A Social Search Model for Large Scale Social Networks

Yunzhong He*
Facebook
Seattle, Washtington
yunzhong@fb.com

Wenyuan Li*
University of California, Los Angeles
Los Angeles, California
liwenyuan.zju@gmail.com

Liang-Wei Chen
Facebook
Seattle, Washtington
benlwchen@fb.com

Gabriel Forgues
Facebook
Seattle, Washtington
gforgues@fb.com

Xunlong Gui
Facebook
Seattle, Washtington
xunlongg@fb.com

Sui Liang
Facebook
Seattle, Washtington
suiliang@fb.com

Bo Hou
Facebook
Seattle, Washtington
jsjhoubo@fb.com

ABSTRACT

With the rise of social networks, information on the internet is no longer solely organized by web pages. Rather, content is generated and shared among users and organized around their social relations on social networks. This presents new challenges to information retrieval systems. On a social network search system, the generation of result sets not only needs to consider keyword matches, like a traditional web search engine does, but it also needs to take into account the searcher's social connections and the content's visibility settings. Search ranking should also be able to handle both textual relevance and the rich social interaction signals from the social network.

In this paper, we present our solution to these two challenges by first introducing a social retrieval mechanism, and then investigate novel deep neural networks for the ranking problem. The retrieval system treats social connections as indexing terms, and generates meaningful results sets by biasing towards close social connections in a constrained optimization fashion. The result set is then ranked by a deep neural network that handles textual and social relevance in a two-tower approach, in which personalization and textual relevance are addressed jointly.

The retrieval mechanism is deployed on Facebook and is helping billions of users finding postings from their connections efficiently. Based on the postings being retrieved, we evaluate our two-tower neutral network, and examine the importance of personalization and textual signals in the ranking problem.

CCS CONCEPTS

• **Information systems** → **Social networks**; **Information retrieval**; **Specialized information retrieval**; *Learning to rank*; Search engine architectures and scalability.

KEYWORDS

information retrieval, social networks, social search, search ranking

1 INTRODUCTION

The popularization of social platforms has changed how content is organized on the internet, and has led to the diagram shift from Web 1.0 to Web 2.0 [41]. In Web 1.0, users passively consume content from web pages, whereas in Web 2.0, users participate in the creation and sharing of content. Compared to Web 1.0 content, content created on social networks (or Web 2.0) bears more personal aspects. For example, the content may be only visible to a small social group and could require more personal context to understand. User studies also show that compared to web search, users seek information differently on social networks, in which social relation plays a critical role [12, 31, 40].

In classic models of information retrieval systems (such as Web search engines), the main concern is about finding relevant documents based on search keywords. The user dimension is generally a secondary concern and is often introduced into the system at the re-ranking stage based on user profiles [10, 14, 37, 38]. Although this works well for web pages, it cannot handle the complex social structures of content created on social networks.

Some recent research introduces social relations into information retrieval models by factoring them into user profiles and creating personalized re-ranking models [5, 11, 43, 47] which are often used in microblog search. Among those studies, the social relationship between a user and a document is either used implicitly to construct a better preference-based user profile [11, 43, 47], or used explicitly as an additive term in the scoring function [5]. The focus on ranking and microblog search overlooks the fact that for large scale social networks, social relation is an important component to tractably generate the candidate result set to be ranked, due to the private nature of social network content. In addition, treating the social connection as an additive term in ranking offers simplicity but fails to capture the interactions between social signals and textual matches.

Inspired by its success in artificial intelligence, researchers start using deep learning to process personalization signals in recommendation systems [8, 17, 19, 45] and textual relevance in web search [23, 26, 32]. Some more recent studies also use deep learning for personalized commerce search [14, 18], which essentially creates

*Both authors contributed equally to this research.

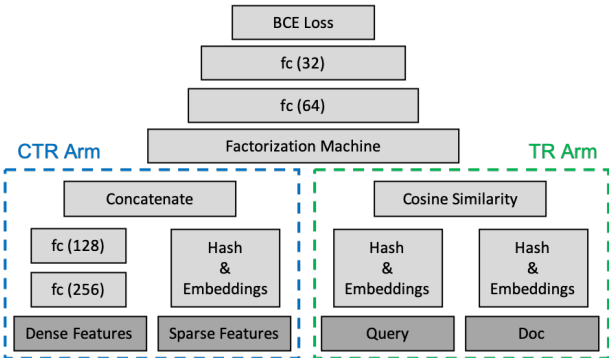

Figure 1: Architecture of our proposed two tower neural network.

neural networks that can jointly handle textual and contextual relevance (e.g. location, interest). Although to our best knowledge, no similar work has been done in the field of social network search, it is natural for us to consider deep learning for the ranking problem due to the rich social contexts behind the results being retrieved. In this paper, we offer the following contributions.

