queryCategorizr: A Large-Scale Semi-Supervised System for Categorization of Web Search Queries

Mihajlo Grbovic†, Nemanja Djuric†, Vladan Radosavljevic†,
Narayan Bhamidipati†, Jordan Hawker‡, Caleb Johnson‡
†Yahoo Labs ‡Yahoo, Inc.
{mihajlo, nemanja, vladan, narayanb, hawkerj, calebj}@yahoo-inc.com
701 First Avenue, Sunnyvale, CA, USA

ABSTRACT

Understanding interests expressed through user's search query is a task of critical importance for many internet applications. To help identify user interests, web engines commonly utilize classification of queries into one or more predefined interest categories. However, majority of the queries are noisy short texts, making accurate classification a challenging task. In this demonstration, we present *queryCategorizr*, a novel semi-supervised learning system that embeds queries into low-dimensional vector space using a neural language model applied on search log sessions, and classifies them into general interest categories while relying on a small set of labeled queries. Empirical results on large-scale data show that queryCategorizr outperforms the current state-of-the-art approaches. In addition, we describe a Graphical User Interface (GUI) that allows users to query the system and explore classification results in an interactive manner.

Keywords

Query categorization; word2vec; query embeddings.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis; Language parsing and understanding*

1. INTRODUCTION

Search engines are used by billions of online users every day as a tool to promptly find desired information. In order to capture and more easily act upon very clear intent that the users channel through queries, query classification is a task of critical importance to search engines. Here, one aims to classify textual queries into one or more predefined interest categories, such as "finance", "sports", or "technology". This allows search companies to better understand user intentions, improve user experience through better personalization, provide more relevant search results and targeted recommendations, and optimize organization of content. However, query categorization poses difficult, quite different challenges than standard text classification, as queries are usually very noisy and short, with majority containing less than three words [1]. In addition, large number of possible queries induces very sparse feature space for bag-of-words (BOW) encoding schemes, which limits applicability of standard BOW-based approaches. To reduce sparsity, some authors proposed to expand the feature space using features computed from documents retrieved when issuing a query [1]. However, this introduces additional latency, which often represents an unacceptable overhead.

To address these issues we take a radically new approach to query categorization by using distributed language models, motivated by their recent success in a number of natural language processing (NLP) tasks [3, 4]. In the context of NLP, distributed models are able to learn word representations in a low-dimensional continuous vector space using a surrounding word context in a sentence, such that semantically similar words are close to each other in the embedding space [3]. However, directly applying these models to the problem of query categorization is a challenging task. Finding distributed representations of queries, as opposed to words, brings unique challenges unlike those found in everyday NLP problems. Contrary to everyday language, where words, sentences, and language rules are clearly defined, in search queries there is no notion of "a sentence of queries" or the surrounding context, that would be equivalent to a natural language domain for which the distributed language models were developed. We address these issues and propose a novel method that exploits the fact that user queries are recorded with the timestamp in search logs, from which we create "query sentences" using logged user sessions. This allowed us to apply language models and learn query representations in a low-dimensional space where semantically similar queries are nearby. As a result, and in contrast to BOW approaches, related queries have a high similarity score even if they do not have any common terms.

Abovementioned state-of-the-art language models are unsupervised, and to use their strengths in the query classification task we propose their semi-supervised extension. Semi-supervised methods [6] are typically used when majority of the training data, in our setting queries, are unlabeled. Such methods aim at achieving high accuracy on the labeled data and ensuring some statistical dependence on unlabeled examples. The proposed algorithm can be placed under such framework. Using a limited number of labeled queries and large amount of unlabeled data, the method learns "category vectors" in the same space as queries, which simplifies the classification to a nearest-neighbor search in the joint space.

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2742850.

In this demonstration we also describe implementation steps of the deployed query categorization system, called *queryCategorizr*. These include details about data acquisition, representation learning, and classification. We also present a graphical user interface (GUI) which enables service that takes a query as input and outputs related query categories. In addition, the service can retrieve search queries semantically similar to the input query. Our key contributions are summarized below:

- We provide the first application of distributed language models to semi-supervised query categorization, where we propose to learn distributed query and category representations in a joint space that compactly captures their semantic information;
- We trained the model using more than 12 billion search sessions collected on Yahoo servers, resulting in highly effective classification method. The queryCategorizr system achieves 86% precision with high recall;
- We provide user interface coupled with sophisticated, cutting-edge back-end technology to demonstrate effectiveness of our approach.

