Graph Learning for Exploratory Query Suggestions in an Instant Search System

Enrico Palumbo[1], Andreas Damianou[2], Alice Wang[3], Alva Liu[4], Ghazal Fazelnia[5], Francesco Fabbri[6], Rui Ferreira[7], Fabrizio Silvestri[8,15], Hugues Bouchard[9], Claudia Hauff[10], Mounia Lalmas[11], Ben Carterette[12], Praveen Chandar[13], David Nyhan[14]

Spotify

[1]Italy, [2]UK, [3]USA, [4]Sweden, [5]USA, [6]Spain, [7]UK, [8]Italy, [9]Spain, [10]Netherlands, [11]UK, [12]USA, [13]USA, [14]USA.

{enricop,andreasd,alicew,alval,ghazalf,francescof,ruif,fabrizios,hb,claudiah,mounial,benjaminc,praveenr,davidn}@spotify.com

[15]Sapienza University of Rome, Italy

ABSTRACT

Search systems in online content platforms are typically biased toward a minority of highly consumed items, reflecting the most common user behavior of navigating toward content that is already familiar and popular. Query suggestions are a powerful tool to support query formulation and to encourage exploratory search and content discovery. However, classic approaches for query suggestions typically rely either on semantic similarity, which lacks diversity and does not reflect user searching behavior, or on a collaborative similarity measure mined from search logs, which suffers from data sparsity and is biased by highly popular queries. In this work, we argue that the task of query suggestion can be modelled as a link prediction task on a heterogeneous graph including queries and documents, enabling Graph Learning methods to effectively generate query suggestions encompassing both semantic and collaborative information. We perform an offline evaluation on an internal Spotify dataset of search logs and on two public datasets, showing that *node2vec* leads to an accurate and diversified set of results, especially on the large scale real-world data. We then describe the implementation in an instant search scenario and discuss a set of additional challenges tied to the specific production environment. Finally, we report the results of a large scale A/B test involving millions of users and prove that *node2vec* query suggestions lead to an increase in online metrics such as coverage (+1.42% shown search results pages with suggestions) and engagement (+1.21% clicks), with a specifically notable boost in the number of clicks on exploratory search queries (+9.37%).

CCS CONCEPTS

• **Information systems → Query suggestion**.

KEYWORDS

graph learning, query suggestions, exploratory search, spotify

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00

https://doi.org/10.1145/3583780.3615481

ACM Reference Format:

Enrico Palumbo[1], Andreas Damianou[2], Alice Wang[3], Alva Liu[4], Ghazal Fazelnia[5], Francesco Fabbri[6], Rui Ferreira[7], Fabrizio Silvestri[8,15], Hugues Bouchard[9], Claudia Hauff[10], Mounia Lalmas[11], Ben Carterette[12], Praveen Chandar[13], David Nyhan[14] . 2023. Graph Learning for Exploratory Query Suggestions in an Instant Search System. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3583780.3615481

1 INTRODUCTION

In online streaming platforms, search systems play a vital role in helping users navigate to the relevant content in a quick and efficient manner, and prior research has identified that users primarily have a *focused* and *exploratory* mindset [18]. When users have an *exploratory* search mindset, meaning that they do not have a specific item in mind, they spend more time searching and are more willing to engage with new content, leading to discovery [38]. The exploratory mindset supports users in serendipitous discoveries, promotes under-served content, and exposes small creators to a wider audience [37]. For instance, if a user has a focused intent such as "stairway to heaven", the search system will necessarily retrieve the Led Zeppelin song, whereas with an exploratory intent such as "new funk soul albums" more diverse and niche content can be presented in the SERP (Search Engine Results Page). Exploratory search provides an opportunity for online content platforms to diversify user's interests and promote the visibility of small creators and help users discover new content. Query suggestion features such as *Related Searches* are a fundamental component in modern search engines, helping users formulate and re-formulate queries. This is beneficial for different use cases, and is particularly useful in exploratory search [11, 38], where the discovery process often requires several iterations of query formulation. For example, imagine that the user is looking for music to stream during a yoga session. She initially types a generic 'yoga' query, but after seeing the query suggestions, she opts for 'vinyasa' and finally refines her intent with 'vinyasa flow'. Exploratory searches particularly benefit from high diversity in the set of query suggestions as this allows users to explore different semantic paths and levels of specificity compared to the original information need [1].