(1) We introduce a social retrieval mechanism that treats content retrieval as constrained optimization of query rewriting. We use it to help billions of users finding postings from their connections, and we discuss the trade-offs between various factors in this optimization problem using online performance.
(2) We explore novel neural network architectures that jointly model textual and social relevance for personalized ranking. Based on search results from our production retrieval system and neural network ranker, we examine the importance of different signals in the ranking problem.

2 RELATED WORKS

2.1 Social Search

Social search refers to the process of finding information based on social connections [1]. With the increase in user-generated content on the web, many researchers start to augment traditionally web search engines with social relations or collaborative behaviors for better search experiences [13, 30]. Another important area of social search is about searching content on social platforms such as Facebook[1] and Twitter[2], which we will refer to as social network search here. In a social network search, social relation plays a more important role than a collaborative web search. Some early works show that having social relations as a term in the ranking objective can significantly improve ranking quality in social network search [2, 5]. In more recent works, social relations are used to construct better user preference models to enhance personalized search ranking in microblog search [11, 43, 47]. The role of the user-author relationship in microblog search ranking is also discussed in [28, 39] with a focus on interests based similarity.

[1]facebook.com
[2]twitter.com

2.2 Personalized Ranking

Click-through rate (CTR) prediction is one of the tasks in information retrieval. It focuses on predicting the probability that content would be clicked if shown to the user. Traditionally, people use handcrafted features extracted from Bayesian [34] or feature selection methods [21, 27]. With the recent developments in deep learning, many CTR prediction methods utilize deep neural networks to reduce the amount of manual feature engineering. For example, neural collaborative filtering [20] uses a multi-layer perceptron to replace inner product for collaborative filtering. Wide and deep network [7] jointly trains wide linear models and deep neural networks to account for both memorizations and generalizations. Deep factorization machine [15] utilizes a factorization machine layer so that it can incorporate the power of factorization machines for the recommendation. Deep and cross network [46] further replaces the factorization machine layer with cross networks, making it more efficient in learning certain bounded-degree feature interactions.

2.3 Textual Relevance

Textual relevance is an extensively studied area in information retrieval which aims to model the semantic similarity between queries and documents [29]. Recent successes of neural methods in the field can be mainly categorized into representation-based and interaction-based methods [16]. Early attempts of neural textual relevance models, such as DSSM [24] and CDSSM [36], are mainly about learning good representations of the queries and documents, and the similarity measure is based on similarities of the representations. Interaction-based methods, on the other hand, directly model query-document matches at the token level. For example, Arc-II [22] uses 1-D convolutional layers to model interactions between two phrases. Match-SRNN [44] introduces the neural tensor layer to model complex interactions between input tokens. MatchPyramid [33] and PACRR [25] are inspired by the neural models for the image recognition task - they both view the matching matrix as a 2-D image and use convolutional neural networks to extract hierarchical matching patterns for relevance estimations.

3 SOCIAL RETRIEVAL

Different from keyword search over web pages, keyword search over a social network requires treating the social dimension as the first-class citizen because the content may be only visible to a small social group on a social network. In addition, social network content is generally more personal and less authoritative, making techniques based on document quality less effective. On a sparse social network like Facebook, without considering social connections in retrieval, the majority of results retrieved may not have any social connections to the searcher and would be either invisible[3], or irrelevant to the searcher. Even within the domain of all socially connected content, not all social connections are equally relevant. For example, there could be postings from a group that the user joined, but never visited, leading to irrelevant postings retrieved. Given these nuances, the problem of social retrieval is about biasing the retrieval space towards the most relevant connections. Several

[3]Due to user-selected privacy settings, a posting might only be visible to a limited audience. Visibility checks are always enforced on Facebook.

possible solutions for efficient social network content indexing have been explored in the past [3, 4, 6, 9], although their implications on large scale social networks are rarely discussed. In this section, we will use Unicorn [9] as our indexing system, and show how keyword search over Facebook postings can be achieved efficiently with its semantics and a few extended edge types.

