Neural Ranking Models with Multiple Document Fields

Hamed Zamani*
University of Massachusetts Amherst
zamani@cs.umass.edu

Bhaskar Mitra†
Microsoft and UCL
bmitra@microsoft.com

Xia Song
Microsoft
xiaso@microsoft.com

Nick Craswell
Microsoft
nickcr@microsoft.com

Saurabh Tiwary
Microsoft
satiwary@microsoft.com

ABSTRACT

Deep neural networks have recently shown promise in the *ad-hoc retrieval* task. However, such models have often been based on one field of the document, for example considering document title only or document body only. Since in practice documents typically have multiple fields, and given that non-neural ranking models such as BM25F have been developed to take advantage of document structure, this paper investigates how neural models can deal with multiple document fields. We introduce a model that can consume short text fields such as document title and long text fields such as document body. It can also handle multi-instance fields with variable number of instances, for example where each document has zero or more instances of incoming anchor text. Since fields vary in coverage and quality, we introduce a masking method to handle missing field instances, as well as a field-level dropout method to avoid relying too much on any one field. As in the studies of non-neural field weighting, we find it is better for the ranker to score the whole document jointly, rather than generate a per-field score and aggregate. We find that different document fields may match different aspects of the query and therefore benefit from comparing with separate representations of the query text. The combination of techniques introduced here leads to a neural ranker that can take advantage of full document structure, including multiple instance and missing instance data, of variable length. The techniques significantly enhance the performance of the ranker, and also outperform a learning to rank baseline with hand-crafted features.

KEYWORDS

Neural ranking models, representation learning, document representation, deep neural networks

*Work done while at Microsoft.
†The author is a part-time PhD student at UCL.

WSDM '18, February 5–9, 2018, Los Angeles, CA.

DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep neural networks have shown impressive performance in many machine learning tasks, including information retrieval models for ranking documents [5, 7, 8, 15, 23, 27]. These deep neural ranking models (NRMs) often consider a single source of document description, such as document title [8, 23] or body text [5, 15]. However, in many retrieval scenarios, additional sources of document descriptions may be available. For instance in web search, each document consists of text fields specified by the document's HTML tags, such as title and body, as well as external sources of meta-information, such as the anchor text from incoming hyperlinks or the query text for which the document has been previously viewed.

Learning a document representation suitable for retrieval tasks can be challenging when multiple document fields should be considered. These challenges primarily stem from the distinct properties of these diverse field types: (i) while the body of a web page is often long, the content of many other fields, such as title, are typically only a few terms in length, (ii) while some fields (e.g., body) contain a single instance of text, other fields may contain bags of multiple short texts (e.g., anchor text), (iii) multi-instance fields generally contain variable number of instances, e.g., zero or more instances of incoming anchor text for a given document, (iv) some fields, such as URL, may not contain natural language text, and finally (v) fields vary in coverage and accuracy, for example a field that memorizes past queries that led to a click on the document may provide a very useful (high-accuracy) ranking signal [1], but the coverage of that field may be relatively low because not every document has been clicked before. Each of these challenges increases the complexity of the representation learning task for documents with multiple fields. However, multiple fields associated with each document may contain complementary information that has motivated us to learn representation for documents by considering multiple fields in order to improve the retrieval performance.

In this paper, we propose NRM-F[1], a *general* framework for learning multiple-field document representation for ad-hoc retrieval. NRM-F is designed to address the aforementioned challenges. More specifically, NRM-F can handle multiple fields, both with single and multiple instances. In NRM-F, although the neural network parameters are shared among multiple instances of the same field, they are distinct across fields. This enables NRM-F to uniquely model the content of each field based on its specific characteristics.

[1]The naming is inspired by BM25F [21].

We employ the same topology for the sub-networks corresponding to the different fields. However, there are a number of controlling hyper-parameters that determine the exact sub-network configuration for each field. We introduce field-level masking to better cope with variable length inputs, i.e., fields with variable number of text instances. We also propose a novel field-level dropout technique that effectively regularizes the network and prevents it from over-dependence on high-accuracy fields, such as clicked queries. Given the intuition that different fields may match different aspects of the query, our model learns different query representations corresponding to different document fields.

