Neural Instant Search for Music and Podcast

Helia Hashemi*
University of Massachusetts Amherst
hhashemi@cs.umass.edu

Aasish Pappu
Spotify
aasishp@spotify.com

Mi Tian
Spotify
mitian@spotify.com

Praveen Chandar
Spotify
praveenr@spotify.com

Mounia Lalmas
Spotify
mounia@acm.org

Ben Carterette
Spotify
benjaminc@spotify.com

ABSTRACT

Over recent years, podcasts have emerged as a novel medium for sharing and broadcasting information over the Internet. Audio streaming platforms originally designed for music content, such as Amazon Music, Pandora, and Spotify, have reported a rapid growth, with millions of users consuming podcasts every day. With podcasts emerging as a new medium for consuming information, the need to develop information access systems that enable efficient and effective discovery from a heterogeneous collection of music and podcasts is more important than ever. However, information access in such domains still remains understudied. In this work, we conduct a large-scale log analysis to study and compare podcast and music search behavior on Spotify, a major audio streaming platform. Our findings suggest that there exist fundamental differences in user behavior while searching for podcasts compared to music. Specifically, we identify the need to improve podcast search performance.

We propose a simple yet effective transformer-based neural instant search model that retrieves items from a heterogeneous collection of music and podcast content. Our model takes advantage of multi-task learning to optimize for a ranking objective in addition to a query intent type identification objective. Our experiments on large-scale search logs show that the proposed model significantly outperforms strong baselines for both podcast and music queries.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**; **Multimedia information systems**.

KEYWORDS

Music search; podcast search; instant search; neural information retrieval

ACM Reference Format:
Helia Hashemi, Aasish Pappu, Mi Tian, Praveen Chandar, Mounia Lalmas, and Ben Carterette. 2021. Neural Instant Search for Music and Podcast. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467188

*A part of this work was done while Helia Hashemi was interning at Spotify.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467188

1 INTRODUCTION

Music listening on streaming platforms is enjoying ever-growing popularity in the last decades, enabled by the abundance of music content in digital format and online streaming services such as Apple Music, Deezer, Pandora, and Spotify. Compared to digital music, podcasts are a relatively new media format with rapid growth in recent years. According to a recent survey,[1] 55% of people in the US have listened to a podcast in 2020, with a 7.8% and 22.2% relative growth from 2019 and 2010, respectively. 37% of people in the US have listened to a podcast in the last month, a growth of 15.6% and 208% from 2019 and 2010, respectively. Increasingly, more audio streaming services are now expanding their item catalogs to support both music and podcast. Such transition calls for developing search technologies that provide access to multiple item types through a single channel. This paper provides the first study on joint music and podcast search. We first answer the following research question:

How do users search and consume music and podcasts?

We address this research question by performing a large-scale log analysis on data collected from Spotify search logs. We focus on highlighting the differences between query characteristics, user effort, consumption patterns, and search goals for both music and podcast. We found that user effort is considerably higher in finding relevant podcasts in comparison to music searches. We further show that the search goals are significantly different in music compared to podcasts. For instance, users are more likely to organize songs returned by the search engine by adding them to a playlist or following the artists. In contrast, they are more interested in downloading the podcasts or sharing a podcast link, e.g., on social media or instant messaging applications.

These insights motivated us to focus our effort in improving information access systems for music and podcasts. We therefore address the following research question:

How can we effectively provide quick access to music and podcast using a unified retrieval model?

Recent work on music search suggested that users evaluate the search experience based on their ability to find the right content (success) and the amount of work needed (effort). User frustration arises when they cannot find what they are looking for or when the effort is too high [17]. This can be even more severe for podcasts compared to music, partly because the user needs to be more

[1] https://www.thepodcasthost.com/listening/podcast-industry-stats/

selective when making a decision due to the high time investment associated with podcast listening. This highlights the importance of information access that requires less efforts from users for both music and podcast search.

In this work, we focus on *instant search* as it aims to reduce user effort. It is a search paradigm that updates the search results pages (SERPs) for each keystroke event, instantly. It has been adopted by a variety of internet applications, such as social networks, e-commerce, and entertainment [39]. Due to the nature of instant search, we develop a character-level transformer-based attention model that re-ranks a set of candidate items (music and podcasts) in response to a given query prefix. In addition to matching the textual description of queries and items, we incorporate an item embedding component to capture item popularity.

Based on the different search behaviors observed for music and podcast intents in our log analysis, we propose a multi-task learning objective for training the model. The model not only learns to distinguish relevant from non-relevant items (i.e., ranking objective), but also learns to distinguish different query intent types (i.e., music vs. podcasts). We use the last interacted item within the search session as a *weak supervision* signal [8] for the query intent type. We show that this enables the model to retrieve from a heterogeneous collection of music and podcasts with higher precision. Some of the techniques used in this paper have existed and explored in other domains; however, in addition to the domain of the task, the effective combination of these techniques is unique.

To evaluate the model, we use a large-scale dataset containing real search queries sampled from Spotify search logs. Our experiments show that the proposed model, referred to as Neural Instant Search (NIS), significantly outperforms strong baselines, ranging from a simple yet effective heuristic model [1] to strong baselines such as Bidirectional Encoder Representations from Transformers (BERT) re-ranking models [33] and character-level neural models adopted from the query auto-completion literature [34]. We perform an extensive ablation study to investigate the impact of each novel component used in NIS on ranking effectiveness. We evaluate the models as it observes more keystrokes (i.e., higher query length), and show that short queries (e.g., less than four characters) largely benefit from the introduced item embedding layer since it can capture concepts such as item popularity. This is especially practical for instant search engines where large amount of prefix queries exist. On the other side, the proposed multi-task learning objective has a higher impact on retrieval effectiveness for longer queries. Therefore, these two aspects complement each other for an instant search in the domain of music and podcasts. We finally demonstrate that NIS improves ranking for both music queries and podcast queries.

2 RELATED WORK

We discuss prior research on topics relevant to our work and introduce background knowledge.

