into different fine-grained query clusters called concepts to indicate the query intents. Plain links show the belonging relations between queries and concepts, and relations between concepts and words. While the doted ones show the weighted edges among concepts which reveal how closely the concepts are related. People can also get more coarse level intents by considering the links among concepts as discussed in the commercial intent tracking application in Section 4. 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 hot queries found in the concept detection stage, we directly return their final concept labels by the links between queries and concepts. Otherwise, for unseen tail queries, we build a large-scale inference algorithm based on n-gram features to assign them to appropriate intent concepts. To build an inverted index to be used for efficient matching against tail queries, the links between words and concepts make the inference process much more faster. We will discuss the algorithms of our system in details.

The StarrySky system can be divided into two stages, i.e., offline training stage and online inference stage. In the offline 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 different query intents, a well developed community detection algorithm is implemented to find fine-grained clusters and a series of well-designed operations are employed to refine them to construct the final intent concepts. In the online inference stage,

[1]http://research.microsoft.com/en-us/projects/probase/
[2]https://www.freebase.com/

a large-scale multi-class classifier is built 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. For queries contained in the concepts, the system directly returns their concept labels. Otherwise, for unseen queries, the inference algorithm will assign them to appropriate intent concepts.

3.1. Query Cluster Detection

Web search click logs from Sogou commercial search engine are used 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. Query-URL bipartite graph is created by using query-URL information extracted from click logs, and is further transformed into a query co-click graph. It is worth to point out that the edges in the co-click graph are weighted by click co-occurrence counts. To ensure reliable relations for the following intent mining algorithms, a threshold is set to remove the low weighted edges. Finally, a sparse and trustable query graph is found. Similar approaches can be also found in other query intent detection approaches [33]. Without loss of generality, any graph clustering algorithm can be used to discover groups of queries in the query graph. Considering the large size of the graph, a community detection algorithm named MMO [42] is chosen in our system due to the following reasons:

1. The MMO algorithm is an iterative algorithm with nearly linear time complexity and linear space complexity, and it can be easily implemented in hadoop distributed computing platform.
2. As the goal of the MMO algorithm is not to get the global maximal value of the modularity [28], it performs well in avoiding forming huge communities caused by the modularity resolution limit problem [13].

Now, we briefly introduce the MMO algorithm, and more details of the algorithm can be found in this paper [42]. The symbols used in the MMO algorithm are summarized in Table 1.

Inspired by some proposed algorithms [5,20,31,32], the multi-resolution modularity based on the spin model can be described as

$$Q(\mathbf{c}) = J(m_{\mathbf{c}} - \gamma \frac{{k_{\mathbf{c}}}^2}{4m}), \tag{1}$$

Table 1
Summary of symbols used in the MMO algorithm.

Symbol	Description
$G = \langle V, E \rangle$	The graph with node set V and edge set E
k_i	Degree of node i
$\mathbf{c}$	A community (i.e., a cluster) with $\lvert\mathbf{c}\rvert$ nodes
$\mathcal{C}_v$	The community node v belongs to
k_v^{int}	Internal degree of node v
k_v^{ext}	External degree of node v
$k_{\mathbf{c}}^{int}$	Internal degree of community $\mathbf{c}$
$k_{\mathbf{c}}^{ext}$	External degree of community $\mathbf{c}$
$k_{\mathbf{c}}$	Total degree of community $\mathbf{c}$
$m_{\mathbf{c}}$	Number of edges in community $\mathbf{c}$

where J is a constant expressing the coupling strength and γ is a parameter expressing the relative contribution to the energy from existing and missing edges in the spin model. The resolution parameter γ enables us to span community scales from very small to very large ones, where $0 \leq \gamma < \infty$. To get a multi-resolution modularity, let J be $\frac{1}{m}$ in Eq. 1. So the new multi-scale modularity $Q_{new}(\mathbf{c})$ minus it original multi-scale modularity of community $\mathbf{c}$ is the gain of modularity $\Delta Q(\mathbf{c}, v)$ by inserting node v into the community $\mathbf{c}$. The result can be described as the following equation:

$$\begin{aligned} \Delta Q(\mathbf{c}, v) &= Q_{new}(\mathbf{c}) - Q(\mathbf{c}) \\ &= \frac{m_{\mathbf{c}} + m_{\mathbf{c}}^v}{m} - \gamma(\frac{k_{\mathbf{c}} + k_v}{2m})^2 - \frac{m_{\mathbf{c}}}{m} + \gamma(\frac{k_{\mathbf{c}}}{2m})^2 \\ &= \frac{1}{m}(m_{\mathbf{c}}^v - \gamma(\frac{k_{\mathbf{c}} k_v}{2m} + \frac{k_v^2}{4m})), \end{aligned} \tag{2}$$

where $m_{\mathbf{c}}^v$ is the number of links between node v and community $\mathbf{c}$. As the term $\frac{1}{m}$ in Eq. 2 is the same for every community, we can simplify Eq. 2 as

$$\Delta Q(\mathbf{c}, v) = m_{\mathbf{c}}^v - \gamma(\frac{k_{\mathbf{c}} k_v}{2m} + \frac{k_v^2}{4m}), \tag{3}$$

when comparing the multi-resolution modularity gains for different neighboring communities of node v. When $\gamma = 1$, Eq. 2 becomes the traditional GN modularity optimization gain, and we set $\gamma = 1$ in the following experiments.

The MMO iterative algorithm based on local quality metric optimization in Eq. 3 is summarized as follows:

1. Initially, each node belongs to a community, and set the iterative step $i = 1$.

2. For each node v, remove it from the community $\mathcal{C}_v$ it belongs to. Calculate the multi-resolution modularity gain $\Delta Q(\mathbf{c}, v)$ between node v to all the adjacent community $\mathcal{C}'_v$, where $v' \in N_v$.
3. Move node v into the adjacent community with the largest modularity gain $\Delta Q(\mathcal{C}_{v'}, v)$.
4. Repeat step 2 to step 3 and set $i = i + 1$ until the fixed sufficient I_{max} steps are reached or the multi-scale modularity $Q(\mathcal{C})$ dose not change obviously.
5. Identify the community $\mathcal{C}_i$ for each node i, and for each edge $e_{i,j}$, if $\mathcal{C}_i \neq \mathcal{C}_j$ mark $e_{i,j}$ as unlinked. Find all the components as communities in the graph.

The pseudo code of MMO is described in Algorithm 1.