We evaluate our models in the context of web search, using the queries sampled from the Bing's search logs. We study five fields in our experiments: title (single short text), body (single long text), URL (single short text, but not in a natural language), anchor texts (multiple short texts), and clicked queries (multiple short texts providing a ranking signal with relatively high accuracy). We consider this effective and diverse set of fields to make our findings more likely to generalize to other combinations of document fields.

In this work, we study the following research hypotheses:

H1 The ad-hoc retrieval performance of NRM-F improves as we incorporate multiple document fields.

H2 NRM-F performs better than competitive baselines, such as term matching and learning to rank.

H3 Learning a multiple-field document representation is superior to scoring based on individual field representations and summing.

H4 Learning per-field query representations performs better than learning a single query representation.

H5 The additional techniques of field-level masking and field-level dropout yield additional performance improvements.

Our experiments validate all these hypotheses, and investigate the effectiveness of our overall NRM-F framework.

2 RELATED WORK

2.1 Retrieval with Multiple Fields

Information retrieval tasks may involve semi-structured data, meaning that the text of each document is divided into sections. Given a sufficiently fine-grained structure, some past research has studied the retrieval of the particular sections that best satisfy the user's query, such as in the INEX XML retrieval initiative [6, 12]. In web search it is more typical to consider coarse-grained sections such as title and body, also referred to as *fields*, and use them to generate features in a document ranking task.

Using evidence from structure to improve document retrieval is well studied in information retrieval. Wilkinson [26] proposed a number of hypotheses about how to combine section-level and document-level evidence. For example, taking the maximum section score, or a weighted sum of section scores, and then potentially combining with a document-level score. Robertson et al. [21] further proposed BM25F, an extension to the original BM25 model [22], arguing that the linear combination of field-level scores is "dangerous", because it bypasses the careful balance across query terms in the BM25 model. The BM25F solution is to first combine frequency information across fields on a per-term basis, then compute a retrieval score using the balanced BM25 approach.

There are a number of alternative approaches to BM25F for the multiple-field document retrieval task. For instance, Piwowarski and Gallinari [19] proposed a model based on Bayesian networks for retrieving semi-structured documents. Myaeng et al. [16] extended the InQuery retrieval system to semi-structured documents. Svore and Burges [25] proposed a supervised approach, called LambdaBM25, that learns a BM25-like retrieval model based on the LambdaRank algorithm [3]. LambdaBM25 can also consider multiple document fields, without resorting to a linear combination of per-field scores. Dealing with multiple document fields without a linear combination was also studied by Ogilvie and Callan [18], who proposed and tested various combinations for a known-item search task, using a language modeling framework. Kim et al. [9] proposed a probabilistic model for the task of XML retrieval. Later on, Kim and Croft [10] introduced a model based on relevance feedback for estimating the weight of each document field.

2.2 Neural Networks for Ranking

Several recent studies have applied deep neural network methods to various information retrieval applications, including question answering [29], click models [2], ad-hoc retrieval [5, 15, 27], and context-aware ranking [30]. Neural ranking models can be partitioned into early and late combination models [5]. They can also be categorized based on whether they focus on lexical matching or learning text representations for semantic matching [15].

The early combination models are designed based on the interactions between query and document as the networks' input. For instance, the deep relevance matching model [7] gets histogram-based features as input, representing the interactions between query and document. DeepMatch [13] is another example that maps the input to a sequence of terms and computes the matching score using a feed-forward network. The local component of the duet model in [15] and the neural ranking models proposed in [5, 27] are the other examples for early combination models.

The late combination models, on the other hand, separately learn a representation for query and document and then compute the relevance score using a matching function applied on the learned representations. DSSM [8] is an example of late combination models that learns representations using feed-forward networks and then uses cosine similarity as the matching function. DSSM was further extended by making use of convolutional neural networks, called C-DSSM [23]. The distributed component of the duet model [15] also uses a similar architecture for learning document representation. We refer the reader to [14] that provides an overview of various (deep) neural ranking models.

In all of the aforementioned work, each document is assumed to be a single instance of text (i.e., single field). However, documents often exist in a semi-structured format. In this paper, we focus on late combination models and propose a neural ranking model that takes multiple fields of document into account. Given the hypothesis provided in [15], our neural model can be further enriched by making use of lexical matching in addition to distributed matching. We leave the study of lexical matching for the future and focus on document representation learning.