In this paper, we show that graph learning methods can be used to generate query suggestions that are highly accurate and diversified, effectively supporting exploratory search, by performing an extensive offline evaluation against collaborative, content-based,

and graph-based baselines and an A/B test involving millions of users. Our main contributions are the following:

(1) We propose to model the query suggestion task as a link prediction problem on a heterogeneous graph composed of query shortcuts and documents and to approach it with a graph learning method (*node2vec* [13]) that learns vector representations of queries and documents.
(2) We conduct an extensive offline evaluation on a large-scale internal dataset collected from Spotify, comparing *node2vec* to purely semantic, collaborative, or traditional graph-based approaches. We also experiment on two public datasets (MS-MARCO [26], AOL [31]) for reproducibility. Our results show that graph learning methods obtain high accuracy in predicting query suggestions, while preserving a high level of diversity in the set, especially in the internal dataset, which most closely mimics the production scenario.
(3) We describe a practical implementation of the proposed approach in a real-world instant search system, addressing specific challenges such as removing incomplete queries and combining with existing sources of query suggestions. We discuss the results of an A/B test conducted on millions of users where adding the graph learning-based query suggestions increased 1) the overall engagement with the query suggestions (+1.21%), and 2) the number of clicks on exploratory search queries (+9.37%).

2 RELATED WORK

Exploratory Search. Exploratory search has been widely studied in Information Retrieval (IR) [1, 23, 27, 38]. Although an unambiguous and unique definition of exploratory search does not exist, many studies highlight the difference in user goals in lookup and exploratory intents, where the former is generally associated with fact retrieval, known-item search and the latter has a strong association with learning and discovery [38]. Another difference is the dynamic nature of exploratory search, which starts with a broad information need that gradually gets refined by interacting and evaluating the search results and re-formulating queries. For this reason, query suggestions are particularly useful for exploratory search [11].

Query Suggestions. The query suggestion problem is typically approached in two ways [25]. The first strategy assumes that users optimize their queries during a search session, turning the problem into a next-query prediction problem. Models that follow this strategy rely only on query-query correlations mined from search logs, either through traditional statistical approaches [2, 17] or using modern language technologies [25, 39]. However, search sessions are typically noisy and there is no guarantee that this approach leads to a query that is successful, in terms of item clicks or content consumption. The second approach focuses on suggesting queries which lead to a successful interaction, and can in principle be applied also in absence of query logs [6]. A specific formulation of this second strategy that combines the use of query logs with the notion of successful queries is *query shortcuts* [4], i.e. a link between a query and the last observed successful query of a search session. In our work, we follow the query-shortcut approach (cf. Sec. 3), as we know when a query is successful based on the user interaction patterns and we can use this information to lead users toward successful queries.

Graph-based Methods for Query Suggestions. Thanks to their ability to model heterogeneous information, graph-based methods have been used in the past for the query suggestion task, leveraging query-query and query-URL interactions from query logs [3, 5, 7, 8, 14]. However, these methods generally suffer from data sparsity and do not take advantage of the recent advances in graph representation learning. Models like DeepWalk [32] and *node2vec* [13] perform random walks on the graph, creating sequences of nodes, and then apply word embedding learning models (e.g. Word2Vec [24]) to learn embeddings of the nodes. This approach is highly scalable and efficient, and the flexibility in defining the random walk strategy allows to effectively model different graph topologies [13]. Similar techniques have successfully been applied to recommendation tasks [30] and to search tasks, for instance by modelling search sessions [19] or predict query suggestions [16]. Graph Neural Networks (GNNs) [40] are even more powerful tools, enabling generic aggregation and representation learning functions that work inductively and generalize to new nodes. However, the online serving and productionization of inductive GNNs on large-scale graphs remains a challenging endeavor [20]. In our work we focus on transductive approaches based on shallow neural networks that have a relatively reduced cost such as *node2vec*.

3 APPROACH

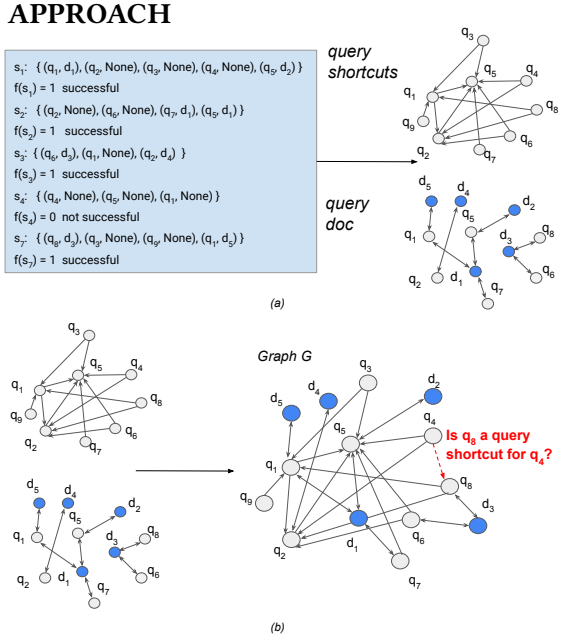


Figure 1: Query suggestion as a query shortcut link prediction task. (a) search sessions are turned into query shortcuts and query doc edges (b) heterogeneous graph is created and the problem of query suggestion is modelled as a link prediction task for the query-shortcut edges.