3.1 Problem Definition

We denote a social network as a directed graph $< U, V >$, where U is the set of nodes, and V is the set of edges. On Facebook, a node can be of different types, such as *person*, *group*, *page*, or *posting*. The interactions between nodes are captured by edges. For example, a person can be a friend of another person, which we denote as $< person_1 \xrightarrow{\text{friend-of}} person_2 >\in V$. Similarly, a person can make a posting in a group, which we denote as $\{< posting_1 \xrightarrow{\text{authored-by}} person_1 >, < posting_1 \xrightarrow{\text{posted-in}} group_1 > \} \subseteq V$. Note that we're treating content (posting) and non-content (person, group, page) equally as nodes on a social graph. Since the focus of this paper is on content search, we do want to make the distinction between the two types of nodes. We will refer to person, group and page nodes as entities from now on. Therefore, we have $U = Entities \cup Postings$, $Entities = Persons \cup Pages \cup Groups$. For a searcher u, let us define its social connections as $Conn(u) = \{e \mid e \in Entities\ AND\ (< e, u >\in V\ OR\ < u, e >\in V\ OR\ u = e)\}$. And we can now formally define its socially connected postings as $ConnPostings(u) = \{p \mid p \in Postings\ AND\ \exists\, e \in Conn(u)\ s.t.\ (< e, p >\in V\ OR\ < p, e >\in V)\}$. Here, $ConnPostings(u)$ represents all the postings on a social network that are directly connected to entities that are within 1-degree of connections between the searcher. As discussed above, we may not want to consider all social connections as the result sets could be noisy, so we would like to filter down to the postings from a set of good social connections $GoodConn(u) \subseteq Conn(u)$, leading to the final retrieval space $GoodConnPostings(u) = \{p \mid p \in Postings\ AND\ \exists\, e \in GoodConn(u)\ s.t.\ (< e, p >\in V\ OR\ < p, e >\in V)\}$. Thus, the goal of social retrieval is to retrieve postings with keyword matches like traditional web search does, but restricted to the retrieval space of $GoodConnPostings(u)$ by enforcing a bias at query rewriting stage towards $GoodConn(u)$.

3.2 Unicorn

We use Unicorn [9] as our indexing system. In Unicorn, an edge on a social network is indexed as a prefixed term in a document's inverted index. For example, edge $< person_1 \xrightarrow{\text{friend-of}} person_2 >$ is indexed as friend:2 in the inverted index of $person_1$ [4]. To support keyword search on postings, we extended Unicorn with edge types as shown in Table 1, which captures various relationships a posting can have with a person, a group, and a page. As for text terms in a posting, they are indexed the same way as traditional web search engines do. For example, a posting about "Billie Eilish" will have the following indexing terms text:billie, text:eilish. [5]

[4] We denote entity an unique identifier x with $entitiy_x$

[5] Extensive studies have been done on optimizing the indexing policies for text terms, but we would like to keep it simple here as it is not the focus of this paper.

Edge-Type	Description
authored-by	A posting is authored by a person
involves	A posting involves a person in any way
page-of	A posting is posted in a page
group-of	A posting is posted in a group

Table 1: Extended edge types

During retrieval, we use the standard query language provided by Unicorn - s-expressions that support logical operators such as *AND*/*OR*. Suppose searcher $person_0$ has best friends $person_1$ and $person_2$, is an active member of $group_3$, and maintains a Facebook page $page_4$, then a possible set of good social connections could be $GoodConn(u) = \{0, 1, 2, 3, 4\}$. Thus the condition of enforcing posting from $GoodConn(u)$ can be expressed as the following query string.

```
(or
    involves:0
    authored-by:1
    authored-by:2
    group-of:3
    page-of:4
)
```

3.3 Social Query Rewriting

We have now formulated the problem of social retrieval as finding an optimal query string to enforce postings from $GoodConn(n)$. Because this is independent of the query string of enforcing text term matches, we will refer to this process as "social query rewriting". More specifically, for a searcher u, and a query query-keyword-match, we enforce the social conditions by rewriting it into a final query string of the form (and query-keyword-match query-social-match(u)). For example, after social query rewriting, the query string for keyword "Billie Eilish" will be the following

```
(and
    (or
        text:billie
        text:eilish
    )
    (or
        involves:0
        authored-by:1
        authored-by:2
        group-of:3
        page-of:4
    )
)
```

Now that we have formulated the problem of social retrieval as an optimal query rewriting problem, the next thing is to define the optimality condition and provide an optimization mechanism. Since we are optimizing for a candidate generation process, we want to focus more on recall rather than precision. However, higher recall generally requires a larger retrieval space to look at, leading to higher computational cost on the index servers. So finding an optimal query rewriting is essentially a constrained optimization that maximizes recall under a fixed budget of CPU cost.

3.4 Ground Truth Recall Set

Before diving into the final optimization formulation, we need to define a measure of the recall that we are trying to optimize. We sampled a ground truth set of 2k ideal results of the form $< query, searcher, ideal\ result >$ through rater data. The ideal result does not necessarily come from Search Engine Results Page (SERP), as raters may be searching a piece of content seen on a friend's profile. This way we can get rid of the presentation bias. We anonymized and featurized the dataset into the form $feature(searcher, author)$, which contains no user-identifiable information, but rather indicators of the social relationship between the user and the author of the ideal result. Table 2 illustrates the features we used. When a search request is issued, the information is pre-fetched as the request's meta-data from Facebook's graph database [42], which makes them readily available in real-time. For model training, we also sampled non-ideal results from their corresponding SERPs as negatives.

3.5 Constrained Optimization for Optimal Social Rewriting

Given the small size of training data and feature space, we use a linear model to optimize the connection terms. For each social prefix p, we want to solve for the optimal social feature weights w_p^*, and a threshold t_p^* that maximizes the recall in the ground truth set D_G under a fixed CPU cost k. More formally,

$$\begin{aligned} \underset{w_p, t_p}{\text{maximize}} \quad & recall_{D_G}(expr(w_p, t_p)) \\ \text{subject to} \quad & cpu\_cost(expr(w_p, t_p)) < k \end{aligned}$$

in which $expr(w_p, t_p)$ is the s-expression parameterized by feature weights w_p of a linear model that selects the best connection terms, and a bound on the number of social connections t_p.