2. PROBLEM SETUP

Let us assume we are given search logs of N users during a period of T months, comprising Q unique queries. In search logs, every query q is recorded along with its timestamp t. For each user u_i we collect data in the form $\mathcal{D}_i = \{(q_i, t_i), i = 1, \ldots, N_i, t_1 < t_2 < \ldots < t_{N_i}\}$, where N_i represents number of queries that user u_i generated. Given data set $\mathcal{D} = \cup_{i=1}^{N} \mathcal{D}_i$, the objective is to find representation of queries that appeared in the data, such that semantically similar queries are nearby in the embedding space.

We consider the task of query classification, where we aim to classify queries into a pre-defined category taxonomy. In our work, we leverage top 2 levels of an in-house interest taxonomy, amounting to 150 categories. These categories cover a variety of topics, such as "sports/baseball', "finance/mortgage", or "entertainment/movies". To map the queries into one or more interest categories, we propose to learn query and category representations in a joint, low-dimensional space using semi-supervised neural language models applied to historical search logs.

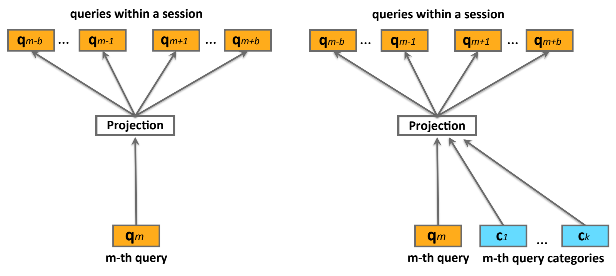
3. BACK-END SYSTEM

In this section we discuss the queryCategorizr system, and describe server-side components that learn query representations from data set $\mathcal{D}$ and classify queries into the taxonomy.

3.1 Data sessionization

The user log data set $\mathcal{D}$ was sessionized and processed into session data set $\mathcal{S}$ containing S search sessions. A search session is defined as an uninterrupted sequences of web search activity, where a session ends when a user was inactive for more than 30 minutes [2].

More specifically, and without loss of generality, session $s = (q_1, \ldots, q_M) \in \mathcal{S}$ is an uninterrupted sequence of M queries (analogous to a sentence in NLP), and each query q consists of L words, $q = (w_1, \ldots, w_L)$. In the case of repetitive queries, e.g., $s = (q_1, q_2, q_3 = q_2, q_4 = q_2, q_5)$, repetitions were de-duplicated, resulting in $s = (q_1, q_2, q_5)$. Lastly, sessions containing only one query were discarded.



(a) Unsupervised (b) Semi-supervised

Figure 1: Skip-gram model variants

3.2 Model training

We assume $\mathcal{D}$ is processed into session data $\mathcal{S} = \cup_{s=1}^{S} \mathcal{S}_s$. Given $\mathcal{S}_s = \{(q_i, t_i), i = 1 \ldots N_i, t_1 < \ldots < t_{N_s}\}$, where N_s represents number of queries in session s, the objective is to find query representation such that semantically similar queries are nearby in the feature space. For this purpose we extend ideas originating from recently proposed language models, as described in the remainder of the section.

The skip-gram model [3] learns representations of queries in a low-dimensional space in an unsupervised fashion, using a query session as a "sentence" and the queries within as "words", to borrow the terminology from NLP domain. The representations are learned by maximizing the objective function $\mathcal{L}$ over the entire set of sessions $\mathcal{S}$, as

$$\mathcal{L} = \sum_{s \in \mathcal{S}} \sum_{q_m \in s} \sum_{-b \leq i \leq b, i \neq 0} \log \mathbb{P}(q_{m+i} | q_m). \tag{3.1}$$

Probability $\mathbb{P}(q_{m+i}|q_m)$ of observing a neighboring query q_{m+i} given current query q_m is defined using the soft-max,