We start from the problem formulation proposed in [4], where the concept of *query shortcut* is introduced. The intuition behind

the query shortcut problem is the following. Consider an example where many users search for q1 → q2 → q3 → q4 until they find the information they wanted by issuing query q4 and then stop searching. We assume that q4 is an effective formulation of the user's intent and that whenever we encounter q1, q2, or q3, we can recommend q4 as a potential *shortcut* to that information need. For instance, a search session such as 'new outfit 2022' → '80s vintage jeans' → '80s vintage shirt' where the last query '80s vintage shirt' leads to a successful interaction, would generate a set of query shortcuts 'new outfit 2022' → '80s vintage shirt', '80s vintage jeans' → '80s vintage shirt'. This approach naturally lends itself to collaborative filtering strategies, where the 'wisdom of the crowds' can be leveraged to recommend effective queries, capturing trends that are hard to model with a purely semantic approach. However, we can also see it as a link prediction problem on a graph, with the advantage that in this way we can easily encompass additional information (e.g. query-document interactions or document metadata) to support the prediction task. For instance, a set of queries that lead to the same document d='https://en.wikipedia.org/wiki/Napoleon' such as 'napoleon' → d, 'napoleonic wars' → d, 'waterloo' → d help model the fact that 'napoleon', 'napoleonic wars' and 'waterloo' are semantically similar queries.

3.1 Query shortcut prediction as a link prediction problem

We now explain how we construct the graph; we model the problem as query shortcut prediction on a graph (Fig. 1).

Definition 3.1 (search sessions). Given a space of queries $Q = \{q_1, q_2, ..., q_n\}$ and documents $D = \{d_1, d_2, ..., d_l\}$, we assume that we have a set of search sessions $S = \{s_1, s_2, ..., s_j\}$ where each $s_i = \{q_i, d_i\}$ is a tuple of a query and its corresponding document interaction (d_i is 'None' if there was no interaction for q_i).

Definition 3.2 (success function). $f(s)$ is a success function, i.e. a known boolean function such that $f(s) = 1$ if and only if the session $s = \{(q_1, d_1), ..., (q_n, d_n)\}$ is successful. The exact shape of $f(s)$ depends on the dataset at hand (see Section 4.1).

Definition 3.3 (graph). $G = (N, E)$ is a heterogeneous graph where the set of nodes $N = Q \cup D$ includes all queries and documents appearing in the set of sessions S. The set of edges $E = E_{shortcuts} \cup E_{docs}$ includes query-shortcut and query-doc edges.

Definition 3.4 (query shortcuts). Given a successful session s_j, query-shortcuts $E_{shortcuts} = \{(q_1, q_n), (q_2, q_n), ..., (q_{n-1}, q_n)\}$ are sets of tuples connecting each query with the last successful query within s_j. These edges reflect purely searching behavior and are the 'collaborative' information in the graph.

Definition 3.5 (query-doc). Given a successful session s_j, query-docs $E_{docs} = \{q_1, d_1), ..., (q_n, d_n)\}$ are sets of tuples connecting a query with a relevant doc in the session. These edges connect a query to a relevant document help model the semantics of the queries (similar queries lead to the same documents) and represent the 'content-based' information in the graph.

The query recommendation problem can now be modeled as a link prediction problem on the query-shortcut edges of the graph G. Hence, the models need to learn a scoring function $\rho(q_i, q_j) \rightarrow \mathbb{R}$ that grows monotonically with the probability of q_j being a query shortcut for q_i. The scores of ρ can then be used to rank a set of candidate queries $C = \{q_1, q_2, ..., q_M\}$, obtaining a sorted list C_{sorted} where $\rho(q_{i+1}) < \rho(q_i)$ for $i = 1...M$. The first k elements of C_{sorted} are then selected as query suggestions as in a classic top-k recommendation task.