To solve for t_p^*, we assume an uniform retrieval space incurred by each social prefix (i.e. each friend, group and page can provide as many results matching the text constraints as all other friends, groups and pages do, respectively), which makes the CPU cost from each prefix effectively a function of the number of connections with the prefix. This assumption is an oversimplification as some groups may contain more postings than others. However, we found t_p to be the dominating factor of CPU cost when it is aggregated over all search sessions. Since we budget CPU cost at an aggregated level, this is in fact a valid assumption. And to maximize the recall, we simply ran parameter sweep experiments online and obtained the maximal t_p under budget k.

As we fix t_p, this constrained optimization becomes a simple maximization of $recall_{Conn(D_G)}@t_p^*(LM(w_p))$. Namely, we want to maximize the top t_p^* recall of the connections that a posting comes from in the ground truth set D_G, when the connections are ranked by a linear model parametrized by weights w_p. We then solve w_p^* using linear regression.

Essentially, what this constrained optimization does is approximate the optimal connection space $GoodConn(u)$, using a ground truth dataset sampled from $GoodConnPostings(u)$. And it is done through first fixing its size, and then selecting its most likely members. The weights and thresholds are computed offline, but the inference on optimal connections is performed online because of the dynamic nature of social connections.

Feature
Whether this user/page/group is recently visited
Time since last visit of this user/page/group
Whether this is a liked page
Whether this is a joined group
Social network coefficient between the searcher and author

Table 2: Social features for query rewriting

Method	CPU	CTR(friend)	CTR(group/page)
Recency	-	-	-
Social Coef	(p>0.05)	0.32% (p =1.2e-4)	0.94% (p =6.8e-5)
Linear Model	(p>0.05)	**0.56% (p =1.2e-6)**	**2.07% (p =3.3e-5)**

Table 3: Post-launch performance of social query rewriting

3.6 Results

We deployed this rewriting system on Facebook search, and it is now powering the retrieval of postings from a searcher's connections. The retrieval system had evolved from naively including postings from all friends and groups, to a heuristics-based approach (i.e. order by interaction recency) that also enables postings from followed pages, and to the constrained optimization approach that allows more general connection types such as liked or recently visited pages, and achieved good quality improvement with even CPU cost savings. Table 3 shows the relative improvements from its post-launch backtesting compared against other approaches, where we rank connections based on the recency in the searcher's last interaction as the baseline. We report the overall CTR on postings from friend and group/page. Note that for this backtesting we fix t_p for a fair evaluation of social rewriting quality, and all of the results were ranked by our production ranking model before being displayed to the searchers.

From Table 3 we can see that CPU cost does not differ significantly between different methods, confirming our hypothesis that CPU cost is a function of t_p. And our linear model indeed outperforms any heuristics-based methods like interaction recency or social graph coefficient. We believe that being able to aggregate different real-time interactions from a social network is an advantage of this online linear model, and is where its gain comes from. We have learned from this backtesting and other past experiments that, in general, strong real-time signals like recent interactions can outperform more sophisticated offline models, while the best performance was often achieved when the two were combined.

4 TWO-TOWER NEURAL NETWORK FOR PERSONALIZED RANKING

After a set of socially connected postings are retrieved, they need to be ordered based on their relevance for the best user experience. Since content on a social network is highly personal, social relevance should be factored into the ranking objective as much as textual relevance. Although social relevance was considered in the ranker from section 3.5, a key difference here is that the aforementioned ranker only ranks the connections to retrieve from but cannot compare the relevance between two postings to display.

Therefore, we want to explore result-level ranking models that can jointly handle social and textual relevance here.

4.1 Multi-stage Ranking

Because result-level ranking can have non-trivial CPU cost, we use a multi-stage ranking in our production system. Our stage 1 ranker is a gradient-boosting decision tree (GBDT) model that uses standard textual relevance features like BM25 [35], as well as social relevance features. And the stage 2 ranker is a neural network that leverages more sparse features on top of the stage 1 features, using a DLRM-like [17] architecture. We observed a big search quality improvement when we launched the stage 2 neural network ranker, even without the additional sparse features. For the sake of space, we will focus our discussions on the architecture explorations for the neural network ranker here. And the exploration is based on our production neural network architecture.