Algorithm 1 The MMO algorithm

```
1: $C = \{\{v\}|v \in V\}$
2: for i = 1 to $I_{max}$ do
3:   $Q_{old} = Q(\mathcal{C})$
4:   for $v \in V$ do
5:     $\mathbf{c}_{old} = \mathcal{C}_v$
6:     $\mathbf{c}_{best} = \mathbf{c}_{old}$
7:     remove $v$ from $\mathbf{c}_{old}$
8:     $Q_{max} = \Delta Q(\mathbf{c}_{best}, v)$
9:     for $v' \in N_v$ do
10:      $Q' = \Delta Q(\mathcal{C}_{v'}, v)$
11:      if $Q'$ is better than $Q_{max}$ then
12:        $Q_{max} = Q'$
13:        $\mathbf{c}_{best} = \mathcal{C}_{v'}$
14:      end if
15:    end for
16:    insert $v$ into $\mathbf{c}_{best}$
17:  end for
18:  $Q_{new} = Q(\mathcal{C})$
19:  $\delta = Q_{new} - Q_{old}$
20:  if $\delta < \epsilon$ then
21:    break
22:  end if
23:  $Q_{old} = Q_{new}$
24: end for
25: for $e_{i,j}$ in $E$ do
26:   if $\mathcal{C}_i \neq \mathcal{C}_j$ then
27:     mark $e_{i,j}$ as unlinked
28:   end if
29: end for
30: Find all the components as communities $\mathcal{C}$.
31: return $\mathcal{C}$
```

3.2. Concept Refinement

Due to the resolution limit issue of modularity [13] and sparseness of clicks, there might still be some giant inconsistent clusters with multiple intents and small isolated ones with similar intents. To make the concepts more semantically consistent, we refine the clusters by considering query similarity measurements.

3.2.1. Quality Evaluation

The first critical problem is giant clusters which contain many small consistent sub-clusters. To reveal this problem, we present a formal quality score $r(\mathbf{c})$ for evaluating the quality of a given intent cluster $\mathbf{c}$, which is defined as:

$$r(\mathbf{c}) = \frac{\sum_{q,s \in \mathbf{c}, q \neq s} f(q,s)}{|\mathbf{c}| \times (|\mathbf{c}| - 1)}, \tag{4}$$

where $f(q,s) \in [0,1]$ is a pre-defined similarity score of a query q and query s from the same cluster $\mathbf{c}$. The value of Eq. 4 indicates the average query similarity score in each cluster c, so the larger the score is the more consistent the cluster will be. We calculate the quality score $r(\mathbf{c})$ in Eq. 4 for each cluster, and then set an empirical threshold $t_s = 0.1$ to find low quality ones. If the relevance score $r(\mathbf{c})$ of a cluster $\mathbf{c}$ is less than t_s, i.e., $r(\mathbf{c}) < t_s$, the cluster $\mathbf{c}$ should be divided into smaller cohesive sub-clusters. After the low quality cluster $\mathbf{c}$ has been found, the subgraphs $G_{\mathbf{c}}$ induced by $\mathbf{c}$ will be split into sub-clusters recursively by the MMO algorithm. Following the splitting operation, we recursively check the qualities of sub-clusters and divide them with the same splitting method if necessary.

The second 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 which have the similar intents. To solve this problem, we also define a relevance score $r(\mathbf{c}_i, \mathbf{c}_j)$ between two clusters $\mathbf{c}_i$ and $\mathbf{c}_j$ as follows:

$$r(\mathbf{c}_i, \mathbf{c}_j) = \frac{\sum_{q \in \mathbf{c}_i, s \in \mathbf{c}_j} f(q,s)}{|\mathbf{c}_i| \times |\mathbf{c}_j|}. \tag{5}$$

Eq. 5 is the average query similarity score of the queries in cluster c_i and c_j respectively, and the larger the score is the more relevant the two clusters will be.

To find out these clusters with similar intents, we need to check similarity scores among all concept pairs. To speed up this process, we only calculate $r(\mathbf{c}_i, \mathbf{c}_j)$ if $\mathbf{c}_i$ and $\mathbf{c}_j$ are connected in the co-click graph. Cluster pairs are merged together when their relevance scores are larger than a certain threshold t_d, which is 0.9 in our system. After merging high relevant cluster-s, small clusters with less than 3 queries are dropped to make sure that clusters are nontrivial.

3.2.2. *Query Similarity*

The $f(q, s)$ score in Eq. 4 and Eq. 5 is defined as the similarity between two queries. Calculating similarity between queries is an interesting and difficult problem, and a series of methods have been proposed [34,45]. This problem can be formalized as follows: given a pair of queries q and s, the goal is to train a probability binary classifier $f(q, s) \rightarrow [0, 1]$ which reveals the possibility of their similarity. It is a critical issue in many real-world IR applications, such as query recommendation and advertisement matching. However, since queries are usually very short and ambiguous, it is imprecise to calculate their similarity only based on query terms. In this part, we present a practical ensemble based algorithm to calculate the similarity between queries by considering various information other than terms.

Feature engineering is very important to machine learning tasks. We develop several groups of base classifier features:

Corpus based features For popular queries, point wise mutual information (PMI) scores are calculated by considering co-occurrence information in query-URL click logs.

Lexical based features Jaccard scores and Cosine s-cores of query pairs are calculated after removing stop words in them [45].

Semantic based features Cosine similarity based on web-based query expansion method [34] and top-ic divergence similarity based on LDA topic mod-el [4] are calculated for each query pair.

After building classifiers based on these information, taking an ensemble approach, a logistic regression model is trained by liblinear library [11] using the outputs of these classifiers as one-hot encoding features. The experiments will be discussed in the following section.

3.3. *Large-Scale Query Intent Inference*

Based on the detected large-scale intent concepts, S-tarrySky tries to build an inference algorithm to assign intents of online queries with high precision and acceptable coverage. In query intent inference stage, the most relevant concept label $\mathbf{c}^*$ for a given query q is found in the candidate finding step. After that, a rejection step is employed to decide whether the candidate label can be finally accepted.

3.3.1. *Candidate Finding*

Candidate finding step focuses on finding the most relevant concept for a given query q. A query q with s n-grams can be represented as a feature vector $\mathbf{x}_q = \langle x_1, x_2, \cdots, x_s\rangle$. For the sake of efficiency, only the 2-gram and 3-gram are kept as features. To resolve the sparsity problem, the sophisticated Laplace smoothing technique is used to estimate the conditional probability $p(x|\mathbf{c})$ of certain n-gram x given the concept label $\mathbf{c}$. 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 dimensional space (i.e., about 20 million features).