Graph Learning for Query Suggestions. Given the above definitions, the models learn node vectors that will be used to predict if a link is present between a pair of nodes. Graph learning allows to learn a function that maps the space of graph nodes N to a m dimensional vector space $L : N \rightarrow \mathbb{R}^d$. In the vector space, we can then create a scoring function $\rho(q_i, q_j)$ using a similarity function $g(x_i, x_j)$, where x_i is the associated learned vector of the query q_i. In this work, we focus on *node2vec* [13], which has proven to be very effective for recommendation tasks on heterogeneous graphs [30], while being scalable and relatively simple to productionize using a vector nearest neighbor search index (see Sec. 6). *node2vec* works by: 1) creating sequences of nodes by performing random walks on the graph G, and, 2) learning node vectors x_i from the sequences using a neural network with a skip-gram architecture [24].

In this paper, we investigate the effectiveness of our approach by answering the following research questions:

RQ1: Does graph learning produce accurate query suggestions, while maintaining a high level of diversity?
RQ2: Is graph learning robust to popularity bias?
RQ3: What is the impact of query-document interactions on the effectiveness of graph learning model?
RQ4: What is the most suitable model for deployment?
RQ5: Do graph learning solutions encourage exploratory search in a large-scale A/B test?

4 EXPERIMENTAL SETUP

In this section, we describe the datasets, metrics, models, and the protocol that we used for the offline evaluation.

4.1 Datasets

We create three graphs for our experiments starting from three datasets (two public and an internal one). Their basic characteristics are summarized in Table 1.

`Internal`. We collect a sample of 100k search sessions from Spotify, an online music streaming platform where users search for audio content such as music or podcasts. For this dataset, we can determine accurately if a search session was successful using a series of follow-up actions to the click (e.g. a stream or save to the user's library), which defines the $f(s)$ function. Query-doc edges correspond to interactions with catalog items such as playlists, tracks, artists or shows.

`MS-MARCO`. `MS-MARCO` [26] is a popular set of large datasets in the IR community and contains 1M queries generated by sampling and anonymizing Bing search sessions[1] and query-doc relevance judgments.[2] In absence of a real search success signal, we use

[1] https://github.com/microsoft/MSMARCO-Conversational-Search, we use 'ms_marco_specify' as it provides the highest semantic coherence
[2] Document ranking task:https://microsoft.github.io/msmarco/TREC-Deep-Learning-2020

Table 1: Datasets summary. l = avg number of characters in query, qs = % of query-shortcut edges, k = average node degree, pop_{bias} = weight of top-10 shortcuts

	MS-MARCO	AOL	Internal
$\|S\|$	57k	74k	105k
$f(s)$	relevant doc	clicked doc	follow-up actions
l	28	13	5
$\|N\|$	596k	105K	100k
$\|E\|$	613k	337K	1.04M
$\|E_{shortcuts}\|$	156k	120k	21k
$\|E_{docs}\|$	456k	127k	1.02M
qs	25%	35%	2%
k	5.1	6.4	21.1
pop_{bias}	1.47%	6.39%	1.25%
Query example	*define width*	*weather channel*	*yoga meditat*

the 'is_selected' flag that signals the most relevant document in the relevance judgement as a proxy to determine if a session was successful or not: $f(s) = 1$ if and only if the last query of the session q_n is associated with a document d_n that has 'is_selected = 1'.

AOL. AOL [31] is a dataset containing anonymized search query logs[3] from 500k users. It provides a set of tuples containing *user id*, *query text*, *click url*, *query time*. We create search sessions by splitting the search activity of a user when the time interval between two interactions is larger than 30 minutes. We consider the search session to be successful if the last query is associated with a click.

For all datasets, we split the query-shortcut edges (i.e. the target of the prediction models) into training, validation, and test (80%-10%-10% of sessions) making sure that the same session appears in only one split to avoid data leakage. When we build the graph, we deduplicate query shortcuts. Although we could take the frequency of a shortcut into account by weighting edges, in this work we focus on evaluating the ability of the models at generalizing and predicting new shortcuts in the test set, rather than memorizing existing frequent shortcuts observed in the training set.

4.2 Evaluation protocol

Given the problem formulation described in Section 3.1, we can leverage well-established evaluation protocols for top-K item recommendation, with the analogy that query shortcuts represent our 'user-item' rating matrix [36]. Specifically, we use the *AllItems* [36] ranking evaluation protocol, which means that, for each query in the test set, the set of candidate query shortcuts to rank is the set of all queries in the graph. In this evaluation protocol, unobserved query shortcuts are considered as negative examples, implying that ranking accuracy metrics are a lower bound estimation of the real ranking quality (e.g. we might recommend a good query shortcut that was just not present in the data). We opt for this protocol as it is the most realistic and the one that most closely resembles a production scenario.