4.2 Motivation

While extensive studies have been done on neural IR models, the focus has been on learning better text representations and query-document interactions [16, 24, 25, 29, 36]. In the settings of personalized web search, personalization signals generally come from historical search behaviors, and the focus is usually on preference-based user profiling [10, 11, 37, 38]. In social network search, however, personalization signals are so much richer and contain an extra dimension of social relations. This motivates us to look at the click-through rate (CTR) models from recommendation systems, as the scope of personalization is closer to CTR prediction problems rather than traditional web search ranking. However, textual relevance does still play an important role in ranking, so naturally, to combine the best of the two, we proposed a two-tower approach which contains a CTR model trained from personalization features, and a textural relevance model trained from n-grams of query and document tokens. In addition, we decided to train the model with click data, not only because of its availability but also because any 3rd-party labeled data cannot capture the social context of the original searcher.

4.3 Two-Tower Architecture

The overall architecture of our two-tower network can be expressed as follows.

$$\Phi(q, d, u) = g(NN_{tr}(\psi(q), \eta(d)), NN_{ctr}(f_d(u, d), f_s(u, d)))$$

where NN_{tr} is a typical query-doc textural relevance (TR) model, modeling the semantic similarity between a query q, and a document d, based on their embedding representation $\psi(q)$ and $\eta(d)$. NN_{ctr}, on the other hand, is a CTR model capturing the contextual relevance between a user u and the document d using a hand-tuned user-doc dense feature vector, $f_d(u, d)$, and sparse feature vector $f_s(u, d)$. The output of NN_{tr} and NN_{ctr} are both vectors, which we denote as x_{tr} and x_{ctr}. Note that x_{tr} is a vector because each dimension of x_{tr} represents the cosine similarity between one type of query n-gram (e.g. query bi-gram, query tri-gram) and one type of document n-gram (e.g. doc-title-trigram, doc-body-bigram). Finally, x_{tr} and x_{xtr} are concatenated into vector $x = x_{tr} \cdot x_{ctr}$ as the input to a final matrix factorization layer, to represent feature-feature interactions. So we have

Feature	Type
If the post is authored by the searcher	user-doc
If the post comes from a friend	user-doc
If the post comes from a joined group	user-doc
If the post comes from a followed page	user-doc
If the searcher has recently seen the post	user-doc
Global click count from SERP of a post	doc-side
If the post contains a photo	doc-side
Age of a post	doc-side
Number of comments on a post	doc-side
Number of friends of the searcher	user-side
Number of followees of the searcher	user-side
Number of SERP impressions of the searcher	user-side
Region id of the searcher	user-side, sparse
City id of the searcher	user-side, sparse

Table 4: Hand-crafted features used in the CTR arm

$$\Phi(q, d, u) = g(x_{tr}, x_{ctr}) = w^T x + x^T upper(VV^T)x$$

Note that different from standalone TR or CTR models, there isn't any softmax layer in NN_{tr} or NN_{ctr} to normalize the final vector representation into a probability value. Instead, we use the vector representation directly into a matrix factorization layer. Although by design the two-tower architecture could support any TR and CTR models, in practice we use DLRM [17] as our CTR tower, and a simple cosine similarity model as our TR tower, as illustrated in Figure 1. The neural network is trained with click data, which is a collection of pairs $< q, u, d, y >$, in which $y \in \{1, 0\}$ representing a user click ($y = 1$), or no click ($y = 0$). Therefore the neural network can be trained by minimizing a BCE loss just like a neural factorization machine [15].

4.4 Representation Details

To learn the query representation $\psi(q)$ and document representation $\eta(d)$, we extract n-grams from their raw texts, and feed them into randomly initialized embedding lookup matrices. Note that $\psi(q)$ and $\eta(d)$ don't share the same embeddings, as we believe queries and documents have different textual characteristics.

To encode signals outside of textual relevance, we use a feature vector $f_d(u, d)$ with around 30 dimensions in our experiments. $f_d(u, d)$ is essentially a concatenation of many hand-tuned features we designed, and those features can be categorized as user-side features, doc-side features, and user-doc features. We also use two user-side sparse features for better personalization, encoded as sparse vectors in $f_s(u, d)$. Table 4 illustrates those features.

5 EXPERIMENTS

5.1 Dataset

Given the personal nature of social content search, it is difficult to create any labeled dataset to evaluate our models. So instead, we use clicks sampled from real search traffic for model training and evaluation. When a search query is issued, for the results shown, their n-gram features from raw texts, as well as the feature vectors $f_d(u, d)$ and $f_s(u, d)$, are computed online. They are then sampled and logged along with any click activities on them as our training and evaluation data. Any user identifiable information is omitted in

Feature
BM25
AVG TF-IDF
Position of the last matched term

Table 5: Textual Relevance Features

the final dataset. So the dataset is a collection of <session-id, hashed n-grams, $f_d(u,d)$, $f_s(u,d)$>, where session-id is a unique identifier of the search session, which can contain many documents displayed. We sampled 120 millions of data from a 30-day window for model training. To mimic the production behavior, the evaluation set is sampled from some dates after the 30-day window of training data, containing 6 million rows in total, as we find empirically that evaluating on future data yields a closer estimation of a model's online performance.