Query Auto-completion. Both *Query auto-completion* (QAC) and instant search [4] deal with query prefixes that evolve with each keystroke. However, the underlying goals behind them are different. Little is published on instant search, thus we mainly focus on QAC in this section, as character-level techniques used for QAC can be related to instant search models. The QAC task is predicting the complete version of users' query given its prefixes, while instant search aim to update the result list at each keystroke. QAC methods often relies on a prebuilt index (e.g. trie-based [18, 23, 26]) to generate a set of candidates for a given prefix. A ranker is then used to re-rank the set of candidate items and the user is presented with the top k items. As described in [4, 10], there are three main re-ranking approaches: (1) heuristic-based, (2) learning to rank, and (3) neural approaches.

One of the early works on QAC described in [2] uses the *Most Popular Completion* (MPC), recognized as a *heuristic based* approach. The method ranks candidates based on frequency counts estimated using a training dataset. An extension, first explored in [37] and further studied in [40], incorporated temporal information when estimating frequencies. Incorporating users' long-term search history[5], exploiting sequential interaction behavior across user sessions [28] and including skipping and other negative signals [43] were all shown to improve performance. We use the enhanced version of MPC model as a simple yet effective baseline in this work.

Various *learning-to-rank* frameworks based on Lambda-Mart combining user behavioral features extracted from search sessions have been used for QAC. These include developing features from user's short-term and long-term search historical logs that are then used to personalize query completions [36], exploiting user's current search session to develop reformulation related features Jiang et al. [22], and extracting features based on semantic relatedness between the prefix and the candidate [3]. These learning-to-rank approaches are based mostly on hand-crafted features, which are tedious to develop and time-consuming. In this work, we rely on an end-to-end neural model to achieve prefix representations. Finally, Li et al. [29] introduced high-resolution query logs, which recorded every keystroke along with the ranked list presented to the user. Similarly, we employ a high-resolution query log dataset (see Section 5.1) to evaluate our proposed approach.

Traditional QAC methods are less effective when a user enters a prefix that does not match any candidates in the index. To overcome this problem, deep generative models have been proposed to automatically generate candidate queries using n-grams and features from a query representation [30–32]. Park and Chiba [34] showed that generative models built using character-level long short-term memory (LSTM) cells exhibit performances comparable to state-of-the-art QAC for already seen queries. Similarly, the use of word2vec [35], recurrent neural networks [11, 20], and attention mechanism [7] have shown promising results. Mitra [31] also proposed the use of a convolutional latent semantic model (CLSM) to learn a distributed representation for prefixes and reformulations. The approach relies on a deep neural network based on CLSM to represent the prefixes and Lambda-Mart for re-ranking. In this work, we use character-level LSTMs and BERT [33] re-ranking algorithm, as two strong neural baselines.

Neural Ranking Models. These have been successful in a variety of retrieval tasks [13, 19, 33, 41, 42]. These models can be generally partitioned into early and late combination models [8]. In early combination models, query and document are combined in first

layers of network by using an interaction matrix [12] or by concatenation and learning self-attention weights between query and document tokens [33]. Late combination models learn query and document representations separately and compute their similarity at the final layers of the network. Similar to Nogueira and Cho [33], we use an early combination model by concatenating the query and item title and feeding it to a Transformer network [38].

Despite of the dominance of neural models in ad-hoc retrieval and web search tasks, their ability in improving search quality for instant search tasks is less studied. Many of the works discussed earlier take word embeddings as their inputs, leading to ineffective performance in instant search tasks where documents (items) should be ranked in response to incomplete queries (query prefixes), which may not be even in the vocabulary. This motivates us to focus on character-level representation in our model. Unlike the neural ranking models designed for ad-hoc retrieval, we also learn a representation for each item in the collection to model item popularity. We also employ multi-task learning to deal with different item types in the collection.

Aggregated Search. While traditional search engines return a ranked list of search results dedicated to certain media types, *aggregated search* aims to assemble information relevant to queries from heterogeneous information sources, also called *verticals* (e.g. images, videos, blogs, etc.), into one result interface to support diverse search needs [25, 27, 44]. An aggregated search system commonly consists of two components: selecting verticals relevant to the queries and incorporating the verticals into a unified presentation.

Our work adds to the body of work related to aggregated search, but with three main differences. First, our work focuses on instant search and the intersection of aggregated search and instant search is relatively unstudied. Also, due to the nature of incomplete prefix queries in instant search, vertical selection would be challenging. Second, the recent progress on neural ranking models has highlighted the importance of end-to-end training for such models [13] and it is difficult or even impractical to optimize the whole aggregated search pipeline in an end-to-end manner. As an alternative, we aim to address the ambiguity of incomplete prefix queries by jointly optimizing the query intent prediction and instant search.

Finally, aggregated search methods often combines multiple rank lists returned by different verticals (e.g., web page ranking and image search). Since podcast queries and click signals are underrepresented in our data, our initial attempts show that training a separate model for podcast retrieval does not beat the baselines included in the paper. This is why we aim for training a single model that can compute the relevance score for different item types using multi-task learning. This enables the model to transfer and generalize knowledge from a relatively larger training data for music queries to podcast search.

3 LOG ANALYSIS: PODCAST VS. MUSIC SEARCH

To demonstrate the need of our proposed approach, we perform an analysis of query logs from Spotify. Similar to the prior works presented in Section 2, our instant search system used a trie-based index to generate candidates and a LambdaMart-based re-ranker with features extracted from user history and query-item pairs.

Table 1: Search goals and behaviors in podcast vs. music.

Search Goal	Behavior	Rel. diff to music
Listen	Stream	+3.13%
Organize	Add to collection	+29.57%
	Add to playlist	-59.12%
	Follow artist	-92.37%
	Follow playlist	-39.88%
	Download	+593.02%
Share	Share link	+44.44%

Specifically, we logged the search result pages (SERPs) returned for each keystroke along with users' interactions (such as clicks or taps) on the results. The query prefixes were grouped into a session using a timeout threshold. Unlike traditional QAC logs, we do not terminate a session when a user clicks on a result item, since our interface allows users to navigate back to the SERP and continue interacting with it after browsing the result item. As an alternative, sessions are terminated when the edit distance ratio between two consecutive prefixes exceeds a threshold or when user abandons the session (eg. timeout).