[3]http://www.cim.mcgill.ca/~dudek/206/Logs/AOL-user-ct-collection/U500k_README.txt

We evaluate two dimensions of quality of query suggestions: Mean Average Precision (MAP) [22] and Word Embedding Diversity (WED) [28]. All metrics are evaluated at $K = 10$, as this is is the most suitable value for the real production system where we deploy this technology. Also, all metrics are computed at the query level first, and then averaged. We test our hypotheses for statistical significance using a paired t-test with Bonferroni correction with $p < 0.01$.

4.3 Models

We focus on *node2vec* [13] as a graph learning model. It learns vector representations of nodes for graphs, introducing a flexible random walk strategy that can be tailored to different connectivity patterns through specific hyper-parameters p and q that govern the trade-off between breadth-first and depth-first search. We optimize the hyper-parameters on the validation sets, finding the configuration $p = 1$, $q = 1$, $d = 65$, $num\_walks = 10$, $window\_size = 20$, $walk\_length = 30$ to be working well across datasets on average.

We compare *node2vec* to models based on different strategies (e.g. collaborative filtering, query logs correlations, semantic similarities): *random*, *Most Popular (MP)* (deterministically returns the top-K most popular query shortcuts), *Personalized Page Rank (PPR)* [8, 15] (traditional random walk based approach), *Sentence BERT (SBERT)* [33] (semantic similarity using pre-trained sentence embeddings), *Log-Likelihood Ratio (LLR)* [17] (statistical correlations between reformulations in query logs), and *Bayesian Personalized Ranking Matrix Factorization (BPRMF)* [34] (a matrix factorization approach using implicit feedback).

5 OFFLINE RESULTS

We report the results of the offline evaluation and discuss the research questions we introduced in Sec. 3.

RQ1: Does graph learning produce accurate query suggestions, while maintaining a high level of diversity? We first measure and compare the ability of the models to predict query shortcuts in terms of ranking accuracy metrics (Fig. 2a). We observe that *node2vec* outperforms all the other models on Internal, whereas the pure collaborative approach BPRMF works better on MS-MARCO and AOL. Looking at Tab. 1, we observe that Internal has a higher average degree ($k = 21.1$) than MS-MARCO ($k = 5.1$) and AOL ($k = 6.4$). The average degree tells us something about the density of the graph, and its level of connectivity, which typically helps the effectiveness of graph learning approaches that rely on modeling neighborhoods using graph topology [12]. In addition, in the two public datasets (AOL and MS-MARCO) the percentage of query-shortcut edges is much higher than for Internal ($qs = 25\%$ in MS-MARCO, $qs = 35\%$ in AOL vs. $qs = 2\%$ in Internal). As introduced before (Sec. 4), query-shortcut interactions can be seen as a user-item feedback matrix, so the fact that BPRMF does not perform as well on Internal can be interpreted in light of the well-known data sparsity issues of collaborative filtering algorithms. The popularity bias is known to skew the evaluation of recommender systems [9] and in our experiments, we observe that BPRMF performs particularly well on AOL ($pop_{bias} = 6.39\%$) and MS-MARCO ($pop_{bias} = 1.47\%$), which have a stronger popularity bias than Internal ($pop_{bias} = 1.25\%$).

We next evaluate the diversity of the suggestions, and report the results in Fig. 2b. We observe that overall purely semantic models such as the sentence encoder naturally tend to produce less diversified suggestions, whereas collaborative filtering models excel at diversifying the set. It is important to notice that a random model achieves a very high diversity, as expected. Hence, diversity needs to be always analyzed together with the ranking accuracy metrics. Graph-based methods obtain a good level of diversity across datasets.

RQ2: Is graph learning robust to popularity bias? We run an ablation study to better understand the effect of popularity bias on the different models (Fig. 2c), i.e. we want to understand if the models are accurate beyond the head of the distribution. We measure *map_no_pop*, which is equivalent to MAP after removing the top 10% of popular shortcuts from the set of relevant query-shortcuts. We observe that this strongly changes the results, and the purely semantic approach of the sentence encoder becomes the best model on the public datasets (`AOL` and `MS-MARCO`), followed by *node2vec*, which is on par with the sentence encoder on `Internal`. This shows that graph-based methods, and in general approaches that take into account the content in addition to user interactions, are much more robust to popularity bias and are generally a better approach when precision beyond the head of the distribution is important as in the case of query suggestions. This is in line with previous research in the recommender systems field [29].

RQ3: What is the impact of query-document interactions on the effectiveness of graph learning models? To probe the importance of adding content-based information such as query-doc edges into the graph, we conduct an ablation study where we train *node2vec* on the query-shortcut edges alone. The results are shown in Fig. 2d. For the `MS-MARCO` and `AOL` datasets where query-shortcut edges are abundant ($qs = 25\%$ and $qs = 35\%$ respectively), the impact of query-doc edges is negligible, whereas the gap is quite significant in the case of the dataset `Internal` where this information is much sparser ($qs = 2\%$).