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

$$\begin{aligned}
\mathbf{c}^* &= \operatorname*{argmax}_{\mathbf{c}\in C} p(\mathbf{c}|\mathbf{x}_q) \\
&\propto \operatorname*{argmax}_{\mathbf{c}\in C} p(\mathbf{x}_q|\mathbf{c}) \times p(\mathbf{c}) \\
&= \operatorname*{argmax}_{\mathbf{c}\in C} \prod_{i=1}^{s} p(x_i|\mathbf{c}) \times p(\mathbf{c}) \qquad (6) \\
&\propto \operatorname*{argmax}_{\mathbf{c}\in C} \sum_{i=1}^{s} \log p(x_i|\mathbf{c}) + \log p(\mathbf{c}),
\end{aligned}$$

which indicates that $\mathbf{c}^*$ is the most likely concept given the condition $\mathbf{x}_q$. As non-zero features of each query are very sparse, calculating Eq. 6 can be greatly sped-up by building an inverted index. We also use the following equation:

$$y(\mathbf{c}|\mathbf{x}_q) = \sum_{i=1}^{s} \log p(x_i|\mathbf{c}) + \log p(\mathbf{c}), \qquad (7)$$

to measure the similarity between concept $\mathbf{c}$ and query q.

3.3.2. *Rejection Options*

After the candidate finding step, we get a list of candidate concepts $C_{cand} = \{\mathbf{c}_i | 1 \leq i \leq k\}$ sorted by their likelihood scores with Eq. 7 in descending order, and choose the most likely one $\mathbf{c}^*$ 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, two rejection options are taken after the candidate finding step.

The first rejection option (Option 1) is straightforward: A significance test strategy based on the likelihood scores is proposed to reject the un-trustful $\mathbf{c}_1$. The significance score $r(\mathbf{c}_1, \mathbf{c}_2|\mathbf{x}_q)$ is defined as the ratio $r(\mathbf{c}_1, \mathbf{c}_2|\mathbf{x}_q) = \frac{y(\mathbf{c}_1|\mathbf{x}_q)}{y(\mathbf{c}_2|\mathbf{x}_q)}$ between the maximal likelihood score $y(\mathbf{c}_1|\mathbf{x}_q)$ and the second biggest one $y(\mathbf{c}_2|\mathbf{x}_q)$. As these likelihood scores are both negative, the smaller the ratio $r(\mathbf{c}_1, \mathbf{c}_2|\mathbf{x}_q)$ is, the more trustful the candidate label $\mathbf{c}_1$ is. An empirical threshold $\lambda_r = 0.80$ is employed to reject the uncertain cases.

In the second rejection option (Option 2), a traditional IR approach is employed to make the rejection decision. By considering each concept as a document, both queries and concepts are represented by TF-IDF weighted vectors here. The rejection strategy is designed based on the following two similarity scores: the query side similarity score $s_q(\mathbf{v}_q, \mathbf{c})$ and the concept side similarity score $s_{\mathbf{c}}(\mathbf{v}_q, \mathbf{c})$.

The feature vector of query q is defined as $\mathbf{v}_q = \langle \text{TF-IDF}(x_1), \cdots, \text{TF-IDF}(x_n) \rangle$, which contains the TF-IDF weight of each n-gram. The concept $\mathbf{c}$ can also be represented in the same way as a TF-IDF weighted vector $\mathbf{v}_{\mathbf{c}} = \langle \text{TF-IDF}(x_1), \cdots, \text{TF-IDF}(x_k) \rangle$ of its k n-grams.

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

$$s_q(\mathbf{v}_q, \mathbf{c}) = \frac{\sum_{x_i \in \mathbf{x}_q \cap \mathbf{x}_{\mathbf{c}}} \mathbf{v}_q(x_i)}{\sum_{x_i \in \mathbf{x}_q} \mathbf{v}_q(x_i)}, \tag{8}$$

which is the ratio of the sum weights of common features to the sum weights of given query. To keep an acceptable precision, an empirical threshold $\lambda_q = 0.6$ is set to reject the cases whose similarity scores are less than it.

The concept similarity score $s_{\mathbf{c}}(\mathbf{v}_q, \mathbf{c})$ is also defined in the same manner:

$$s_{\mathbf{c}}(\mathbf{v}_q, \mathbf{c}) = \frac{\sum_{x_i \in \mathbf{x}_q \cap \mathbf{x}_{\mathbf{c}}} \mathbf{v}_{\mathbf{c}}(x_i)}{\sum_{x_i \in \mathbf{x}_{\mathbf{c}}} \mathbf{v}_{\mathbf{c}}(x_i)} \propto \sum_{x_i \in \mathbf{x}_q \cap \mathbf{x}_{\mathbf{c}}} \mathbf{v}_{\mathbf{c}}(x_i). \tag{9}$$

As it is hard to set a common threshold for all the concepts because of the huge difference in sizes of concepts, a local threshold $\lambda_{\mathbf{c}}$ is used for each concept $\mathbf{c}$, which is defined as:

$$\begin{aligned} \lambda_{\mathbf{c}} &= \frac{\sum_{q \in \mathbf{c}} s_{\mathbf{c}}(\mathbf{v}_q, \mathbf{c})}{|\mathbf{c}|} \\ &= \frac{\sum_{q \in \mathbf{c}} \sum_{x_i \in \mathbf{x}_q \cap \mathbf{x}_{\mathbf{c}}} \mathbf{v}_{\mathbf{c}}(x_i)}{|\mathbf{c}| \times \sum_{x_i \in \mathbf{x}_{\mathbf{c}}} \mathbf{v}_{\mathbf{c}}(x_i)} \\ &\propto \frac{\sum_{q \in \mathbf{c}} \sum_{x_i \in \mathbf{x}_q \cap \mathbf{x}_{\mathbf{c}}} \mathbf{v}_{\mathbf{c}}(x_i)}{|\mathbf{c}|}, \end{aligned} \tag{10}$$

which indicates the average query similarity of concept $\mathbf{c}$. To speed up the rejection process in practice, we omit the identical part $\sum_{x_i \in \mathbf{x}_{\mathbf{c}}} \mathbf{v}_{\mathbf{c}}(x_i)$ in Eq. 9 and Eq. 10 during the calculation. If the concept side similarity score $s_{\mathbf{c}}(\mathbf{v}_q, \mathbf{c})$ is less than the local threshold $\lambda_{\mathbf{c}}$, the concept class $\mathbf{c}$ will be rejected in the inference stage.

In the following section, we will conduct two experiments to show the effectiveness of StarrySky in different real-world intent tracking tasks, i.e., one for coarse-grained query taxonomy classification and the other one for temporal fine-grained commercial intent tracking. In order to show the improvements for query classification by using StarrySky, we will give more details of experiments in Section 5.