It is worth pointing out that the instant search system in our setup is required to provide a personalized ranked list of candidate items, both podcasts and music. For a given prefix the ranked list is updated for each keystroke entry. Most importantly, as the user is typing the query, the system is unaware of their intent, i.e., whether they are looking for podcasts or music-related content.

The data that we use is introduced in section 5.1. We label the user intent type based on the type of the users' last interacted URI in the session. Based on this assumption the user intent type is either podcast or music.[2]

Interpreting query logs for domain-specific search applications can be complex due to uncertainty around user goals and expectations. In this work, we follow the framework proposed by Hosey et al. [17] and Chandar et al. [6] and describe user interaction behavior with *consumption* and *effort* metrics. Consumption metrics such as *clicks/stream-rate*, *number of follows*, *downloads*, or *shares* help us understand how users interact with the SERPs, when they are looking for podcast vs. music. Metrics such as *number of deletions*, *keystrokes*, *clicked rank positions*, etc. quantify the user effort needed to find their desired music or podcast. Consumption and effort metrics provide insights into the performance of an effective system when dealing with heterogeneous candidates.

After the user session is complete, we attribute the query prefixes to a consumption metric. Since we rely on consumption metrics to determine podcast vs. music intent for given prefix in our logs, we discard all sessions with no interactions. While this may result in an overestimation of the absolute metrics, we focus on the relative comparisons between podcast and music-related queries for the scope of this paper.

Table 2: Podcast search efforts compared to music.

Effort type	Rel. diff to music
Avg. deletions (# characters)	+53.25%
Avg. query length (# words)	+0.38%
Avg. query length (# characters)	+12.63%

3.1 Analysis of Consumption Patterns

Table 1 shows the relative difference in consumption metrics for podcasts and music queries. Interestingly, download and share metrics are considerably higher for podcasts search compared to music. This behavior suggests that unlike music content, users are likely curating podcasts for future listening. This behavior is somewhat expected given that users perceive podcasts as a high investment activity. The lower "add to playlist" and "follow playlist" metrics, on the other hand, may indicate a lesser need for repeated listening. Further, higher share rates suggest that podcast listening is likely more social compared to music, at least when users are actively seeking podcasts in search.

3.2 Analysis of User Effort

Table 2 reports the difference in user effort between queries with music and podcast consumption metrics. Queries with podcast consumption metrics are 12.63% longer in characters compared to queries with music consumption metrics. The average deletions for the former is 53.9% higher than the latter. We also observe more pagination events and "see more results" clicks for podcast queries, which suggests a higher search effort.

This suggest that the search engine used in this analysis, which is based on learning to rank model, needs more effort to find podcasts than music. Next, we investigate the potential reasons for podcast content to be harder to search for compared to music.

3.3 Summary

We showed the distinctions between user consumption and effort metrics between podcast and music queries. While some of the differences in consumption can be explained by the differences in user expectations associated with different content types, there still exists a huge disparity in performance in terms of effort metrics. This motivates us to jointly optimize the classification of queries between music and podcasts, as well as rank the items that best match the prefix query. While our analysis is conducted using data from a single platform, this disparity could be observed in other search platforms in which content heterogeneity exists.

4 NEURAL INSTANT SEARCH

Our goal is to develop an instant search model for re-ranking items (i.e., music and podcasts) in response to a given query. The major characteristic that distinguishes instant search from most retrieval tasks is the nature of queries. In instant search the model refreshes the result list for each keystroke. To deal with such interactive queries, we design our model based on character-level embedding. The other main characteristic of our task is the heterogeneous nature of items. There are two different types of items in the collection and our log analysis in Section 3 shows that users interact with music and podcast results differently. To address this issue, we propose a multi-task learning objective for training Neural Instant Search (NIS) models. The NIS optimization minimizes a ranking loss in addition to a query intent type identification loss.

In the rest of this section, we introduce the problem formulation and describe the network architecture and optimization in NIS.

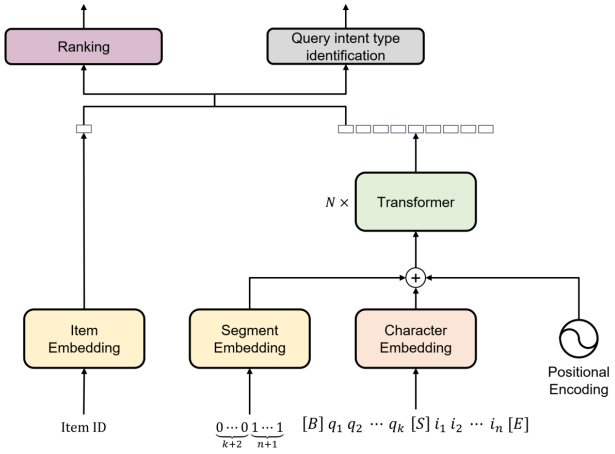

Figure 1: Neural Instant Search Architecture. The model uses multi-task learning to simultaneously optimize two losses, one for item ranking and another for query intent type identification. The input representation is based on character-level embedding suitable for instant search problems.

4.1 Problem Formulation

Let $D = \{(q_1, I_1, T_1, R_1), (q_2, I_2, T_2, R_2), \cdots, (q_N, I_N, T_N, R_N)\}$ be the training set for N queries, in which q_i denotes the query text for the i^{th} query and $I_i = \{I_{i1}, I_{i2}, \cdots, I_{in}\}$ denotes the set of n candidates items for the query q_i. We define T_i such that it represents the last interacted item type for the query q_i, a proxy for query intent type. In this study, item types can be either music or podcast. The j^{th} element in R_i also denotes the relevance label for the item I_{ij} to the query q_i. In our study, we use click data as implicit relevance label, therefore, we assume that the relevance signals are binary.

To this end, we formulate our task as training a model M using the training data D in order to score a given query and item pair in the test set.