RQ4: What is the most suitable model for deployment? The `Internal` dataset is the most faithful depiction of our deployment scenario, and it is therefore the ultimate yardstick against which we make the decision of which model we want to deploy and A/B test. *node2vec* is the best model at predicting query shortcuts on `Internal`, obtaining a good level of semantic diversity, and is robust to popularity bias, keeping high accuracy even when popular shortcuts are removed, in contrast to collaborative filtering systems. In addition, it is relatively easy to productionize as suggestions can be served leveraging well-established approximate nearest neighbor look-ups in a vector space and the cost is moderate thanks to the use of a shallow neural net (Word2Vec model) that trains in a reasonable time also on CPUs.

6 ONLINE RESULTS

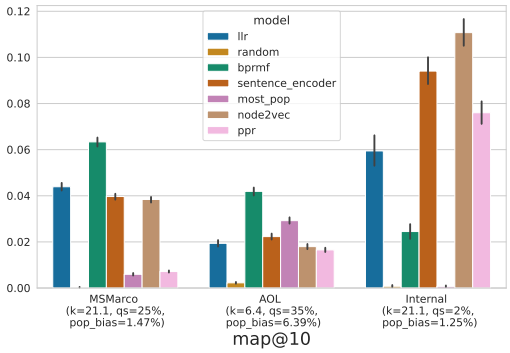
In this section, we describe the high-level implementation of the Related Searches system and the results of the A/B test.

Complete Query Detector. Our search system is based on instant results. Instant search systems display documents on the SERP as the query is being typed, allowing users to quickly retrieve known items with a few keystrokes. For instance, as a user types 'true cr', several true crime podcasts will appear. Users generally stop typing and click on the result as soon as they find what they want, leading to a fast experience, but leading to the majority of queries in the search logs being incomplete. This is an additional challenge for a query suggestion system, as incomplete prefixes such as 'true cr' are not suitable for being recommended to users. We thus develop a classifier that allows us to filter out incomplete queries from the query recommendations. The classifier is based on distilBERT [35], a smaller and faster-distilled version of the popular BERT model [10], which is suitable for fine-tuning on downstream classification tasks. We model the complete query detection as a classification task, where the objective is to classify whether a query is complete or not (e.g. 'best podcasts by' → *incomplete*, 'true cr' → *incomplete*, 'true crime' → *complete*). We create training data for the complete query detector using a weak labelling heuristic: 1) collect pairs of queries and clicked documents (q_i, d_i) from the search logs; 2) extract a set of textual metadata m_{ij} from document d_i such as the title, author, topics or genre; 3) if there is $EXACT\_MATCH(q_i, m_{ij}) = True$ for any j, we consider q_i a *complete* query; and 4) if there is $PREFIX\_MATCH(q_i, m_{ij}) = True$ for any j, we consider q_i as an *incomplete* query. In cases where q_i is annotated as complete and incomplete, depending on the rule used, we sort out the disagreement based on frequency of the assigned class at the end of the process.

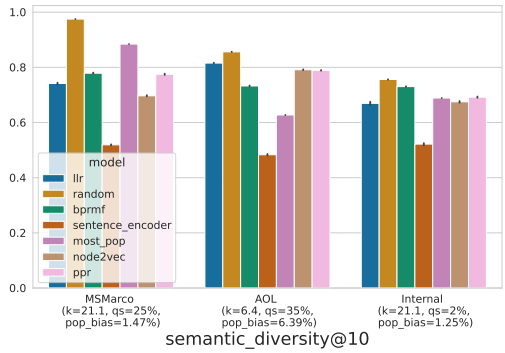
We obtain a balanced training set of $\approx 200k$ queries on which we fine-tune a pre-trained distilBERT model. We then evaluate on a separate small set of real queries ($\approx$ 500 examples), which we manually annotate, and measure an $F1 \approx 85\%$.