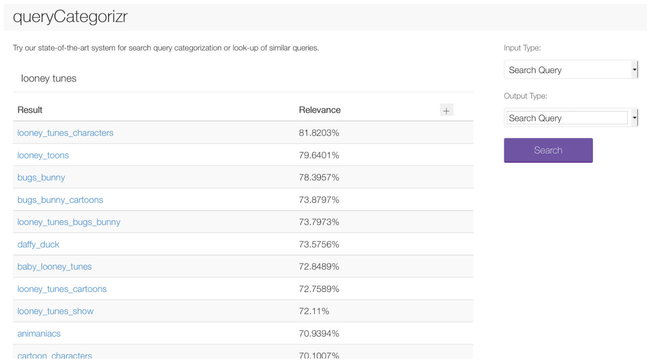
$$\mathbb{P}(q_{m+i}|q_m) = \frac{\exp(\mathbf{v}_{q_m}^\top \mathbf{v}'_{q_{m+i}})}{\sum_{q=1}^{Q} \exp(\mathbf{v}_{q_m}^\top \mathbf{v}'_q)}, \tag{3.2}$$

where $\mathbf{v}_q$ and $\mathbf{v}'_q$ are the input and output vector representations of query q of dimensionality d, and b defines the length of the context for query sequences.

As illustrated in Figure 1a and equation (3.2), skip-gram uses central query q_m to predict b queries that come before and b queries that come after it in the search session. Thus, queries that often co-occur and queries with similar contexts (i.e., with similar neighboring queries) will have similar representations as learned by the model.

Semi-supervised skip-gram (SS-SG) assumes that some queries from the training data $\mathcal{D}$ are labeled with categories from the taxonomy. Then, we assign a vector to each category, and leverage query contexts in sessions to jointly learn query vectors and category vectors in the same feature space. To this end, given the labeled queries, we extend $\mathcal{D}_s$ to obtain data set $\mathcal{D}_l$ where categories were imputed into sessions. In particular, labeled queries were accompanied by assigned categories, and every time a vector of labeled central query q_m is updated to predict the surrounding queries, vectors of categories assigned to q_m are updated as well.

More formally, assuming central query q_m is labeled with C_m of C categories in total, $\zeta_m = \{c_1, \ldots, c_{C_m}\}$, the semi-supervised skip-gram learns query and category representa-



queryCategorizr

Try our state-of-the-art system for search query categorization or look-up of similar queries.

looney tunes

Result	Relevance
looney_tunes_characters	81.8203%
looney_toons	79.6401%
bugs_bunny	78.3957%
bugs_bunny_cartoons	73.8797%
looney_tunes_bugs_bunny	73.7973%
daffy_duck	73.5756%
baby_looney_tunes	72.8489%
looney_tunes_cartoons	72.7589%
looney_tunes_show	72.11%
animaniacs	70.9394%
cartoon_characters	70.1007%

Input Type: Search Query

Output Type: Search Query

Figure 2: Nearest neighbors for query "looney tunes"

tions by maximizing the following objective function $\mathcal{L}$,

$$\sum_{s\in\mathcal{S}}\sum_{q_m\in s}\sum_{-b\le i\le b,i\neq 0}\Big(\log\mathbb{P}(q_{m+i}|q_m)+\sum_{c\in\zeta_m}\log\mathbb{P}(q_{m+i}|c)\Big). \tag{3.3}$$

Probability $\mathbb{P}(q_{m+i}|c)$ of observing query q_{m+i} given label c of the current query q_m is defined using the soft-max,

$$\mathbb{P}(q_{m+i}|c)=\frac{\exp(\mathbf{v}_c^\top\mathbf{v}'_{q_{m+i}})}{\sum_{q=1}^{Q}\exp(\mathbf{v}_c^\top\mathbf{v}'_q)}. \tag{3.4}$$

The SS-SG model is illustrated in Figure 1b.

3.3 Classification

In order to classify an unlabeled query q_u we perform the following operations: 1) lookup vector representation of q_u; 2) lookup vectors for all C categories; 3) calculate cosine similarity between category vectors and query q_u, and label the query with the category that has the highest similarity.

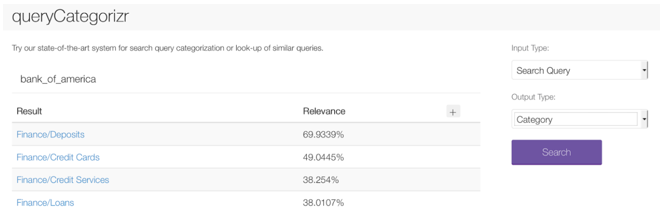
Handling out-of-vocabulary queries. To classify a query not seen during training, we segment it using Conditional Random Field [5]. Then, its representation was obtained as a sum of vectors which correspond to the segments.