4.2 The NIS Architecture

As outlined above, there are two properties that distinguish our problem from traditional retrieval tasks. First, we deal with the instant search problem in which the result list needs to be updated at each keystroke. Second, our collection consists of different item types (i.e., music and podcasts), which results in a heterogeneous

[2]We are aware that this way of labeling leads to binary intent types and does not support mixed intents. However, given the application of music and podcast, and also based on our observation from the data, the binary assumption of intent types (music vs. podcast) is not far from reality.

collection. In our network architecture, we address the first property by introducing character-level embedding vectors to the model. The goal is to enable the model to match the input query prefix to the item description. We approach the second property by introducing a query intent type identification component to the network. In fact, this component allows us to take advantage of a multi-task learning optimization, which is further explained in Section 4.3.

Given an input query text q_i and item title I_{ij}, we first perform character-level tokenization on both the query and the item. Inspired by [9], we concatenate the obtained characters using a beginning, a separation and an end token, and pad the representation to the maximum sequence length if needed. As the first step, we take the embedding vectors for each character in the input sequence. Since our model is based on the Transformer architecture, we also compute a positional encoding for each input character, as described in [38]. A segment embedding vector is also used to enable the model distinguish query characters from item characters. Similar to the BERT model [9], segment embedding is implemented as two embedding vectors, one for the query and another for the item. Adding character, positional, and segment embeddings shapes the input representation for our Transformer layers [38].

We use Transformer layers in our network that use a self-attention mechanism by introducing three weight tensors: the query tensor Q,[3] the key tensor K, and the value tensor V. Each self-attention function is computed as follows:

$$Z = \text{softmax}(\frac{Q \times K^T}{\sqrt{d}})V \tag{1}$$

where d is the embedding dimensionality. Equation 1 thereby represents an attention probability distribution over the sequence tokens using the $softmax$ operator applied to the query-key similarities. Then the representation of each token is computed based on the linear interpolation of values. We use Transformer layers in our model due to the following reasons: (a) the self-attention mechanism in Transformer leads to contextual representation of the input sequence using the linear interpolation of all value vectors in the input, and contextual representations have been found effective in a number of IR and NLP tasks [9, 14, 15, 33]; and (b) learning attention weights from every character in the query to every character in the item title by the $Q \times K^T$ component in Equation (1) empowers the model to match query prefixes to any part of the item title (e.g., in the beginning or in the middle of the title). Multiple levels of abstraction for the input sequence have been computed by employing multiple layers of Transformer (the number of layers was set to four in most of the experiments).

As suggested by previous work [9, 15], the representation learned for the beginning token, called *the query-item matching representation*, is used to represent the whole input sequence. This vector represents how the query text matches the item title. However, it cannot represents some notions, such as item popularity, which are important for retrieval tasks. Therefore, inspired by the neural collaborative filtering approaches [16], we introduce an item embedding vector that is independent of its title and is modeled as a look-up table for all items in the collection. We concatenate the query-item matching representation with the item embedding vector. Finally, as shown in Figure 1, we feed the obtained representation to two independent fully-connected layers, one responsible for producing the relevance score $\hat{R}_{ij}$, and another for predicting the query intent type $\hat{T}_i$.

[3]Note that the query tensor is not related to the search query and they just share the same terminology.

4.3 The NIS Multi-Task Training

Our analysis in Section 3 suggests that music and podcast search, despite having much in common, are substantially different, and thus should be treated differently. As introduced earlier, we propose a multi-task learning objective to provide signals to the model by considering their unique characteristics in ranking. We use a pointwise ranking objective, which is the main task of the model.

We implement the loss function using the cross entropy function, which is equivalent to maximum likelihood optimization. The loss function for the i^{th} query q_i and the j^{th} candidate item I_{ij} is defined as:

$$L_{\text{ranking}} = -R_{ij} \log \hat{R}_{ij} - (1 - R_{ij}) \log (1 - \hat{R}_{ij}) \tag{2}$$

where R_{ij} and $\hat{R}_{ij}$ denote the binary relevance label from the training set D and the model's relevance score, respectively.

The second objective in our model is to predict the query intent type (i.e., whether user is looking for music or podcast content). The label for this task is obtained based on the last item type that the user interacted with in the search session logs (see Section 4.1 for more information). Since in our experiments we only have two item types (i.e., music and podcasts), we again use binary cross entropy for this auxiliary loss function:

$$L_{\text{intent type}} = -T_i \log \hat{T}_i - (1 - T_i) \log (1 - \hat{T}_i) \tag{3}$$

where T_i and $\hat{T}_i$ denote the query intent type (0 for music and 1 for podcast) and the model's predicted intent type for the given query.

Our final loss function is obtained by a linear interpolation of the two described losses. Therefore, we use a gradient-based optimization approach to minimize the following loss function:

$$\mathcal{L} = L_{\text{ranking}} + \alpha L_{\text{intent type}} \tag{4}$$

where α is a hyper-parameter controlling the impact of the auxiliary loss function (i.e., the query intent type identification loss).

5 EXPERIMENTS

We empirically address the following four research questions:

- **RQ1** How does NIS perform in comparison with state-of-the-art baselines?
- **RQ2** What is the impact of major NIS components (contributions) on the retrieval performance? (ablation study)
- **RQ3** How does NIS perform for different query intents (music vs. podcast)?
- **RQ4** How robust is the NIS to varying query lengths?

We collected data from real user interactions with the Spotify's instant search engine (Section 5.1). We describe our experimental setup and evaluation metrics in Sections 5.2 and 5.3, respectively, and answer our four research questions in Section 5.4.

5.1 Data

Traditionally, research on both query auto-completion and instant search has relied on web search logs where user interactions are

Table 3: An example of the log data used to train and evaluate the instant search models.

session id	query prefix	rank	candidate	click	query intent type
1	d	1	**D**rake	0	music
1	dr	2	**Dr**ake	1	music
2	b	1	**B**eyonce	0	music
3	d	1	**D**rake	0	music
3	da	2	**Da**vid Bowie	0	music
3	dai	3	**Dai**sies	0	music
3	dail	4	**Dail**y show	1	podcast

simulated from the submitted queries. In contrast, in this work, we rely on realistic search logs generated by users interacting with a real-world instant search system within the Spotify platform. In our setup, users interact with an interactive search system that returns personalized search results, consisting of music and podcast items displayed to the user for a given query prefix. The result page is updated in response to each keystroke.