Two-stage Multi-source Query Suggestions with node2vec. The Related Searches system works with a two-stage multi-source architecture. In the first stage, multiple sources generate query suggestions based on different logics, allowing us to focus on different aspects of the problem and to experiment with new models with small disruptions to the customer experience. Then, a neural network model acts as a second stage re-ranker, leveraging a number of features based on the query, the suggestion, and the user's personal taste. We deploy the *node2vec* model as an additional source of query suggestions, following the offline results that show a strong performance in terms of ranking accuracy and diversity (see Sec. 5). The *node2vec* model is retrained weekly and produces a set of 65-dimensional embeddings for prefixes and complete queries from our logs. We perform approximate nearest neighbors lookup by using Hierarchical Navigable Small World (HNSW) [21] indices with the embeddings, a source index with the prefix query embeddings and a target index with the complete query suggestion embeddings (i.e. filtering out incomplete queries using the complete query detector). The HNSW files are distributed across multiple regions for faster download. A backend service periodically checks for updates and loads new indices it detects into memory. At request time, the service looks up the embedding for the user's prefix query in the source index, and finds the closest complete query suggestions in the target index to recommend to the user.

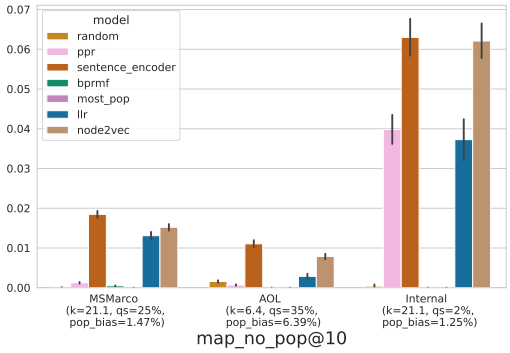
RQ5: Do graph learning solutions encourage exploratory search in a large-scale A/B test? We run an A/B test to measure



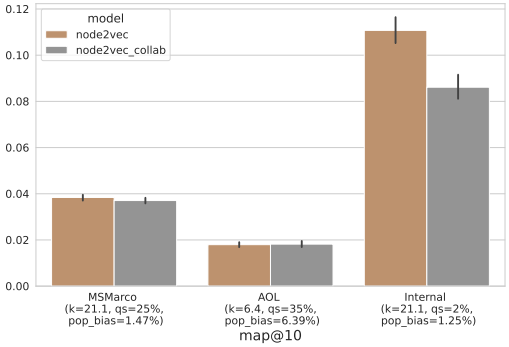
(a) ranking accuracy



(b) diversity



(c) popularity bias



(d) query-doc edge impact

Figure 2: Offline evaluation results across our three datasets.

the online impact of adding the *node2vec* source of query suggestions into the system, comparing it with the control variant of Related Searches (i.e. all sources but *node2vec*). We target $\approx 6M$ users per treatment for 2 weeks and measure overall engagement (i.e. number of clicks on Related Searches), coverage (i.e. number of SERPs for which we can generate query suggestions), and exploratory engagement (i.e. number of clicks on Related Searches on exploratory queries). We rely on an internal classification of queries to determine whether a query is exploratory or not, which is based on several interaction signals such as the item type (e.g. playlists are more associated to exploratory intents than tracks). The results show that adding the graph learning query suggestion source improves coverage (+1.42%), engagement (+1.21%), with a specifically significant boost in the number of clicks on exploratory search queries (+9.37%). All the relative increases are statistically significant with a confidence level $\alpha = 0.01$. No impact on latency metrics was observed, confirming the efficiency of the *node2vec* solution for a production scenario.

7 CONCLUSIONS

In this work, we show that the problem of query suggestion can be modelled as a link prediction problem on a graph, where the task is to predict a query shortcut edge observed in the search logs. Through an extensive offline evaluation, we demonstrate that this leads to high accuracy and diversity, especially on the `Internal` dataset, addressing the shortcomings of pure collaborative filtering, which suffers from popularity bias and data sparsity, and of purely semantic approaches, which lack diversity. Following these observations, we report the findings of an A/B test targeting millions of users where the *node2vec* query suggestion source is added into a large-scale query suggestion system, improving coverage and clicks, especially among the exploratory query suggestions, to improve discovery and catalog usage. One of the limitations of *node2vec* is that it is a transductive approach, and thus its coverage is limited to queries observed in the search logs. In the future, we plan to improve this aspect by experimenting with inductive Graph Neural Network models that are able to leverage the graph together with node/edge features such as text embeddings, popularity, and contextual signals.

8 ACKNOWLEDGMENT

This work was partially supported by projects FAIR (PE0000013) and SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

REFERENCES

[1] Kumaripaba Athukorala, Antti Oulasvirta, Dorota Głowacka, Jilles Vreeken, and Giulio Jacucci. 2014. Narrow or broad? Estimating subjective specificity in exploratory search. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 819–828.

[2] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2005. Query recommendation using query logs in search engines. In *Current Trends in Database Technology-EDBT 2004 Workshops: EDBT 2004 Workshops PhD, DataX, PIM, P2P&DB, and ClustWeb, Heraklion, Crete, Greece, March 14-18, 2004. Revised Selected Papers 9*. Springer, 588–596.