5.2 Hand-tuned Textual Relevance Features

To evaluate how our models can learn textual relevance end to end from raw texts, we adopt several popular query-doc relevance features from information retrieval for comparison. Table 5 shows the textual relevance features we used. For simplicity we will refer to features in Table 4, features in Table 5, and raw text n-grams as CTR features, TR features, and n-gram features respectively.

5.3 Experiment Setup

To investigate how social/personalization signal and textual relevance signal can influence the final search ranking results, we train and evaluate our model under the following settings: (1) DLRM with CTR features only; (2) DLRM with TR features only; (3) DLRM with CTR + TR features; (4) cosine similarity with n-gram features only; (5) two-tower with CTR + n-gram features (6) two-tower with CTR + TR + n-gram features.

Note that under our two-tower design, the CTR arm is essentially a DLRM model without a final softmax layer, so experiment (1) (2) and (3) are essentially the NN_{ctr} tower adding a softmax layer to generate a probability value for ranking. On the other hand, the textual relevance arm adopts a DSSM structure with two heads to process query and document respectively. Setting (4) uses textual relevance arm only. Similarly, we also add a softmax layer for ranking in (4). Setting (5) and (6) use two-tower structures to process social/personalization and textual relevance signals. The only difference is that in setting (6), we also use TR features in the CTR arm, so that it has the richest signals. A detailed model architecture of our proposed method can be found in Table 6.

Our model is implemented in Pytorch and trained on 2 Tesla M40 GPU. We use an Adam optimizer with the initial learning rate of 0.01. We also use batch normalization to ease the training, and add a dropout layer after each fully connected layer for regularization.

5.4 Evaluation

We evaluate our model performance on 0.1 million search results seen by the users, where clicked results are labeled as positives, and non-clicks as negatives. We use ROC-AUC and Normalized Discounted Cumulative Gain (NDCG) as the evaluation metrics here. See **Table 7** for the detailed numbers.

CTR and TR features: We first focus on evaluating the effectiveness of CTR and TR features in the context of search ranking. We evaluate the features using a DLRM model, which is effectively our two-tower model without the TR arm. We compare the model performance using social features, TR features, and CTR+TR features. The model with the lowest AUC/NDCG uses only TR features, which indicates that manually designed textual relevance features are not expressive enough for a social search problem. Adding CTR and features further improves the AUC from 59.01% to 62.16% and NDCG from 80.66% to 81.55%. Overall, we observe that personalization signals are important for the social search ranking model.

N-gram Features: Next, we experiment with text n-gram features using a simple DSSM-like model, in which each n-gram will have its embeddings. Using n-gram features alone outperforms TR features by at 1.7% AUC, likely because embeddings improve the expressiveness of the representations. For example, for query "kitten" and a document titled "cat videos", traditional TR features may consider it as a total mismatch. However, such similarity could be modeled in the space of text embeddings.

Two-tower architecture: Finally, we test our proposed two-tower method using CTR + n-gram features. Our proposed model achieves 64.49% AUC, which is another 2.0%+ ROC-AUC increase over any of the single tower approaches (with the highest NDCG). That is, combining social and textual relevance signals with embeddings yields the best performance. In addition, combining the n-gram features with TR features further improves AUC from 64.49% to 65.02%.

Overall, we conclude that social/personalization and textual relevance signals are both important for post ranking, and manually engineered textual relevance features are not as effective as text embeddings learned from n-grams. Our proposed two-tower architecture, which jointly models personalization and textual relevance, yields the best ranking based on ROC-AUC and NDCG of click prediction data.

6 CONCLUSION

In this paper, we present the challenges of social search on large scale social networks such as Facebook. We argue that it is important to tractably generate a meaningful result set is to bias retrieval towards a set of good connections, and we solve this through a constrained optimization problem of query rewriting. We also demonstrate the effectiveness of this approach by comparing online performance across different methods.

Finally, we introduce a neural learning-to-rank model, which jointly learns personalization and textual relevance, in the social network search domain. We argue that the scope of personalization signals in a social network search is similar to recommendation systems, while textual relevance still plays an important role as it is in classic information retrieval models. So we propose a two-tower neural network that combines the best of the two, and our experiments show that it indeed outperforms state-of-the-art models like DLRM, with substantial gains coming from our two-tower design.

REFERENCES

[1] 2016. Social Networks and Information Retrieval, How Are They Converging? A Survey, a Taxonomy and an Analysis of Social Information Retrieval Approaches

	CTR Arm		Textual Relevance Arm	
Input	Dense Features	Sparse Features	Query	Document
Model	2-Layer MLP: fc(256), batchnorm, relu dropout (0.2) fc(128), batchnorm, relu	Hash & Embedding	Hash & Embedding	Hash & Embedding
	Concatenate		Cosine Similarity	
	Factorization Machine 3-Layer MLP: fc(64), batchnorm, relu dropout (0.2) fc(32), batchnorm, relu fc(1)			
Output	Logistic Layer (click probability)			

Table 6: Detailed model architecture.