This setup provides a more realistic scenario for evaluation. For a given prefix, each log line consists of a ranked list of results displayed to the user along with interactions such as click, streams, etc. Additionally, the query prefixes are grouped into individual sessions based on user interactions. A search session is terminated when a user becomes inactive, or abandons the query (e.g. navigates away or timeout), or when the edit distance ratio exceeds certain threshold. A sample of the logs used in this work is show in Table 3.

To evaluate the proposed approach, we compiled a dataset by sampling logs from the instant search sessions over the period of one week in June 2020. We do not exclude any language or symbol that exists in the input of the system in this period. The dataset consists of about 100 million sessions. We restrict the sample to sessions that resulted in at least one click on music or podcast result and subsample from that set. We split the data by user IDs and dedicated the data for 80% of random users to training, 10% to validation, and 10% to test sets. This resulted in 6M query prefixes in the training set, 750K in the validation set, and 750K in the test set. It should be noted that in instant search, unlike many other retrieval tasks, splitting the data based on queries (prefixes) would be problematic as two prefixes for the same query may end up in the training and test sets. On the other hand, splitting based on user IDs evaluates the ability of the models on generalizing from the data observed from a group of users to another (e.g. new users).

5.2 Experimental Setup

In our experiments, we re-rank the items returned by a strong candidate generation models that use a trie based index and re-rank them using LambdaMART with a number of hand-crafted features as described in Section 3. The auxiliary task introduced in Section 4.2, query intent type identification, is solely used in the training phase. Relevance labels for the re-ranking task are acquired based on users click logs. The judgments for podcast query prediction task comes from the user's last interacted item within a session. The same prefixes in the test set are aggregated and then the results are reported. The hyper-parameters were selected based on the performance (in terms of NDCG@10) on the validation set. We then used the selected hyper-parameters to evaluate the model on the test set. The same procedure was used for the proposed model and all the baselines.

We implemented our model using TensorFlow.[4] We optimize the network parameters using the Adam optimizer [24] with the initial learning rate of 3×10^6, $\beta_1 = 0.9$, $\beta_2 = 0.999$, L_2 weight decay of 0.01, learning rate warm-up over the first 5000 steps, and linear decay of the learning rate. The dropout probability 0.1 is used in all hidden layers. The number of attention heads in multi-headed attentions is set to 12. The maximum sequence length for the queries and the items are set to 25 and 100, respectively. We use 4 layers of Transformer in our model. The batch size is set to 128. The other hyper-parameters, including the parameter α (Equation 4) were selected based on the performance on the validation set.

5.3 Evaluation Metrics

Due to the nature of our task, i.e., re-ranking based on click logs, we focus more on precision-oriented metrics such as mean reciprocal rank (MRR) and normalized discounted cumulative gain (NDCG) [21] with ranking cut-off of 10, and R-Precision (RPrec). For the sake of generality, we also include mean average precision (MAP) as a recall-oriented metric in our experiments. We report the average performance across different prefixes in the test set. We identify statistical significant improvements using the paired t-test with Bonferroni correction at 95% and 99% confidence intervals (i.e., p-value less than 0.05 and 0.01, respectively).

5.4 Results and Discussion

In this section, we empirically study the research questions introduced earlier, one by one.

5.4.1 ***RQ1: Comparison Against Baselines.*** To address our first research question, we compare the proposed model against the following strong baselines for instant search.

Prefix Match and Item Popularity (PMIP). This is a simple yet effective baseline that first ranks items based on prefix matching. The longer the matched prefix between query and item, the higher the retrieval score. Since we re-rank the candidate items retrieved by the production search engine, many of them already have high prefix matching scores (even as high as the query length). Therefore, in many cases, multiple items achieve the same prefix matching score. To break the ties, we use item popularity. This means that more popular items are ranked higher, with the assumption that they are more likely to be clicked. Item popularity has been computed based on the number of times the item has been clicked in the training set. Therefore, this model is an enriched version of the popular heuristic MPC model [2, 37].

BERT Re-Ranking Model (BERT). Large-scale pre-trained language models, such as BERT [9], have been successfully adopted by the information retrieval community for various retrieval tasks [15, 33]. These models use subword embedding, as opposed to word embedding, which makes them a more appropriate baseline

[4] https://www.tensorflow.org/

Table 4: The performance of NIS compared to strong baselines for instant search on a collection of music and podcast. The superscript $*$ denotes significant improvements compared to all baselines using two-tail paired t-test with Bonferroni correction with 99% confidence ($p\_value < 0.01$).

Model	NDCG@10	RPrec	MRR	MAP
PMIP	0.6264	0.3216	0.5573	0.5353
BERT	0.6468	0.3728	0.5852	0.5711
LSTM-Char	0.6522	0.3662	0.5886	0.5673
NIS	**0.6740***	**0.4056***	**0.6245***	**0.5968***

for instant search models, since most queries in instant search are not in the vocabulary but rather can be represented as multiple subwords.[5] Following Nogueira and Cho [33], we concatenate the query and item description with a separation token and fine-tune the BERT model for ranking. In our experiments, we used the pre-trained BERT-base model. The cross-entropy loss function is used for optimization and the hyper-parameters were selected based on the result on the validation set.

Character-based LSTM (LSTM-Char). Park and Chiba [34] proposed a neural language model based on LSTM networks for query auto-completion. We adapt their model to the instant search task by modifying the final layers to produce a single relevance score for query and item pairs, and change the learning objective to ranking (i.e., cross entropy). This model, similar to ours, uses character-level tokenization.

The results for the proposed model and the baselines are reported in Table 4. Among the baselines, BERT shows the strongest performance in terms of RPrec and MAP, while LSTM-Char performs better in terms of NDCG@10 and MRR. This might be surprising at first glance, as BERT uses large-scale pre-training and Transformer networks. However, while BERT uses subword tokenization (or WordPiece) that breaks a word into several subwords, the result lists are updated for every new keystroke in the context of instant search. Therefore, a simpler model based on LSTM networks (i.e., LSTM-Char) that is able to represent each character (as opposed to subwords) yields comparable results. In addition, the PMIP baseline, despite of its simplicity, also shows a strong performance. Our proposed model outperforms all the baselines in terms of all metrics. According to two-tailed paired t-test with Bonferroni correction, the improvements are statistically significant with $p\_value < 0.01$. These results answer the first research question (RQ1).