[3] Ricardo Baeza-Yates and Alessandro Tiberi. 2007. Extracting semantic relations from query logs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 76–85.

[4] Ranieri Baraglia, Fidel Cacheda, Victor Carneiro, Diego Fernandez, Vreixo Formoso, Raffaele Perego, and Fabrizio Silvestri. 2009. Search shortcuts: a new approach to the recommendation of queries. In *Proceedings of the third ACM Conference on Recommender Systems*. 77–84.

[5] Doug Beeferman and Adam Berger. 2000. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 407–416.

[6] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. 2011. Query suggestions in the absence of query logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 795–804.

[7] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. 2009. Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data*. 56–63.

[8] Nick Craswell and Martin Szummer. 2007. Random walks on the click graph. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 239–246.

[9] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. 39–46.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Wai-Tat Fu, Thomas G Kannampallil, and Ruogu Kang. 2010. Facilitating exploratory search by model-based navigational cues. In *Proceedings of the 15th international conference on Intelligent user interfaces*. 199–208.

[12] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.

[13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[14] Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. 2003. Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the American Society for Information Science and Technology* 54, 7 (2003), 638–649.

[15] Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*. 271–279.

[16] Jyun-Yu Jiang and Wei Wang. 2018. RIN: Reformulation inference network for context-aware query suggestion. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 197–206.

[17] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*. 387–396.

[18] Ang Li, Jennifer Thom, Praveen Chandar, Christine Hosey, Brian St Thomas, and Jean Garcia-Gathright. 2019. Search mindsets: Understanding focused and non-focused information seeking in music search. In *The World Wide Web Conference*. 2971–2977.

[19] Xinyi Liu, Wanxian Guan, Lianyun Li, Hui Li, Chen Lin, Xubin Li, Si Chen, Jian Xu, Hongbo Deng, and Bo Zheng. 2022. Pretraining Representations of Multimodal Multi-query E-commerce Search. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3429–3437.

[20] Hehuan Ma, Yu Rong, and Junzhou Huang. 2022. Graph Neural Networks: Scalability. In *Graph Neural Networks: Foundations, Frontiers, and Applications*, Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao (Eds.). Springer Singapore, Singapore, 99–119.

[21] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[22] Christopher D Manning. 2008. *Introduction to information retrieval*. Syngress Publishing,.

[23] Gary Marchionini. 2006. Exploratory search: from finding to understanding. *Commun. ACM* 49, 4 (2006), 41–46.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013).

[25] Agnès Mustar, Sylvain Lamprier, and Benjamin Piwowarski. 2021. On the study of transformers for query suggestion. *ACM Transactions on Information Systems (TOIS)* 40, 1 (2021), 1–27.

[26] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *CoCo@ NIPs*.

[27] Emilie Palagi, Fabien Gandon, Alain Giboin, and Raphaël Troncy. 2017. A survey of definitions and models of exploratory search. In *Proceedings of the 2017 ACM workshop on exploratory search and interactive data analytics*. 3–8.

[28] Enrico Palumbo, Andrea Mezzalira, Cristina Sánchez-Marco, Alessandro Manzotti, and Daniele Amberti. 2020. Semantic Diversity for Natural Language Understanding Evaluation in Dialog Systems. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*. 44–49.

[29] Enrico Palumbo, Diego Monti, Giuseppe Rizzo, Raphaël Troncy, and Elena Baralis. 2020. entity2rec: Property-specific knowledge graph embeddings for item recommendation. *Expert Systems with Applications* 151 (2020), 113235.

[30] Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, Elena Baralis, Michele Osella, and Enrico Ferro. 2018. Knowledge graph embeddings with node2vec for item recommendation. In *European semantic web conference*. Springer, 117–120.

[31] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems*. 1–es.

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[33] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[35] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[36] Harald Steck. 2013. Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM conference on Recommender systems*. 213–220.

[37] Federico Tomasi, Rishabh Mehrotra, Aasish Pappu, Judith Bütepage, Brian Brost, Hugo Galvão, and Mounia Lalmas. 2020. Query Understanding for Surfacing Under-served Music Content. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2765–2772.

[38] Ryen W White and Resa A Roth. 2009. Exploratory search: Beyond the query-response paradigm. *Synthesis lectures on information concepts, retrieval, and services* 1, 1 (2009), 1–98.

[39] Bin Wu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Query suggestion with feedback memory network. In *Proceedings of the 2018 World Wide Web Conference*. 1563–1571.

[40] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.