3 THE NRM-F FRAMEWORK

In this section, we first provide our motivation for studying the task of representation learning for documents with multiple fields, and formalize the task. We then introduce a high-level overview of our framework, and further describe how we implement each component of the proposed framework. We finally explain how we optimize our neural ranking model.

3.1 Motivation and Problem Statement

In many retrieval scenarios, there exist various sources of textual information (*fields*) associated with each document d. In web search in particular, these sources of information can be partitioned into three categories. The first category includes the information provided by the structure and the content of document d itself. Different elements of the web page specified by the HTML tags, e.g., title, header, keyword, and body, as well as the URL are examples of fields of this type. The second category includes the information provided by the other documents for representing d. For instance, when there is a hyperlink from document d' to d, the corresponding anchor text may provide useful description of d. The third category contains information that we can infer from interactions between the retrieval system and its users. For instance in web search, when a user clicks on the document d for a query q, the text of query q can be used to describe d. Svore and Burges [25] refer to these last two categories as popularity fields.

There are several previous studies showing that different fields may contain complementary information [21, 25]. Therefore, incorporating multiple fields can lead to more accurate document representation and better retrieval performance. For example, clicked queries are highly effective for the retrieval tasks [1, 25, 28]. A number of prior studies [21, 25], have also investigated the usefulness of anchor texts for web search. However, for fresh or less popular documents that may not have enough anchor or clicked query text associated with them, the body text provides important description of the document. Similarly, the URL field may be useful for matching when the query expresses an explicit or implicit intent for a specific domain. These complementary and diverse sources of textual descriptions have motivated us to study representation learning for ad-hoc retrieval by incorporating multiple fields.

The unique properties of these diverse document fields, however, make it challenging to model them within the same neural architecture. For example, the vocabulary and the language structure of clicked queries may be distinct from those of the body text, and in turn both may be distinct from the URL field. The document body text may contain thousands of terms, while the text in other fields may be only few terms in length. Finally, a key challenge also stems from the fact that a number of fields consist of multiple instances. For example, there are multiple anchor texts for each document d, and multiple queries can be found that previously led users to click on document d. A neural ranking model that considers these fields for document ranking must handle variable number of text instances per document field. To formulate the task, let $\mathcal{F}_d = \{F_1, F_2, \cdots, F_k\}$ denote a set of fields associated with the document d. Each field F_i consists of a set of instances $\{f_{i1}, f_{i2}, \cdots, f_{im_i}\}$ where m_i denotes the number of instances in the field F_i. The task is to learn a function $\Phi_D(\mathcal{F}_d)$ whose output is a representation for document d, suitable for the ad-hoc retrieval task.

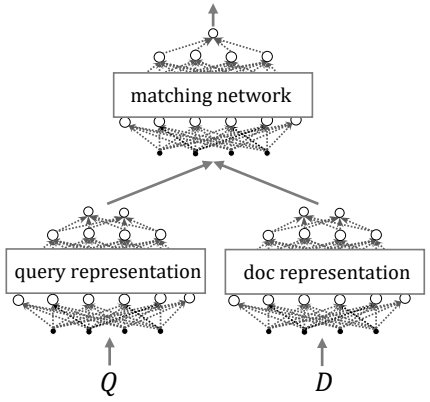

Figure 1: A neural ranking model architecture that consists of three major components: query representation, document representation, and matching network.

3.2 High-Level Overview of the Framework

In this paper, as shown in Figure 1, we focus on a late-combination and representation-focused neural ranking model. This architecture consists of three major components: document representation (Φ_D), query representation (Φ_Q), and the matching network (Ψ) which takes both representations and computes the retrieval score (i.e., score $= \Psi(\Phi_Q, \Phi_D)$). In this section, we describe the high-level architecture used for the document representation network, which is the focus of the paper. Sections 3.7 and 3.8 review how the query representation and matching network components are respectively implemented.

To learn multiple-field document representation (i.e., Φ_D), the framework first learns a representation for each individual instance in a field. The framework then aggregates these learned vector representations to represent the field as a whole. It finally aggregates all the field specific representations for the document. This framework is visualized in Figure 2.