5.4.2 ***RQ2: Ablation Study.*** To address the second research question, we perform an ablation study by omitting some of the major components of the model introduced in the paper. We compare NIS with the following variants of model: (1) without multi-task learning (i.e., NIS- query intent type identification (STL)), (2) without introducing the item embedding in the last layer of network (i.e., NIS- item embedding), and (3) without both multi-task learning and item embedding. The results for this ablation study is presented in Table 5. Comparing the results obtained by NIS and NIS- query intent type identification (STL), the proposed multi-task learning objective leads to significant improvements. This demonstrates the effectiveness of using an auxiliary task for query intent type identification on ranking.

[5]Note that there is no public pre-trained BERT model with character-level tokenization.

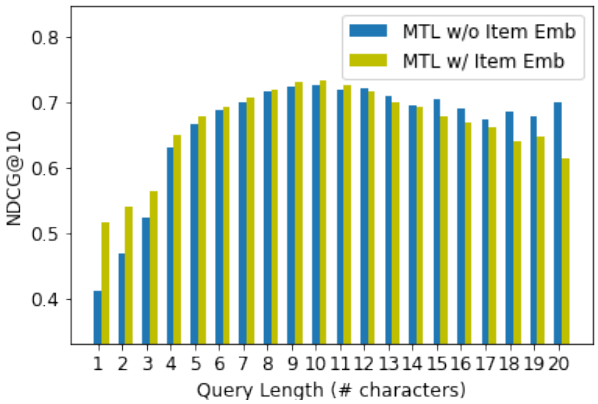

Figure 2: Results of multi-task model with and without the item embedding module for different query lengths.

Comparing NIS and NIS-item embedding suggests that introducing an item embedding component to capture features, such as item popularity, leads to significant improvements in terms of all metrics. NIS-query intent type identification (STL) and NIS-item embedding perform on par. They both perform significantly better than the model that does include neither the multi-task learning objective nor the item embedding component.

This ablation study answers the second research questions (RQ2) by showing that both multi-task learning setting introduced in the paper and the item embedding component are crucial parts of the model to achieve a strong performance.

5.4.3 ***RQ3: Performance by Query Intent Types.*** To study the third research question, we first identify the query intent types for each test query by looking at the type of item (music or podcast) that the user click on in the search result list. If the user clicks on multiple items and if more than 70% of the clicked items are from the same item type, we assign it to the query. Otherwise, we omit the query in the analysis performed in this set of experiments.

In this set of experiments, we consider LSTM-Char as it achieves the highest performance among all the baselines in terms of MRR and NDCG@10 (see Table 4). We also include different variations of the proposed model as discussed in our ablation study experiment. The results are reported in Table 6. We observe consistent improvements by NIS for both music and podcasts queries. The improvements on podcast queries are more significant. For instance, comparing the MRR results obtained by NIS and LSTM-Char shows 7.5% relative improvements on podcast queries and 3.9% relative improvements on music queries. The results also suggest that all models perform better for music queries compared to podcast queries. This is most likely to be due to the highly unbalanced podcast and music queries in our data.

Table 5: Ablation study results. The superscripts 1/2/3 denote significant improvements compared to the methods with ID 1/2/3, using the two-tail paired t-test with Bonferroni correction with 99% confidence ($p\_value < 0.01$).

ID	Model	NDCG@10	RPrec	MRR	MAP
–	NIS	0.6740^{123}	0.4056^{123}	0.6245^{123}	0.5968^{123}
1	NIS-query intent type identification (STL)	0.6630^{3}	0.3902^{3}	0.6013^{3}	0.5852^{3}
2	NIS-item embedding	0.6618^{3}	0.3910^{3}	0.5994^{3}	0.5849^{3}
3	NIS- query intent type identification - item embedding	0.6581	0.3823	0.5954	0.5791

Table 6: Performance of the models for different query intent types (music and podcast). The superscripts $*/\dagger$ denote significant improvements compared to LSTM-Char/NIS variations (other than NIS itself) using two-tail paired t-test with Bonferroni correction with 99% confidence ($p\_value < 0.01$).

Model	Music Queries				Podcast Queries			
	NDCG@10	RPrec	MRR	MAP	NDCG@10	RPrec	MRR	MAP
LSTM-Char	0.6586	0.3684	0.6120	0.5736	0.6325	0.3633	0.5169	0.5425
NIS	$0.6801^{*\dagger}$	$0.4087^{*\dagger}$	$0.6360^{*\dagger}$	$0.6034^{*\dagger}$	$0.6618^{*\dagger}$	0.4098^{*}	0.5559^{*}	$0.5787^{*\dagger}$
NIS- item embedding	0.6675	0.3926	0.6087	0.5907	0.6557	0.4082	0.5527	0.5737
NIS- query intent type identification - item embedding	0.6533	0.3589	0.6021	0.5657	0.6323	0.3504	0.5695	0.5157

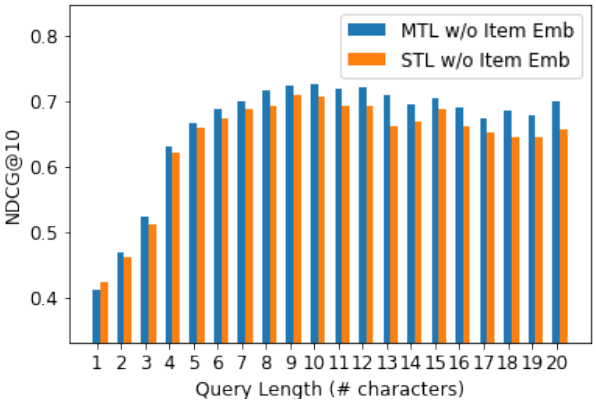

Figure 3: Results of the multi-task model compared to the single-task model for different query lengths.