4. Applications

In this section, we present two examples to demonstrate how real-world applications can benefit from query intent tracking results using StarrySky. As we have mentioned in Section 2, intent of a certain query can be represented by its search goals such as its coarse-grained semantic categories or fine-grained clustering approach. Taking a coarse-grained approach, the first application is to use StarrySky to enhance the precision of coarse-grained classification algorithms with manually manageable categories. By taking a quite different approach, the second application tries to identify the commercial intents and track their changes over time. More details of the applications will be discussed in the following parts in this section.

4.1. Multi-class Query Classification

Query classification is a difficult task to understand coarse-grained user search intents, due to the sparse, noisy and ambiguous nature of queries. In real-word search query classification tasks, domain experts prefer to maintain a small taxonomy which can be easily managed manually. The taxonomy of web query classification tasks may contain from tens of to thousands of categories, and some of them are organized in hierarchical structure [6,7,35]. Traditional classification algorithms such as Naive Bayesian method and linear support vector machines have been widely used in short text classification [35,39,47]. These methods suppose that the feature space is separable. Recent studies have also shown that there are tremendous fine-grained intent topics in daily search traffic [30]. However, there is a big gap between pre-defined small size taxonomy and the massive coherent topics in each category.

4.1.1. The Pomegranate Phenomenon

The gap between predefined taxonomy and fine-grained topics shows up not only in quantity but also in semantics which leads us to find an interesting phenomenon in query classification which we call "the pomegranate" [44]. This phenomenon indicates that there may be massive fine-grained topics in each category of the taxonomy, and these relatively independent fine-grained topics in the categories are just like seeds in pomegranates. Furthermore, the fine-grained topics in the same category may be textually more relevant to the topics in other categories. Because of this phenomenon, the hyper-planes between different categories in the textual feature space are hard to find. Thus, considering query intent features can get better performance than only with query textual features.

To have a close insight of "the pomegranate phenomenon", let us show some examples from real-world search engine. "Canon printer" seems more relevant to "the driver of Canon printer" than to "Gree air conditioner", if we only consider textual features. However, we prefer to categorize "Canon printer" and "Gree air-conditioner" into the same category as "electronics", as users who search these queries usually tend to buy these electronic products. While we tend to classify "the driver of Canon printer" as the "software" category, as users are more likely to have already bought the printer and just want to download the software from web-sites. There are more examples. The textual relation of "gynecology doctors online viewing" seems more textually relevant to "gynecological examining" than to "free high-definition movies". To be precise, "gynecology doctors" is the name of a TV play and "gynecology doctors online viewing" belongs to the category of "entertainment", as users want to find the TV play from online web-sites. Meanwhile, most users who search for "gynecological examining" tend to find information of medical service. So is the case of "gaming PC" and "PC game". The former belongs to the "electronics" category, and the latter should be classified as the "computer game".

4.1.2. K-Nearest Topic classifier

To overcome the problem caused by "the pomegranate phenomenon", we propose a general method to enhance the performance of real-world query classification based on the labeled concepts of StarrySky.

After detecting millions of fine-grained concepts in StarrySky, each concept get a category label in StarrySky by majority voting based on the query labels in it. This majority voting process can be formalized as the following purity function $f(l)$, i.e.,

$$l^* = \operatorname*{argmax}_{l \in L} f(l) = \operatorname*{argmax}_{l \in L} \sum_{q \in \mathbf{c}} \frac{\mathbb{I}(l_q = l)}{|\mathbf{c}|}, \quad (11)$$

where $\mathbb{I}(\cdot)$ is an indicator function and L is the set of class labels. To make the class label of each cluster more trustable, we remove the concepts whose purities are less than an empirical threshold λ. Finally, we call these fine-grained concepts with consistent labels as topics.

For a given query q, a K-Nearest Topic classifier (KNT) is proposed to assign a class label l^* to the query based on the K most relevant topics. We use the likelihood score in Eq. 7 as a distance metric to find the K nearest topics, and give each query a coarse-grained category label by taking a majority voting approach. We describe the KNT method in algorithm 2. To speed up the algorithm, for queries in the topics, we directly return their labels as shown in the first three lines in algorithm 2. After the topic likelihood ranking step in line 4, a list of nearest topics $C_N = \{\mathbf{c}_i | 1 \leq i \leq N\}$ are sorted in descending order by their likelihood scores with Eq. 7. To make the assigning process more precisely, we simultaneously consider two different similarity scores, i.e., $s_q(\mathbf{v}_q, \mathbf{c})$ in Eq. 8 and $s_{\mathbf{c}}(\mathbf{v}_q, \mathbf{c})$ in Eq. 10. Finally, the given query q is labeled as the most common category among its K nearest topics in C_N by majority voting. We will demonstrate that our algorithm is highly effective at query classification in the next section.

Algorithm 2 The K-Nearest Topic classifier algorithm.

```
Require:
    The query $q$, the set of all labeled topics $C$, the parameter $K$, the maximal length $N$ of the nearest topics set $C_N$.
 1: if q in a topic $\mathbf{c} \in C$ then
 2:     return the class label of $\mathbf{c}$.
 3: end if
 4: Get a set of nearest topics $C_N \subset C$ using Eq. 7.
 5: $C_k = \varnothing$
 6: for $c \in C_N$ do
 7:     Get the query side similarity score $s_q(\mathbf{v}_q, \mathbf{c})$ in Eq. 8.
 8:     Get the topic side similarity score $s_c(\mathbf{v}_q, \mathbf{c})$ in Eq. 9.
 9:     if $s_q(\mathbf{v}_q, \mathbf{c}) > \lambda_q$ and $s_\mathbf{c}(\mathbf{v}_q, \mathbf{c}) > \lambda_\mathbf{c}$ then
10:         $C_k = C_k \bigcup \{\mathbf{c}\}$
11:     end if
12: end for
13: Sort $C_k$ by their likelihood scores in Eq. 7 in descending order.
14: Get a topic set $C_K$ by selection the top $K$ concepts in $C_k$.
15: Get class label $l^*$ from the labels in $C_K$ by majority voting.
16: return The class label $l^*$.
```

4.2. Commercial Search Intent Tracking

To evaluate the user intent concepts tracked by our system, a case study is proposed based on StarrySky to analyze the search traffic during the period of the 'online shopping day on November 11'. November 11 has become the largest online shopping day in the world, with sales in Alibaba's sites Tmall and Taobao at 9.3 billion dollars in 2014[3]. We try to identify the major commercial intents and track their changes during that period with our system. We compare query counts of concepts in two periods from anonymized query logs to track the trends of user intents. These two datasets contain queries of two entire weeks from October 6 to October 12, and from November 4 to November 11 in 2014. We use the former dataset to get a background knowledge of daily query count for each concept. And then we define the concept $\mathbf{c}$'s activity score on a given day d as:

$$\alpha_d(\mathbf{c}) = \frac{PV_d(\mathbf{c})}{PV_{avg}(\mathbf{c}) + PV_d(\mathbf{c})}, \qquad (12)$$