4. FRONT-END SYSTEM

To build GUI for the queryCategorizr system, we considered modern web application standards and employed a browser-heavy approach with HTML/CSS/JavaScript code, as well as a light API layer in Python.

4.1 Architecture

Cutting-edge front-end technologies were employed in development of the interface, enabling the creation of a fast, robust web application. The project uses Node[1] as an I/O serving layer with Ember-CLI[2] (powered by Broccoli[3]), providing a lightning-fast asset pipeline to support constant-time rebuilds and compact build definitions. Ember-CLI also allowed us to write next-generation JavaScript code compliant with upcoming ECMAScript 6 language specifications[4] by enabling a built-in transpiler[5] to convert ES6 syntax to ES5.1-compliant AMD (RequireJS[6]) modules. NPM[7] and Bower[8] package managers were utilized in concert to bring in external library dependencies. Lastly, JQuery[9] was used to perform AJAX requests to retrieve data from the SS-SG model via the API layer written in Python.



queryCategorizr

Try our state-of-the-art system for search query categorization or look-up of similar queries.

bank_of_america

Result	Relevance
Finance/Deposits	69.9339%
Finance/Credit Cards	49.0445%
Finance/Credit Services	38.254%
Finance/Loans	38.0107%

Input Type: Search Query

Output Type: Category

Figure 3: Categorization of input query "bank of america"

4.2 Implementation

Our choice of front-end technologies was derived from the goal of creating a Single Page Application (SPA) to handle business logic effectively on the client-side of the system, without the need to pass page assets back and forth to the server-side of the application after the initial page load. EmberJS[10] serves as the primary backbone of the project, providing an opinionated MVC framework with hierarchical routing and two-way data binding to the DOM. We utilized Ember-idiomatic UI components, such as tables and selects, that bring feature-rich client interaction to these classic web elements. Bootstrap[11] provided responsive layouts, as well as stylized inputs and buttons to give the application a clean, aesthetically-pleasing interface.

4.3 Design

Primary design choices for user interface revolved around data exploration. The UI of queryCategorizr is shown in Figures 2 and 3. It is composed of three major components: a query input box, results box, and a drop-down menus to select different functionalities. The demo works by typing in queries in the query input box. Depending on the selected functionality, searching for a query in the application reveals 1) the list of highly relevant queries identified by the SS-SG model; or 2) interest categories for a given query. Results are updated immediately, together with relevance scores, as a user types in a query or clicks on one from the results list. This allows the user to view results as quickly as possible with near-instant feedback on their queries. We also created a user experience that allows data relations to be investigated on an iterative basis. By clicking on retrieved queries/interests from the results table, the user can easily evolve their initial query into a series of relevant inquiries.

5. EVALUATION

We learned query representations using more than 12 billion search sessions extracted from search logs collected on Yahoo servers. For the purposes of SS-SG training a limited portion of queries were manually labeled by human editors, resulting in approximately 2,000 labeled queries per category from the interest taxonomy.

The semi-supervised skip-gram model was optimized using stochastic gradient ascent, suitable for large-scale prob-

[1]http://nodejs.org/, all URLs last accessed in January 2015
[2]http://www.ember-cli.com/
[3]https://github.com/broccolijs/broccoli
[4]https://people.mozilla.org/ jorendorff/es6-draft.html
[5]https://github.com/esnext/es6-module-transpiler
[6]http://requirejs.org/
[7]https://www.npmjs.com/
[8]http://bower.io/
[9]http://jquery.com/
[10]http://emberjs.com/
[11]http://getbootstrap.com/