5.4.4 ***RQ4: Performance by Query Length.*** To address the last research question, we extend our analysis to study the effect of query length on model's performance. As shown in Figure 3, NIS with a multi-task objective consistently outperforms NIS-query intent type identification (STL). The only exception here is the single character queries. The reason is that single character queries are too short, and it is difficult to determine whether the query intents are music or podcast. Therefore, rather than benefiting from the auxiliary task, the model gets misled in case of single character queries. We observe from Figure3 that as the query length increases, the query intent identification task becomes easier for the model, and as a result the performance margin between single-task model and multi-task model becomes larger.

We also study the performance of NIS with and without the item embedding for different query lengths in Figure 2. Interestingly, shorter queries benefit more from the item embedding component. The reason is that matching query text with item title for queries with few characters (e.g., less than or equal to three characters) is not sufficient for the effective retrieval of items. Additionally, features such as item popularity can play a key role in retrieval for such short queries. Item embedding can capture such features. Although adding the item embedding components on average improves the performance (see Table 5), Figure 2 shows that long queries (more than 11 characters) do not benefit from this component. Improving this component for all query lengths remains as future work.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we studied instant search in the context of music and podcast search. We collected data from Spotify as a large streaming platform that includes both music and podcasts, and studied how users search for music and podcasts using a large-scale query log analysis. We quantified search effort metrics and showed that search effort for podcast retrieval is relatively higher than music. We also observed difference characteristics in user consumption behaviors for music and podcast in the context of streaming app.

Inspired by unique characteristics of search in a heterogeneous collection like music and podcast, we proposed a novel instant search model that matches query prefixes to any part of the item title using a self-attention mechanism, which is trained to distinguish different query intent types in addition to ranking items using multi-task learning. Our model outperforms strong baselines for both music and podcasts queries. Specifically, we gained larger improvements for podcast queries that are underrepresented in the data. Last but not least, our ablation study showed that the multi-task learning and the item embedding component introduced in our model not only contribute to ranking metrics individually, but also complement each other for improved model performance.

In the future, we intend to enrich the model by augmenting the meta data such as user history, artist, podcasts' host, etc. We also plan to extend the model to consume the full item descriptions, such as podcast transcripts and music lyrics. In addition, we are interested in studying voice queries for music and podcast search. They are mostly in a complete form, as opposed to the ones typed by users, which are mostly incomplete query prefixes due to the use of

instant search. Bridging the gap between the observed incomplete instant search query prefixes and the complete spoken queries (e.g., voice search) to train a unified model is an important and challenging problem.

REFERENCES

[1] Ziv Bar-Yossef and Naama Kraus. [n.d.]. Context-Sensitive Query Auto-Completion. In *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*. 107–116.

[2] Ziv Bar-Yossef and Naama Kraus. 2011. Context-Sensitive Query Auto-Completion. In *WWW '11* (Hyderabad, India). Association for Computing Machinery, New York, NY, USA, 107–116. https://doi.org/10.1145/1963405.1963424

[3] Fei Cai and Maarten de Rijke. 2016. Learning from Homologous Queries and Semantically Related Terms for Query Auto Completion. *Inf. Process. Manage.* 52, 4 (July 2016), 628–643. https://doi.org/10.1016/j.ipm.2015.12.008

[4] Fei Cai and Maarten de Rijke. 2016. A Survey of Query Auto Completion in Information Retrieval. *Foundations and Trends® in Information Retrieval* 10, 4 (2016), 273–363. https://doi.org/10.1561/1500000055

[5] Fei Cai, Shangsong Liang, and Maarten de Rijke. 2014. Personalized Document Re-Ranking Based on Bayesian Probabilistic Matrix Factorization. In *SIGIR '14* (Gold Coast, Queensland, Australia). Association for Computing Machinery, New York, NY, USA, 835–838. https://doi.org/10.1145/2600428.2609453

[6] Praveen Chandar, Jean Garcia-Gathright, Christine Hosey, Brian St. Thomas, and Jennifer Thom. 2019. Developing Evaluation Metrics for Instant Search Using Mixed Methods Methods. In *SIGIR '19* (Paris, France). ACM, New York, NY, USA, 925–928. https://doi.org/10.1145/3331184.3331293

[7] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. 2018. Attention-Based Hierarchical Neural Query Suggestion. In *SIGIR '18* (Ann Arbor, MI, USA). ACM, New York, NY, USA, 1093–1096. https://doi.org/10.1145/3209978.3210079

[8] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *SIGIR '17*. 65–74.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL '19*.

[10] Giovanni Di Santo, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2015. Comparing Approaches for Query Autocompletion. In *SIGIR '15* (Santiago, Chile). Association for Computing Machinery, New York, NY, USA, 775–778. https://doi.org/10.1145/2766462.2767829

[11] Nicolas Fiorini and Zhiyong Lu. 2018. Personalized neural language models for real-world query auto completion. In *NAACL '18*. Association for Computational Linguistics, New Orleans - Louisiana, 208–215. https://doi.org/10.18653/v1/N18-3026

[12] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-Hoc Retrieval. In *CIKM '16* (Indianapolis, Indiana, USA). 55–64.

[13] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. 2019. A Deep Look into neural ranking models for information retrieval. *Information Processing & Management* (2019).

[14] Helia Hashemi, Hamed Zamani, and W. Bruce Croft. 2019. Performance Prediction for Non-Factoid Question Answering. In *ICTIR '19*. 55–58.

[15] Helia Hashemi, Hamed Zamani, and W. Bruce Croft. 2020. Guided Transformer: Leveraging Multiple External Sources for Representation Learning in Conversational Search. In *SIGIR '20* (Virtual Event, China). Association for Computing Machinery, New York, NY, USA, 1131–1140.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW '17* (Perth, Australia). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[17] Christine Hosey, Lara Vujović, Brian St. Thomas, Jean Garcia-Gathright, and Jennifer Thom. 2019. Just give me what I want: How people use and evaluate music search. In *CHI '19* (Glasgow, UK). Association for Computing Machinery, 1–12.

[18] Bo-June (Paul) Hsu and Giuseppe Ottaviano. 2013. Space-Efficient Data Structures for Top-k Completion. In *WWW '13* (Rio de Janeiro, Brazil) *(WWW '13)*. Association for Computing Machinery, New York, NY, USA, 583–594. https://doi.org/10.1145/2488388.2488440

[19] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *CIKM '13* (San Francisco, California, USA). 2333–2338.