To formally describe our framework, the document representation learning function Φ_D can be calculated as:

$$\Phi_D(\mathcal{F}_d) = \Lambda_D(\Phi_{F_1}(F_1), \Phi_{F_2}(F_2), \cdots, \Phi_{F_k}(F_k)) \quad (1)$$

where Φ_{F_i} denotes the representation learning function for the field F_i. Note that the representation learning functions differ for different fields, since the fields have their own unique characteristics and need their own specific functions. Λ_D aggregates representations learned for all the fields. Each Φ_{F_i} is also calculated as:

$$\Phi_{F_i}(F_i) = \Lambda_{F_i}(\Phi_{f_i}(f_{i1}), \Phi_{f_i}(f_{i2}), \cdots, \Phi_{f_i}(f_{im_i})) \quad (2)$$

where Φ_{f_i} denotes the representation learning function for each instance of the i^{th} field (e.g., each anchor text). Note that Φ_{f_i} is the same function for all the instances of a given field. The function Λ_{F_i} aggregates the representation of all instances in the i^{th} field.

To summarize, our document representation framework consists of three major components: learning representation for an instance of each field (i.e., Φ_{f_i}), field-level aggregation (i.e., Λ_{F_i}),

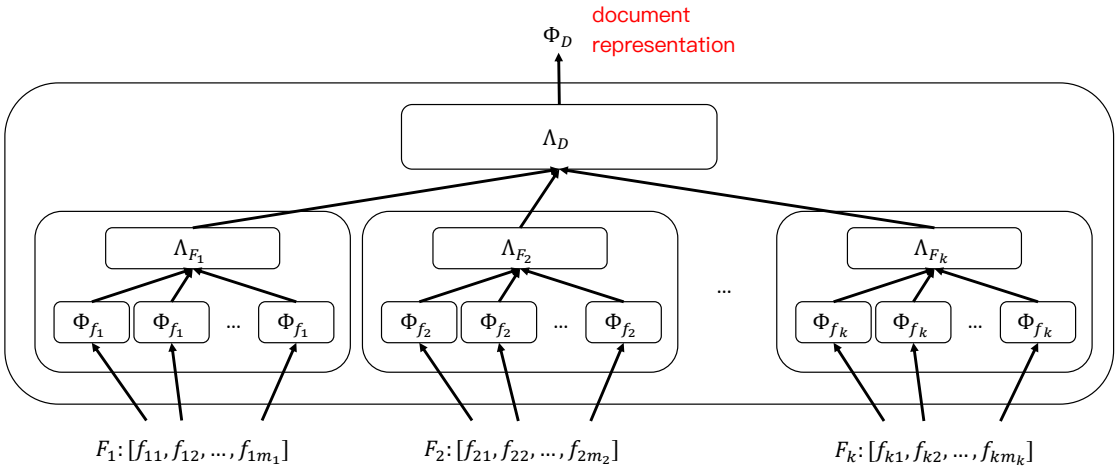

Figure 2: The high-level architecture of our document representation framework. In this architecture, aggregating field-level representations using Λ_D produces the document representation Φ_D. The representation for the i^{th} field is computed by aggregating (Λ_{F_i}) the representations learned for the instances of the field using Φ_{f_i}.

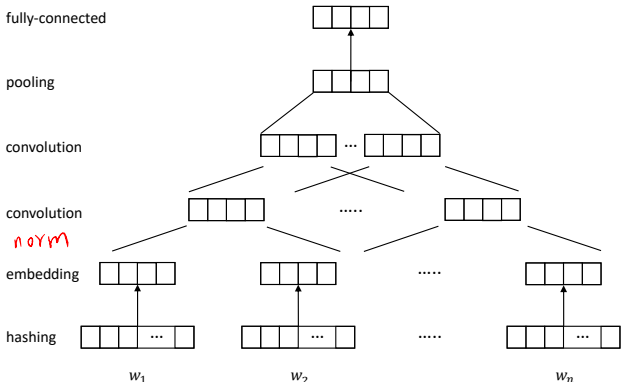

Figure 3: Instance-level representation learning network. This model embeds the character n-gram representation of each word w_i which is followed by two 1-D convolutional layers. The outputs of the second set of convolutional operations are pooled and then fed to a fully-connected layer to compute the final representation for each instance of a field.

and document-level aggregation (i.e., Λ_D). Sections 3.3 and 3.4 describe how we define or learn these functions.