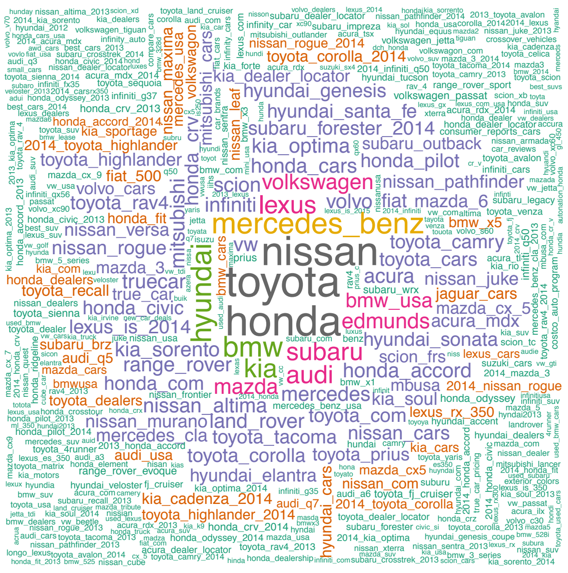


Figure 4: Queries categorized into "automotive" category

lems. However, computation of gradients $\nabla\mathcal{L}$ in (3.1) and (3.3) is proportional to the vocabulary size Q, which is computationally expensive for our application as Q could easily reach tens of millions queries. As an alternative, we used negative sampling approach proposed in [3], which significantly reduces the computational complexity of the training.

Vector representations were trained for 60 million most frequent queries found in the search logs. Training was done using a machine with 256GB of RAM memory and 24 cores. Dimensionality of the embedding space was set to $D = 300$, while context neighborhood size was set to 5. Finally, we used 10 negative samples in each vector update.

In the first experiment we verified that the semi-supervised algorithm maps semantically similar queries close to each other in the embedding space, and also close to the best-matching categories. This is illustrated in Figure 4, where we show nearest queries to the vector of "automotive" category (size of a query in the word-cloud is proportional to the cosine similarity). We can see that SS-SG grouped semantically related queries into the same part of the space, while keeping them close to the appropriate category vector.

To better quantify the value of our approach, we evaluated the proposed semi-supervised algorithm on the labeled query set using 5-fold cross-validation. We compared semi-supervised-based classification to the logistic regression (LR) and linear support vector machine (SVM) classifiers that use BOW features. We also compared to LR and SVM where we used features learned by unsupervised SG to represent the labeled queries. We report the results in Table 1, where we can see that classification using the SS-SG method achieved higher precision than the competing methods, while at the same time maintaining competitive recall measure.

6. DEMONSTRATION

Our demonstration system consists of three critical components: 1) front-end user interface; 2) back-end service that processes all user requests; and 3) back-end server that performs classification and query similarity calculations. During the demonstration, we will showcase two functionalities mentioned previously: 1) finding semantically similar queries; and 2) classifying queries into interest categories. We will first go over the set of queries prepared in advance to demonstrate strengths and weaknesses of the queryCategorizr system. Then, we will ask users to test capabilities of queryCategorizr using their own queries[12].

Table 1: Precision and recall of different methods

Method	Precision	Recall
LR-BOW	0.71	**0.66**
SVM-BOW	0.74	0.65
LR-SG	0.80	0.64
SVM-SG	0.82	0.62
SS-SG	**0.86**	0.63

7. CONCLUSION

In this paper we described queryCategorizr, a state-of-the-art system for understanding of user search queries. The system includes functionalities such as finding semantically similar queries and categorizing queries into interest categories, useful in a number of online applications. We demonstrated the benefits of the system via graphical user interface in which users freely explors system capabilities in an intuitive manner. In the future, we plan to expand the system by incorporating embedded representations of other user events, such as clicked ads or read articles.

8. REFERENCES

[1] E. Gabrilovich, A. Broder, M. Fontoura, A. Joshi, V. Josifovski, L. Riedel, and T. Zhang. Classifying search queries using the web as a source of knowledge. *ACM Transactions on the Web*, 3(2):1–28, April 2009.

[2] D. Gayo-Avello. A survey on session detection methods in query logs and a proposal for future evaluation. *Inf. Sci.*, 179(12):1822–1843, May 2009.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[4] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[5] X. Yu and H. Shi. Query segmentation using conditional random fields. In *Proceedings of the First International Workshop on Keyword Search on Structured Data*, KEYS '09, pages 21–26, 2009.

[6] K. Zhang, J. T. Kwok, and B. Parvin. Prototype vector machine for large scale semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1233–1240. ACM, 2009.

[12]Demonstration video is available online at the following URL: http://youtu.be/RSQ7mK-xpH8