[3]http://en.wikipedia.org/wiki/Singles_Day

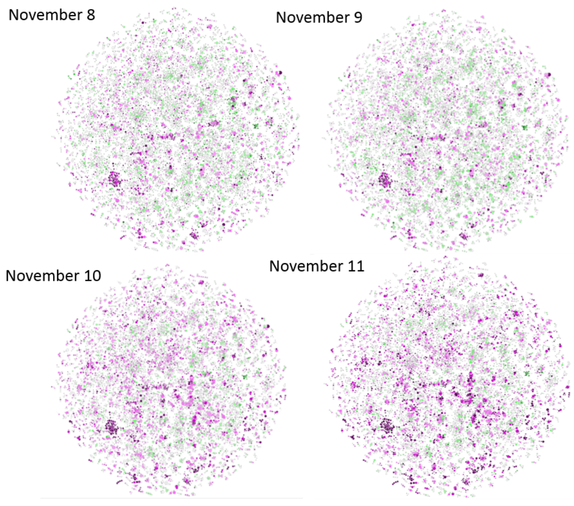


Fig. 2. The evolution of temporal commercial intent concept graph.

where $PV_{avg}(\mathbf{c})$ denotes the baseline daily query count of concept $\mathbf{c}$ in October, and $PV_d(\mathbf{c})$ is the query count of concept $\mathbf{c}$ on day d in November.

To identify the concepts with commercial intents, a pre-developed query classifier in our company is employed to give each concept a category label by majority voting. To focus on the analysis of commercial intents, only the concepts of 4 categories are chosen: daily supplies, household electrical appliances, e-commercial web sites and adult products. To overview the major concepts, a graph with chosen commercial intent concepts is constructed, and the tail concepts with query counts less than 100 in two weeks have also been removed. Finally, the graph contains about 9000 commercial intent concept nodes and 11000 edges as shown in Fig. 2 and Fig. 3.

Fig. 2 shows the activity scores $\alpha_d(\mathbf{c})$ of concepts from November 8 to November 11. The size of the node indicates the query counts of concepts in each day. The one with green color indicates the query count declines (e.g., $\alpha_d(\mathbf{c}) \leq 0.45$), and the one with purple color indicates the query count increases (e.g., $\alpha_d(\mathbf{c}) \geq 0.55$). The darker these colors are, the more obvious these trends are. The concepts with activity scores $\alpha_d(\mathbf{c})$ between 0.45 and 0.55 are regarded as stable ones and are colored as white. As shown in Fig. 2, during November 9 and November 11, there are many green nodes in graph. However, much more purple nodes appear during November 10 and November 11. There are more purple nodes on November 11, and many of them become darker than usual. This interest-

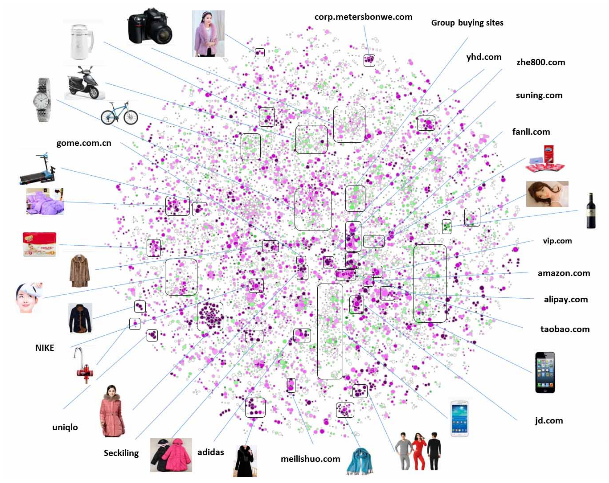


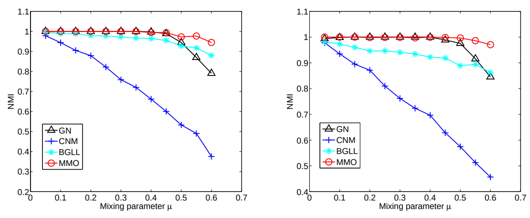
Fig. 3. The commercial intent concept graph on November 11, 2014 found by StarrySky.

ing phenomenon indicates that many commercial intents burst during these two days, and it is congenial with reason and common sense for the search traffic of these days.

Fig. 3 gives a close observation of the commercial graph on November 11, some key goods or products are listed in the graph. As shown in Fig. 3, in the center of the graph, there are many purple nodes which stand for famous 'e-commercial web sites' in China, such as 'www.taobao.com', 'www.jd.com', 'www.suning.com' and 'amazon.cn'. In the 'daily supplies' category, users are very likely to search for various clothes, especially the winter clothes such as down jackets, overcoats, furs and thermal under-wears. Users also tend to search for various consumer goods, such as diapers, cosmetics and some famous brands like *Nike*, *Uniqlo* and *adidas*. However, the counts of queries for the group-buying web sites and mobile phones such as 'Samsung mobile phone' and 'iphone' do not have apparent growth trends on that day. This may be caused by the low discount of these goods. The graph also indicates that users do not tend to search for bikes and electric motorcycles, and this may be caused by the cold weather which is not suitable for outdoor riding during that time.

5. Experiments

In this section, experiments are conducted to verify the performance of StarrySky from different perspectives.



(a) $\gamma = 2, \beta = 1$ (b) $\gamma = 2, \beta = 2$

Fig. 4. Performance of community detection algorithms, in the benchmark graphs. Each point is an average over 40 realizations of the networks.

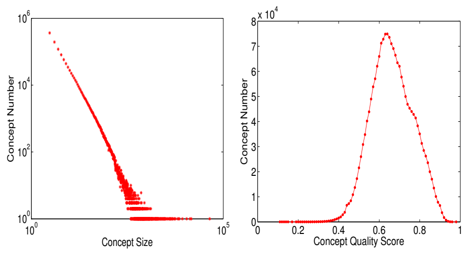
5.1. Offline Dataset Preparing

For the offline training stage, web search query-URL pairs are extracted from 2 years of anonymized click logs in a commercial search engine to form a click-though bipartite graph. The edges in graph are weighted by click counts. To eliminate unreliable clicks, the noisy edges with the smallest weights are removed. After that, there are 13 million queries and 16.5 million URLs left in the graph. An one-mode query co-click graph is formed by connecting two queries with co-click URLs in the bipartite graph. Finally, the large-scale reliable query co-click graph contains about 13 million nodes and 800 million edges by omitting unreliable links.