3.3 Instance-Level Representation Learning

In this subsection, we describe our neural architecture for learning representations of individual text instances in a document field. In particular, we explain how the functions Φ_{f_i} are implemented. As pointed out in Section 3.1, each field has its own unique characteristics. One approach would be to use different neural architectures for different fields. However, in the interest of proposing a general framework, we choose an architecture that can be used for all fields, but the exact configurations are controlled by a set of hyper-parameters specified per field. These hyper-parameters are selected for each field individually based on a validation set.

Figure 3 shows the design of the per-instance model architecture. In our architecture, each term is represented using a character n-gram hashing vector, introduced by Huang et al. [8]. These are extremely sparse vectors whose dimensions correspond to all possible character n-grams. Therefore, we represent the input layer of our network using sparse tensors which is memory-efficient and also improves the efficiency of the model. Similar to [8, 23], n was set to 3 in our experiments which causes limited number of term hash collisions. We use the character n-gram representation for the following reasons: (1) it can represent out of vocabulary terms, and (2) the number of all possible tri-grams is much lower than the term-level vocabulary size which significantly reduces the number of parameters needed to be learned by the network. We use a linear embedding layer to map a character n-gram representation to a dense low-dimensional representation by multiplying the sparse input tensor for each term and an embedding matrix $\mathcal{E} \in \mathbb{R}^{N \times l}$ where N is total number of possible n-grams and l denotes the embedding dimensionality. The output of this layer for each word is normalized to prevent over-weighting long words, and represents relevance-based word embedding [32]. Inspired by C-DSSM [23] and the Duet Model [15], this layer is followed by a one-dimensional convolution layer. The aim of this layer is to capture the dependency between terms. We further use an additional convolution layer whose window size is set to be larger for the body field to capture sentence-level representations, and smaller for the short texts fields. We pool the output of the second convolution layer which is followed by a fully-connected layer to compute the final representation for an instance of a given field. The choice of max-pooling and average-pooling is a hyper-parameter in our model. In this network, we use dropout [24] to avoid over-fitting.

3.4 Aggregating Representations

As shown in Figure 2, NRM-F consists of two sets of aggregation components: Λ_{F_i} and Λ_D. Λ_{F_i} aggregates the representations learned for the instances of a specific field. For each multi-instance field F_i, we select a bag of at most M_i instances and use zero padding when less than M_i instances are available. Λ_{F_i} averages the individual representations learned for the instances of the field F_i.

The component Λ_D aggregates the representations learned for different fields. To be able to learn different query representations for each field, Λ_D only concatenates the input vectors to be served in the matching function explained in Section 3.8.

3.5 Field-Level Masking

The number of instances in multi-instance fields, such as anchor text, varies across documents. As shown in Table 1, a significant number of documents may not contain any anchor text or clicked queries. To deal with such cases, as mentioned in Section 3.4, we use zero padding. Although padding is a popular approach and has been previously used in neural ranking models [5, 15, 23], it suffers from a major drawback: by doing padding, the network assumes that a part of the input vector is zero; however padding represents missing values. In the extreme case, assume that there is no available anchor text for a given document; therefore, the input for the anchor text field is all zero. The gradients, however, are not zero (because of the bias parameters). This means that the back-propagation algorithm updates the weights for the sub-network corresponding to the anchor text field; which is not desirable—we do not want to update the weights when the input data is missing. This becomes crucial when there are many missing values in training data, similar to our task.

To tackle this problem, we propose a simple approach, called *field-level masking*. Let $R_i \in \mathbb{R}^{M_i \times D_i}$ denote the representation learned for the i^{th} field (i.e., the output of Λ_{F_i}) where M_i and D_i respectively represent the maximum number of instances (fixed value) and the dimensionality for instance representation. We generate a binary masking matrix $B_i \in \mathbb{B}^{M_i \times D_i}$ whose rows are all zero or all one, showing whether each field instance exists or is missing. In masking, we use $R_i \circ B_i$ (i.e., element-wise multiplication) as the representation for F_i. We multiply the representations for existing field instances by one (means no change) and those for the missing instances by zero. This not only results in zero representation for missing values, but also forces the gradients to become zero. Therefore, the back-propagation algorithm does not update the weights for the sub-networks corresponding to missing values.

The masking matrix is also useful for computing the average in Λ_{F_i} (see Section 3.4). Averaging is a common approach for aggregating different representations, such as average word embedding for query representation [31] and neural ranking models [5]. However, in case of variable length inputs, averaging penalizes short inputs which are padded by zero. To address this issue, we can compute the exact average vector by summing the inputs and dividing them by the summation over the masking matrix. Note that the masking technique should be applied at both training and testing times.