[20] Aaron Jaech and Mari Ostendorf. 2018. Personalized Language Model for Query Auto-Completion. In *ACL '18*. Association for Computational Linguistics, Melbourne, Australia, 700–705. https://doi.org/10.18653/v1/P18-2111

[21] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[22] Jyun-Yu Jiang, Yen-Yu Ke, Pao-Yu Chien, and Pu-Jen Cheng. 2014. Learning User Reformulation Behavior for Query Auto-Completion. In *SIGIR '14* (Gold Coast, Queensland, Australia). Association for Computing Machinery, New York, NY, USA, 445–454. https://doi.org/10.1145/2600428.2609614

[23] Dimitrios Kastrinakis and Yannis Tzitzikas. 2010. Advancing search query autocompletion services with more and better suggestions. In *ICWE '10*. Springer, 35–49.

[24] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. http://arxiv.org/abs/1412.6980

[25] Arlind Kopliku, Karen Pinel-Sauvagnat, and Mohand Boughanem. 2014. Aggregated search: A new information retrieval paradigm. *ACM Computing Surveys (CSUR)* 46, 3 (2014), 1–31.

[26] Unni Krishnan, Alistair Moffat, and Justin Zobel. 2017. A Taxonomy of Query Auto Completion Modes. In *ADCS '17* (Brisbane, QLD, Australia). ACM, New York, NY, USA, Article 6, 8 pages. https://doi.org/10.1145/3166072.3166081

[27] Mounia Lalmas. 2011. Aggregated Search. In *Advanced Topics in Information Retrieval*, Massimo Melucci and Ricardo Baeza-Yates (Eds.). The Information Retrieval Series, Vol. 33. Springer, 109–123. https://doi.org/10.1007/978-3-642-20946-8_5

[28] Liangda Li, Hongbo Deng, Anlei Dong, Yi Chang, Hongyuan Zha, and Ricardo Baeza-Yates. 2015. Analyzing User's Sequential Behavior in Query Auto-Completion via Markov Processes. In *SIGIR '15* (Santiago, Chile). Association for Computing Machinery, New York, NY, USA, 123–132. https://doi.org/10.1145/2766462.2767723

[29] Yanen Li, Anlei Dong, Hongning Wang, Hongbo Deng, Yi Chang, and ChengXiang Zhai. 2014. A Two-Dimensional Click Model for Query Auto-Completion. In *SIGIR '14* (Gold Coast, Queensland, Australia). Association for Computing Machinery, New York, NY, USA, 455–464. https://doi.org/10.1145/2600428.2609571

[30] David Maxwell, Peter Bailey, and David Hawking. 2017. Large-Scale Generative Query Autocompletion *(ADCS 2017)*. Association for Computing Machinery, New York, NY, USA, Article 9, 8 pages.

[31] Bhaskar Mitra. 2015. Exploring Session Context Using Distributed Representations of Queries and Reformulations. In *SIGIR '15* (Santiago, Chile). Association for Computing Machinery, New York, NY, USA, 3–12. https://doi.org/10.1145/2766462.2767702

[32] Bhaskar Mitra and Nick Craswell. 2015. Query Auto-Completion for Rare Prefixes. In *CIKM '15* (Melbourne, Australia). Association for Computing Machinery, New York, NY, USA, 1755–1758. https://doi.org/10.1145/2806416.2806599

[33] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *CoRR* (2019).

[34] Dae Hoon Park and Rikio Chiba. 2017. A Neural Language Model for Query Auto-Completion. In *SIGIR '17* (Shinjuku, Tokyo, Japan). Association for Computing Machinery, New York, NY, USA, 1189–1192. https://doi.org/10.1145/3077136.3080758

[35] Taihua Shao, Honghui Chen, and Wanyu Chen. 2018. Query Auto-Completion Based on Word2vec Semantic Similarity. *Journal of Physics: Conference Series* 1004 (apr 2018), 012018. https://doi.org/10.1088/1742-6596/1004/1/012018

[36] Milad Shokouhi. 2013. Learning to Personalize Query Auto-Completion. In *SIGIR '13* (Dublin, Ireland). ACM, New York, NY, USA, 103–112. https://doi.org/10.1145/2484028.2484076

[37] Milad Shokouhi and Kira Radinsky. 2012. Time-Sensitive Query Auto-Completion. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Portland, Oregon, USA) *(SIGIR '12)*. Association for Computing Machinery, New York, NY, USA, 601–610. https://doi.org/10.1145/2348283.2348364

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS '17*. 5998–6008.

[39] Ganesh Venkataraman, Abhimanyu Lad, Viet Ha-Thuc, and Dhruv Arya. 2016. Instant Search: A Hands-on Tutorial. In *SIGIR '16* (Pisa, Italy). ACM, 1211–1214.

[40] Stewart Whiting and Joemon M. Jose. 2014. Recent and Robust Query Auto-Completion. In *WWW '14* (Seoul, Korea). Association for Computing Machinery, New York, NY, USA, 971–982. https://doi.org/10.1145/2566486.2568009

[41] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. 2017. Situational Context for Ranking in Personal Search. In *WWW '17* (Perth, Australia). 1531–1540.

[42] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *CIKM '18* (Torino, Italy). 497–506.

[43] Aston Zhang, Amit Goyal, Weize Kong, Hongbo Deng, Anlei Dong, Yi Chang, Carl A. Gunter, and Jiawei Han. 2015. AdaQAC: Adaptive Query Auto-Completion via Implicit Negative Feedback. In *SIGIR '15* (Santiago, Chile). Association for Computing Machinery, New York, NY, USA, 143–152. https://doi.org/10.1145/2766462.2767697

[44] Ke Zhou, Ronan Cummins, Mounia Lalmas, and Joemon M. Jose. 2012. Evaluating Aggregated Search Pages. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Portland, Oregon, USA) *(SIGIR '12)*. Association for Computing Machinery, New York, NY, USA, 115–124. https://doi.org/10.1145/2348283.2348302