5.2. Concept Extraction

First, to get the performance of different community detection algorithms, we generate controllable random networks with already known community structure. We evaluate the performances of the MMO, the GN [16], the CNM [9] and the BGLL [5] algorithms on the LFR artificial benchmark random graphs [24] for which the built-in community structure is already known. In the experiments, we use the normalized mutual information score (NMI) [10] as the metric of community detection algorithms. The LFR graphs are generated with 3 parameters γ, β and μ. In details, γ and β control the degree and community sizes which follow the power-law distributions parameterized by them. Each node shares a fraction $1 - \mu$ of its links with nodes in its own community and a fraction u with other nodes in benchmark network.

As shown in Fig. 4, all the graphs contain 1000 nodes, and we set $\beta = 1$ and $\beta = 2$, respectively. The MMO algorithm performs quite perfectly when these communities are more fuzzy. In Fig. 4(a), when μ is



(a) Concept Size (b) Concept Quality

Fig. 5. The size and quality distributions of concepts.

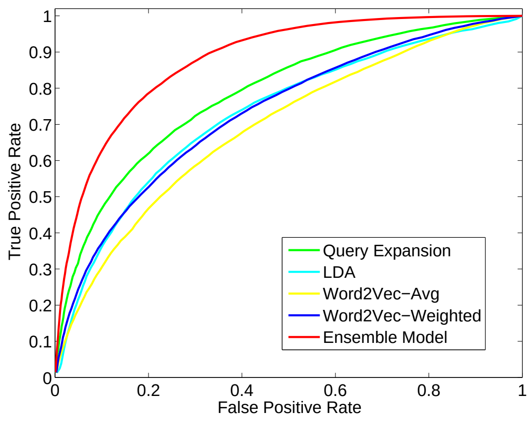


Fig. 6. The ROC (Receiver Operating Characteristic) curves of different query similarity algorithms.

0.60, the NMI score got by the MMO is 0.94, while the result got by the BGLL algorithm is only 0.88. In Fig. 4(b), the MMO algorithm is also much more robust compared to other algorithms. When μ equals 0.60, the NMI score of clusters got by the MMO algorithm is 0.97, while the score got by the BGLL algorithm is 0.86. Both results show that the MMO algorithm is very robust in the artificial random graphs with built-in communities, and it has the best performance comparing with other algorithms in the experiment. The reason is that, when the network becomes more fuzzy, most of these community detection algorithms tend to produce large clusters, while the MMO algorithm tend to find smaller ones. According to our experience in commercial search engine, using the coarse-grained clustering methods, it is hard to detect unambiguous query intents. So we prefer the fine-grained clusters to the larger ones.

Second, to evaluate the cluster qualities on real-world co-click graph which has no ground-truth of community structure, we need to learn the query similarity model $f(q, s)$ in Eq. 4. A team of annotators has been kept to label query bid-words pairs for several years to track the qualities of search advertisements in our company, and we have collected about several millions of labeled query bid-word pairs. We sample about 1 million query bid-word pairs with balanced binary class labels (i.e., relevant or not relevant) as the training dataset for the ensemble classifier. The ensemble model is compared with the query expansion [34] algorithm and the LDA method [4] in the experiment with another sampled dataset as testing data which contains about 165 thousand query pairs. Moreover, we also compare the model with some other well-known algorithm, i.e., Word2Vec [26]. The Word2Vec algorithm[4] is trained with the skip-gram implementation on a large search query corpus which contains 160 million queries with the window size of 5 and the word vectors have 200 dimensions. Based on the representations of words, two methods are used to get the distributed representations of queries. One is simply averaging the word vectors (Word2Vec-Avg) in a bag of words fashion. The other one is combining the word vectors by their TF-IDF weights (Word2Vec-Weighted). The AUC scores (Area Under ROC Curve shown in Fig. 6) are employed to evaluate the quality of different query similarity algorithms. The AUC score got by our ensemble 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. Considering the AUC scores, the proposed ensemble model is substantially better than all the baseline methods.

With the help of the ensemble model, we apply the proposed concept detection algorithm on the large-scale co-click graph, and find about 1.9 million concepts. Fig. 5(a) shows the statistics of the concept sizes which follow the 'power-law' distributions. This phenomenon has also been found in many real-world networks [12]. Some huge concepts which contain more than 10000 queries are still found, all of which are found to be related to porn intents with quality scores above 0.5 during the observations. Fig. 5(b) shows the concept quality distributions. The concept quality scores centre at 0.64, which indicates that most concepts are highly qualified. To improve the qualities of concepts in StarrySky, 4213 concepts whose scores are less than 0.4 are discarded in the following experiments.

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

Table 2

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)

Table 3

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%
SimRank++	92.7%	58.9%	91.0%	20.8%
Nearest-Neighbor	76.3%	87.9%	72.5%	65.5%

The precision of the concepts is also evaluated by human annotation. For each concept, we randomly sample query pairs for labeling. We conduct a manual annotating on randomly selected 10000 query pairs. 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 concepts are very accurate ($\sim 98.5\%$) regarding both the highly relevant and the relevant ones as correct in our system. Table 2 gives some examples of the concepts including IDs, sampled queries and head queries with the largest daily query counts. As most queries are in Chinese, their English explanations are also given in the table. As shown in Table 2, the lexical ambiguous intents such as ‘the wholesale of apple fruit’ and ‘the wholesale of apple accessories’ can be well distinguished.

5.3. *Inference Precision & Coverage*

To evaluate the performance of the inference algorithm, we infer the intents of queries sampled from daily search traffic. Table 3 shows the experiment results of different rejection options. The metrics in Table 3 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 considering query counts (Pre_2 & Cov_2) to show their performance for unique queries. About 30000 queries are randomly sampled from the daily search traffic to evaluate performances. An inferred concept is returned for each query, then we randomly sample a query in the concept and ask three annotators to judge its correctness.