3.6 Field-Level Dropout

As widely known and also demonstrated in our experiments, clicked queries is an effective field for representing documents in the retrieval task [1, 25]. When such a high-accuracy field is available, there is a risk that the network relies on that field, and pays less attention to learning proper representations for the other fields. This can lead to poor performance of the model when the high-accuracy field is absent (low coverage).

Although we use dropout in our neural ranking model (see Section 3.3), it is not sufficient for the task of document representation learning with multiple document fields, in particular when at least a dominant input field exists. To regularize the network in such cases, we propose a simple *field-level dropout* technique—randomly dropping all the units corresponding to a field. In other words, we may randomly drop, say, the clicked queries field or the body field at training time to prevent the neural ranking model from over-dependence on any single field. This approach is back-propagation friendly (all the proofs presented in [24] are applicable to the field-level dropout). Field-level dropout contains k hyper-parameters, where k denotes the total number of fields and each parameter controls the probability of keeping the corresponding field. Note that dropout only happens at the training time and all the units are kept at the validation and test times.

3.7 Query Representation

Since in this paper we focus on the ad-hoc retrieval task, the only available information for the query is the query text. Therefore, to represent the query (i.e., Φ_Q), we use the same network architecture as the one used for each instance of a document field (see Section 3.3). Note that different document fields may match with different aspects of a query. Therefore, the output dimensionality of the query representation network is equal to the sum of the dimensions for all fields' representations. In other words, NRM-F learns different representations of the query for each document field.

3.8 Matching Network

In this subsection, we describe how we compute the retrieval score given the output of query representation and document representation networks (i.e., the function Ψ). To do so, we compute the Hadamard product of the representations; which is the element-wise product of two matrices with the same dimensionality. We then use a fully-connected neural network with a single non-linear hidden layer to compute the final retrieval score. We avoid computing dot product or cosine similarity which would reduce the contribution of each field to a single score, forcing us to combine them linearly which is less effective as demonstrated by Robertson et al. [21] and our results in Section 4.3.

3.9 Training

We use a pairwise setting to train the designed neural ranking model. Let $T = \{(q_1, d_{11}, d_{12}, y_{11}, y_{12}), (q_2, d_{21}, d_{22}, y_{21}, y_{22}), \cdots, (q_n, d_{n1}, d_{n2}, y_{n1}, y_{n2})\}$ be a set of n training instances. Each training instance consists of a query q_i, two documents d_{i1} and d_{i2}, as well as their corresponding labels y_{i1} and y_{i2}. We consider cross entropy loss function to train neural ranking models:

$$\mathcal{L} = -\frac{1}{|T|} \sum_{i=1}^{|T|} \frac{g(y_{i1})}{g(y_{i1}) + g(y_{i2})} \log p_{i1} + \frac{g(y_{i2})}{g(y_{i1}) + g(y_{i2})} \log(1 - p_{i1})$$

where $g(\cdot)$ is a gain function. We use an exponential gain function same as the one used in calculating NDCG. p_{i1} is the estimated probability for d_{i1} being more relevant than d_{i2}. p_{i1} is calculated via softmax on the predicted labels: $p_{i1} = \exp(\hat{y}_{i1})/(\exp(\hat{y}_{i1}) + \exp(\hat{y}_{i2}))$, where $\hat{y}_{i1}$ and $\hat{y}_{i2}$ denote the estimated scores for d_{i1} and d_{i2}, respectively.

4 EXPERIMENTS

4.1 Data

To evaluate our models, we randomly sampled ~140k queries from the Bing's search logs for the English United States market from a one-year period. For each query, the documents returned by the Bing's production ranker in addition to those retrieved by a

Table 1: Statistics and characteristics of the document fields used in our experiments.