Inputs	ROC-AUC	NDCG
CTR dense features	61.74%	81.30%
TR dense features	59.01%	80.66%
CTR + TR	62.16%	81.55%
n-gram sparse features	60.81%	80.80%
CTR + n-gram	**64.49%**	**81.67%**
CTR + TR + n-gram	**65.02%**	**81.98%**

Table 7: Model performance under different settings.

and Platforms. *Inf. Syst.* 56, C (March 2016), 1–18. https://doi.org/10.1016/j.is.2015.07.008

[2] Matthias Bender, Tom Crecelius, Mouna Kacimi, Sebastian Michel, Thomas Neumann, Josiane Xavier Parreira, Ralf Schenkel, and Gerhard Weikum. 2008. Exploiting social relations for query expansion and result ranking. *2008 IEEE 24th International Conference on Data Engineering Workshop* (2008), 501–506.

[3] Truls A. Bjørklund, Michaela Götz, Johannes Gehrke, and Nils Grimsmo. 2011. Workload-aware Indexing for Keyword Search in Social Networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11)*. ACM, New York, NY, USA, 535–544. https://doi.org/10.1145/2063576.2063656

[4] Mohamed Reda Bouadjenek, Hakim Hacid, Mokrane Bouzeghoub, and Athena Vakali. 2013. Using social annotations to enhance document representation for personalized search. 1049–1052. https://doi.org/10.1145/2484028.2484130

[5] David Carmel, Naama Zwerdling, Ido Guy, Shila Ofek-Koifman, Nadav Har'el, Inbal Ronen, Erel Uziel, Sivan Yogev, and Sergey Chernov. 2009. Personalized Social Search Based on the User's Social Network. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. ACM, New York, NY, USA, 1227–1236. https://doi.org/10.1145/1645953.1646109

[6] H. Chen and H. Jin. 2018. Efficient Keyword Searching in Large-Scale Social Network Service. *IEEE Transactions on Services Computing* 11, 5 (Sep. 2018), 810–820. https://doi.org/10.1109/TSC.2015.2464819

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 7–10.

[8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 7–10. https://doi.org/10.1145/2988450.2988454

[9] Michael Curtiss, Iain Becker, Tudor Bosman, Sergey Doroshenko, Lucian Grijincu, Tom Jackson, Sandhya Kunnatur, Soren Lassen, Philip Pronin, Sriram Sankar, Guanghao Shen, Gintaras Woss, Chao Yang, and Ning Zhang. 2013. Unicorn: A System for Searching the Social Graph. *Proc. VLDB Endow.* 6, 11 (Aug. 2013), 1150–1161. https://doi.org/10.14778/2536222.2536239

[10] Mariam Daoud, Lynda Tamine-Lechani, Mohand Boughanem, and Bilal Chebaro. 2009. A Session Based Personalized Search Using an Ontological User Profile. In *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC '09)*. ACM, New York, NY, USA, 1732–1736. https://doi.org/10.1145/1529282.1529670

[11] Amna Dridi and Yahya Slimani. 2017. Leveraging social information for personalized search. *Social Network Analysis and Mining* 7, 1 (26 Apr 2017), 16. https://doi.org/10.1007/s13278-017-0435-4

[12] Brynn M. Evans and Ed H. Chi. 2008. Towards a Model of Understanding Social Search. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work (CSCW '08)*. ACM, New York, NY, USA, 485–494. https://doi.org/10.1145/1460563.1460641

[13] Fernando Figueira Filho, Gary M. Olson, and Paulo Lício de Geus. 2010. Kolline: A Task-oriented System for Collaborative Information Seeking. In *Proceedings of the 28th ACM International Conference on Design of Communication (SIGDOC '10)*. ACM, New York, NY, USA, 89–94. https://doi.org/10.1145/1878450.1878465

[14] Mihajlo Grbovic and Haibin Cheng. 2018. Real-time Personalization Using Embeddings for Search Ranking at Airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. ACM, New York, NY, USA, 311–320. https://doi.org/10.1145/3219819.3219885

[15] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-machine Based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press, 1725–1731. http://dl.acm.org/citation.cfm?id=3172077.3172127

[16] 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. *arXiv preprint arXiv:1903.06902* (2019).

[17] Udit Gupta, Xiaodong Wang, Maxim Naumov, Carole-Jean Wu, Brandon Reagen, David Brooks, Bradford Cottel, Kim M. Hazelwood, Bill Jia, Hsien-Hsin S. Lee, Andrey Malevich, Dheevatsa Mudigere, Mikhail Smelyanskiy, Liang Xiong, and Xuan Zhang. 2019. The Architectural Implications of Facebook's DNN-based Personalized Recommendation. *CoRR* abs/1906.03109 (2019). arXiv:1906.03109 http://arxiv.org/abs/1906.03109

[18] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C. Turnbull, Brendan M. Collins, and Thomas Legrand. 2019. Applying Deep Learning to Airbnb Search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. ACM, New York, NY, USA, 1927–1935. https://doi.org/10.1145/3292500.3330658

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 173–182. https://doi.org/10.1145/3038912.3052569

[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 173–182.