The evaluation results of different algorithms are shown in Table 3. The first row shows the performance of the algorithm without taking any inference step (i.e., No Infer), only considering the queries already existed in concepts. This method is very accurate for the head queries, and can cover about 56.4% query counts with few mistakes. However, it only covers 12.7% (Cov_2) unique queries. 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 cover 33% more unique queries than with option 2. We also compare our method with the SimRank++ algorithm which is a commonly used collaborative filtering algorithm for query suggestion [1]. By using the same bipartite graph as StarrySky, we run the SimRank++ for relevant query rewriting and record the top 10 similar queries for each query of our query samples. As we can also get query pairs got by SimRank++, we can use the same relevance metrics to evaluate its precision and coverage. SimRank++ has a better coverage than StarrySky without taking any inference step (i.e., No Infer). That is because the query graph to extract concepts is optimized from the raw bipartite graph with reasonable reduction of graph size, so it covers less queries than the raw graph. However, as SimRank++ does not contain any inference method for the unseen queries, it cannot be applied to tail queries which are not in the bipartite graph and it covers less queries than our algorithm. Furthermore, our algorithm with both rejection options can be much more precise than the results got by SimRank++. Finally, taking the common used nearest-neighbor approach, the nearest concept induced by Eq. 7 is chosen as a comparing method. The last row shows the result of taking the nearest-neighbor approach without any rejection option (Nearest-Neighbor). Although this method can achieve considerable coverage, its precision values are too low to meet the requirements of online applications. 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. The whole inference process only takes about 97 microseconds for each query on average. These results suggest that the online inference algorithm of StarrySky is potentially effective and efficient. Our evaluation did not include topic models such as LDA, as both of our experience and a number of researchers show that these models obtain suboptimal performance on short texts such as queries [17].

5.4. Multi-class Query Classification

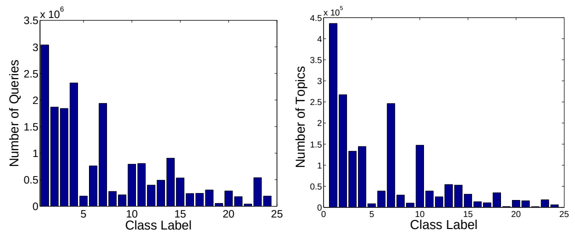
In this part, experiments are conducted to demonstrate the improvements of query classification by using the query intent results from StarrySky.

5.4.1. Training Dataset and Metrics

In this experiment, we use a pre-defined taxonomy with nearly 300 mutually exclusive hierarchical categories maintained in our company. For simplicity, we only consider 24 first-level categories of the taxonomy. We sample about 17 million unique queries from the web search logs, and use some tricks such as manual classification, semi-supervised learning and hand-craft rules to give these queries exclusive labels to make this annotating process more fast and easy. The labeled dataset still contains noise whose average precision is about 95.5%. As eliminating the noise from such a big dataset is quite expensive, we do not attempt to correct the wrong labels.

Fig. 7(a) shows the histogram of labeled query numbers in different categories. Clearly, the numbers of queries in different categories are highly imbalanced. The smallest one only contributes 0.21% of the whole training dataset, while the largest one contributes about 17.72%.

The quality of a classification algorithm is usually measured in terms of precision and recall. However, in this article, we use coverage instead of recall because coverage is more practical and meaningful in real-world query classification applications [37,41]. There are at least two advantages of using coverage rather than recall in real-world evolving dataset. First, coverage is much easier to estimate than recall. We would need 24 labeling tasks to know the ground truth label of queries when estimating recall. Second, coverage can reveal the performance of classifier in evolving search queries rather than recall which is obtained in a closed test set.



(a) The number of queries (b) the number of topics

Fig. 7. The histogram of the number of queries and topics in training dataset in each category.

To evaluate the overall performance of classifiers in imbalanced multi-class classification tasks, we use the following two metrics, i.e., the average precision (AvgPre) and the geometric mean of precision (GMPre) [2,14]:

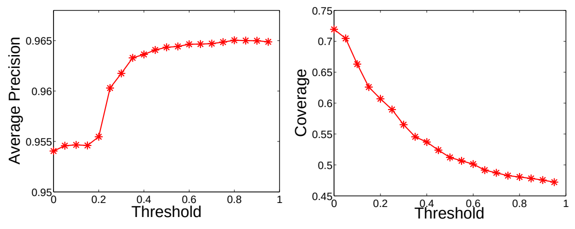
$$AvgPre = \frac{1}{|L|}\sum_{i=1}^{|L|} Pre_i, \qquad (13)$$

and

$$GMPre = \sqrt[|L|]{\prod_{i=1}^{|L|} Pre_i}, \qquad (14)$$

where $|L|$ is the number of classes and Pre_i is the precision of class i.

We use n-gram ($n = 1, 2, 3$) here as features for traditional classifiers. There are about 35 million features in the experiment. Finally, we get about 1.78 million labeled topics by setting the empirical threshold $\lambda = 0.5$. The histogram of topic numbers in different categories are shown in Fig. 7(b). The largest one contains 436303 topics, while the smallest one only contains 1724 topics. The result shows that the fine-grained topics are also highly imbalanced. We evaluate the average accuracy of the classified queries in these



(a) Average precision (b) Coverage

Fig. 8. The values of average precision and coverage got by the KNT algorithm with different λ_q threshold in daily real-world traffic.

purified topics by human annotation. The average precision is about 96.96%.

5.4.2. *The Performance of KNT Classifier*

We randomly sample 20000 queries from daily search traffic, and ask 3 annotators to select the class labels for each query. These labeled queries are used to evaluate the performance of the KNT classifier with different values of λ_q as shown in Fig. 8(a) and Fig. 8(b) by considering the query counts in real-world search traffic. Considering both precision and coverage, we choose a threshold of λ_q, i.e., $\lambda_q = 0.25$, and the average precision remains 96.02% and the coverage is about 58.95%. In the experiment, we set the number of the nearest neighbors $K = 3$ as the default setting.

In the following part, we study the performance of classifiers by combining the KNT with the MNB classifier and with the SVM classifier with a linear kernel [11], respectively. We also use the MNB classifier with a uniform prior distribution (MNB$_U$) to overcome the imbalanced class problem in priors as a comparison. The linear SVM classifier with default parameters is trained by the liblinear library [11]. We evaluate their performance by real-world search queries with search counts considered.

Table 4 shows that both the MNB and the MNB$_U$ classifiers work better than the SVM classifier. The result is similar with the conclusion given by Wang and Manning [39] in short text classification. The MNB$_U$ classifier works slightly better than the MNB classifier in real-world query classification. The KNT classifier achieves the highest average precision of 96.02% with the coverage of 58.95% . By incorporating the KNT classifier with traditional classifiers, both the precision and the coverage values of the traditional classifiers are greatly improved. The last line in Table 4 shows that the KNT + MNB$_U$ classifier for imbalanced data works best considering the average precision metric (AvgPre) and the geometric mean of precision metric (GMPre). The KNT + MNB$_U$ classifier achieves the average precision of 91.16% with nearly 100% coverage, while the precision got by a single SVM classifier is only about 85.10% covering about 95.13% daily traffic. By combining the KNT algorithm, the traditional classifiers get about 5.46%, 4.90% and 4.82% relative improvement respectively in the average precision and coverage as shown in Table 4. As it has been suggested that the traditional classifiers are still considered as strong baselines for text classification problems [23], we regard that the performance of traditional classifiers can be greatly improved by introducing our KNT classifier.

Table 4

Performance of different classification algorithms in real-world datasets.

Model	**AvgPre**	**GMPre**	**Coverage**
SVM	85.10%	84.39%	95.13%
MNB	86.09%	85.75%	95.13%
MNB$_U$	86.97%	86.77%	95.13%
KNT + SVM	89.75%	89.50%	99.35%
KNT + MNB	90.31%	90.06%	99.35%
KNT + MNB$_U$	**91.16%**	**90.90%**	99.35%

6. 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 offline 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. Furthermore, we also demonstrate that real-world applications can benefit from StarrySky for enhancing the performances of real-world query classification tasks and for tracking daily commercial query intents. 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.

As the query intents in real-world search engines are evolving over time, it is important to detect these newly appearing ones and discard the expired ones automatically. We could improve the intent detection algorithm to track the involving intents in the future. Besides that, there are still many avenues of the future work. The most important one is motivated by the issue that the meanings of words may change over time, especially in the cyber-words, e.g., 'apple' before and after the appearance of Apple Inc. In this case, we need to find these ambiguous queries and adjust the concept assignment. As it is not the major focus here, we leave it to the future work.

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.

References

[1] I. Antonellis, H. G. Molina, C. C. Chang. Simrank++: query rewriting through link analysis of the click graph. Proceedings of the VLDB Endowment, 2008, 1(1): pages 408-421.

[2] R. Barandela, J. S. Sanchez, and et al. Strategies for learning in class imbalance problems. Pattern Recognition, 36(3):849 – 851, 2003.

[3] R. Bekkerman and M. Gavish. High-precision phrase-based document classification on a modern scale. In Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2011, pages 231 – 239.

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

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

[6] A. Broder. A taxonomy of web search. SIGIR Forum, 36(2):3–10, Sept. 2002.

[7] 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, 2007, pages 231–238.

[8] J. C. K. Cheung and X. Li. Sequence clustering and labeling for unsupervised query intent discovery. In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, New York, NY, USA, 2012, pages 383–392.

[9] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. Phys. Rev. E, 70(6):066111, December 2004.

[10] L. Danon, J. Duch, A. Arenas, and A. Dı́az-guilera. Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment, 9008:09008, 2005.

[11] 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.

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

[13] S. Fortunato and M. Barthelemy. Resolution limit in community detection. Proc. Natl. Acad. Sci., 104:36–41, 2007.

[14] M. Galar, A. Fernandez, and et al. Empowering difficult classes with a similaritybased aggregation in multi-class classification problems. Information Sciences, 264:135–157, apr 2014.

[15] U. Gargi, W. Lu, V. Mirrokni, and S. Yoon. Large-scale community detection on youtube for topic discovery and exploration. In in Proc. of the Fifth international AAAI Conference on Weblogs and Social Media, 2011, pages 486–489.

[16] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. Proc. Natl. Acad. Sci., 99(12):7821–7826, June 2002.

[17] M. Grbovic, N. Djuric, and et al. Scalable Semantic Matching of Queries to Ads in Sponsored Search Advertising. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, New York, NY, USA, pages 375–384.

[18] 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, New York, NY, USA, 2011, pages 259–268.

[19] 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, New York, NY, USA, 2009, pages 267–274.

[20] Y. Hu, H. Chen, and et al. Comparative definition of community and corresponding identifying algorithm. Physical Review E, 78:026121, 2008.

[21] 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, New York, NY, USA, 2012, pages 305–314.

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

[23] A Joulin, E Grave, P Bojanowski, et al. Bag of Tricks for Efficient Text Classification. arXiv preprint arXiv:1607.01759, 2016.

[24] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. Physical Review E, 78(4):046110, Oct 2008.

[25] Shangsong Liang, Emine Yilmaz, and Evangelos Kanoulas. Dynamic Clustering of Streaming Short Documents. In Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA,

2016, pages 995–1004.

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

[27] K. P. Murphy. Chapter 26. Machine Learning: A Probabilistic Perspective. The MIT Press, 2012.

[28] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. Physical Review E, 69:026113, 2004.

[29] X. H. Phan, L. M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from largescale data collections. In Proceedings of the 17th International Conference on World Wide Web, New York, NY, USA, 2008, pages 91–100.

[30] 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, New York, NY, USA, 2010, pages 1171–1172.

[31] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. Physical Review E, 76(3):036106, Sep 2007.

[32] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. Physical Review E, 74(1):016110, Jul 2006.

[33] 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, New York, NY, USA, 2010, pages 841–850.

[34] 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, New York, NY, USA, 2006, pages 377–386.

[35] D. Shen, J.-D. Ruvini, and B. Sarwar. Large-scale item categorization for ecommerce. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, New York, NY, USA, 2012, pages 595- – 604.

[36] V. Simonet. Classifying youtube channels: A practical system. In Proceedings of the 22Nd International Conference on World Wide Web Companion, Republic and Canton of Geneva, Switzerland, 2013, pages 1295–1304.

[37] C. Sun, N. Rampalli, F. Yang, and A. Doan. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. Proceedings of the VLDB Endowment, 7(13), August 2014, pages 1529–1540.

[38] 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, Beijing, China, 2014, pages 190–198.

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

[40] F. Wang, Z. Wang, and et al. Concept-based short text classification and ranking. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, Shanghai, China, November 3–7 2014, pages 1069–1078.

[41] 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, New York, NY, USA, 2014, pages 1907–1916.

[42] 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 Efficiently with Flexible Resolution, Springer, 2013, pages 373–392.

[43] Q.Ye, F. Wang, B. Li: Starrysky: A practical system to track millions of highprecision query intents. In Proceedings of the 25th International Conference on World Wide Web Companion Volume, WWW'2016, 961–966, Montreal, Canada, April 11–15, 2016, April 2016.

[44] Q. Ye, F. Wang, and et al. Enhanced query classification with millions of fine-grained topics. In Proceedings of the 17th International Conference on Web-Age Information Management, Nanchang, China, June 3–5 2016, pages 120–131.

[45] W.-T. Yih and C. Meek. Improving similarity measures for short segments of text. In Proceedings of the 22Nd National Conference on Artificial Intelligence–Volume 2, AAAI Press, 2007, pages 1489 – -1494.

[46] H.-F. Yu, C.-H. Hoy, and et al. Product title classification versus text classification. Technical report, Department of Computer Science, The University of Texas at Austin., 2012. http://www.csie.ntu.edu.tw/ cjlin/papers/title.pdf.

[47] G.-X. Yuan, C.-H. Ho, and C.-J. Lin. Recent advances of large-scale linear classification. In Proceedings of the IEEE, 100(9)(2012), pages 2584–2603.