[21] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 1–9.

[22] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. 2042–2050.

[23] 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 *Proceedings of the 22Nd ACM International Conference on*

Information & Knowledge Management (CIKM '13). ACM, New York, NY, USA, 2333–2338. https://doi.org/10.1145/2505515.2505665

[24] 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 *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2333–2338.

[25] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A position-aware neural IR model for relevance matching. *arXiv preprint arXiv:1704.03940* (2017).

[26] Aaron Jaech, Hetunandan Kamisetty, Eric K. Ringger, and Charlie Clarke. 2017. Match-Tensor: a Deep Relevance Model for Search. *CoRR* abs/1701.07795 (2017). arXiv:1701.07795 http://arxiv.org/abs/1701.07795

[27] Michael Jahrer, A Toscher, Jeong-Yoon Lee, J Deng, Hang Zhang, and Jacob Spoelstra. 2012. Ensemble of collaborative filtering and feature engineered models for click through rate prediction. In *KDDCup Workshop*.

[28] Y. Jiang, Y. Xu, and L. Shao. 2016. A Personalized Microblog Search Model Considering User-Author Relationship. In *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*. 508–513. https://doi.org/10.1109/DSC.2016.91

[29] Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509* (2017).

[30] Meredith Ringel Morris and Eric Horvitz. 2007. SearchTogether: An Interface for Collaborative Web Search. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 3–12. https://doi.org/10.1145/1294211.1294215

[31] Anne Oeldorf-Hirsch, Brent Hecht, Meredith Ringel Morris, Jaime Teevan, and Darren Gergle. 2014. To Search or to Ask: The Routing of Information Needs Between Traditional Search Engines and Social Networks. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. ACM, New York, NY, USA, 16–27. https://doi.org/10.1145/2531602.2531706

[32] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep Sentence Embedding Using Long Short-term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 24, 4 (April 2016), 694–707. https://doi.org/10.1109/TASLP.2016.2520371

[33] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[34] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 521–530.

[35] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (April 2009), 333–389. https://doi.org/10.1561/1500000019

[36] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 373–374.

[37] Ahu Sieg, Bamshad Mobasher, and Robin Burke. 2007. Web Search Personalization with Ontological User Profiles. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management (CIKM '07)*. ACM, New York, NY, USA, 525–534. https://doi.org/10.1145/1321440.1321515

[38] Micro Speretta and Susan Gauch. 2005. Personalized Search Based on User Search Histories. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI '05)*. IEEE Computer Society, Washington, DC, USA, 622–628. https://doi.org/10.1109/WI.2005.114

[39] S. A. Tabrizi, A. Shakery, M. A. Tavallaei, and M. Asadpour. 2019. Search Personalization Based on Social-Network-Based Interestedness Measures. *IEEE Access* 7 (2019), 119332–119349. https://doi.org/10.1109/ACCESS.2019.2935425

[40] Jaime Teevan, Daniel Ramage, and Merredith Ringel Morris. 2011. #TwitterSearch: A Comparison of Microblog Search and Web Search. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining (WSDM '11)*. ACM, New York, NY, USA, 35–44. https://doi.org/10.1145/1935826.1935842

[41] O'Reilly Tim. 30.09.2005. What Is Web 2.0. - O'Reilly Media. (30.09.2005). http://oreilly.com/web2/archive/what-is-web-20.html

[42] Venkateshwaran Venkataramani, Zach Amsden, Nathan Bronson, George Cabrera III, Prasad Chakka, Peter Dimov, Hui Ding, Jack Ferris, Anthony Giardullo, Jeremy Hoon, Sachin Kulkarni, Nathan Lawrence, Mark Marchukov, Dmitri Petrov, and Lovro Puzar. 2012. TAO: How Facebook Serves the Social Graph. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12)*. ACM, New York, NY, USA, 791–792. https://doi.org/10.1145/2213836.2213957

[43] Jan Vosecky, Kenneth Wai-Ting Leung, and Wilfred Ng. 2014. Collaborative Personalized Twitter Search with Topic-language Models. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '14)*. ACM, New York, NY, USA, 53–62. https://doi.org/10.1145/2600428.2609584

[44] Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378* (2016).

[45] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1235–1244. https://doi.org/10.1145/2783258.2783273

[46] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. ACM, 12.

[47] Yukun Zhao, Shangsong Liang, and Jun Ma. 2016. Personalized Re-ranking of Tweets. In *Web Information Systems Engineering – WISE 2016*, Wojciech Cellary, Mohamed F. Mokbel, Jianmin Wang, Hua Wang, Rui Zhou, and Yanchun Zhang (Eds.). Springer International Publishing, Cham, 70–84.