Fields	Type	Coverage	Specific Feature
Title	Single Instance	100%	Short text.
URL	Single Instance	100%	Short text, but not in a natural language.
Body	Single Instance	100%	Long text.
Anchor texts	Multiple Instance	61%	Short texts with relatively low coverage.
Clicked queries	Multiple Instance	73%	Short texts with relatively low coverage. A high-accuracy field.

diverse set of experiments were labelled by human judges on a five-point scale: perfect, excellent, good, fair, and bad. In total, the data consists of ~3.8 million query-document pairs which was randomly partitioned into three sets—80% for training, 10% for validation, and 10% for testing—such that no distinct query appears in more than one set. Similar to [8, 15, 17], we evaluate all models under the *telescoping* setting by re-ranking the candidate documents for each query. Since our neural ranking model is a pairwise learning to rank model, for each query we generate all possible $< q, d_1, d_2 >$ triples such that the relevance label for d_1 and d_2 are different with respect to q. To avoid biasing towards the queries with many documents, at most 50 triples per query were sampled for training based on a uniform distribution over all possible label pairs.

The contents of web pages were retrieved from the Bing's web index and were parsed using a proprietary HTML parser. We made sure that all the documents in our data contain title and body. All texts were normalized by lower-casing and removing non-alphanumerical characters. The URLs were split using a simple proprietary approach. We set the maximum length of 20, 10, 1000, 10, and 10 for title, URL, body, anchor text, and clicked query, respectively. We used at most 5 anchor texts and at most 5 clicked queries per document.[2] They were selected based on a simple count-based functions; means that the most common anchor texts and clicked queries for each document were selected. The statistics of our data for each field is reported in Table 1.

4.2 Experimental Setup

All the models were implemented using TensorFlow[3]. We used Adam optimizer [11] to train our models. The learning rate was selected from $[1e-3, 5e-4, 1e-4, 5e-5, 1e-5]$. We set the batch size to 64 and tuned the hyper-parameters based on the loss values obtained on the validation set. We selected the layer sizes from $\{100, 300, 500\}$ and the convolution window sizes from $\{1, 3, 10, 20, 50\}$ for long texts (i.e., body) and from $\{1, 3, 5, 10\}$ for short texts (i.e., the other fields). The convolution strides were selected from $\{1, \lfloor ws/4 \rfloor, \lfloor ws/2 \rfloor, ws\}$ where ws denotes the convolution window size. The keep probability parameters for both conventional and field-level dropouts were selected from $\{0.5, 0.8, 1.0\}$.

As explained in Section 3.3, the input layer of the networks uses tri-gram hashing with $\sim 50k$ dimensions, i.e, all possible character tri-grams with alphanumerical characters plus a dummy character for the start and the end of each word. The tri-gram embedding dimensionality (i.e., the first layer) was set to 300. This embedding matrix is shared among all fields. Following [8, 15, 23], we used tanh as the activation function for all hidden layers.

[2]The maximum number of instances per field can be set to a much larger value. Since the network parameters for instances of each field are shared and the inputs are represented as sparse tensors, increasing the maximum number of instances would have a minor memory effect.

[3]http://tensorflow.org/

Table 2: Performance of the proposed framework with different fields. The superscript + shows significant improvements for the models with two fields compared to the ones with each of the fields, individually. The superscript * denotes significant improvements over all the other models.

Field(s)	NDCG@1	NDCG@10
Title	0.4226	0.5883
URL	0.4366	0.5865
Body	0.4115	0.5850
Anchor texts	0.4386	0.5933
Clicked queries	0.4661	0.6116
Title + URL	0.4425^{+}	0.6065^{+}
Title + Body	0.4316^{+}	0.6098^{+}
Title + Anchor texts	0.4507^{+}	0.6062^{+}
Title + Clicked queries	0.4680	0.6180^{+}
All	$\mathbf{0.4906}^{*}$	$\mathbf{0.6380}^{*}$

We use NDCG at two different ranking levels (NDCG@1 and NDCG@10) to evaluate the models. The significance differences between models are determined using the paired t-test at a 95% confidence level ($p\_value < 0.05$).

4.3 Experimental Results

In this subsection, we empirically address the hypotheses mentioned in Section 1.

H1: The ad-hoc retrieval performance of NRM-F improves as we incorporate multiple document fields. In this set of experiments, we address our first hypothesis (H1) by evaluating our model with each single field individually, with field pairs with title, and finally with all the fields together. The results are reported in Table 2. Although title, URL, and body have much higher coverage compared to anchor texts and clicked queries (see Table 1), the performances achieved by anchor texts and clicked queries are superior to the other fields.[4] Incorporating clicked queries demonstrates the highest performance. Pairing Title with any of the other field "X" leads to a better performance compared to Title and "X", individually. These improvements are statistically significant, except for NDCG@1 in Title+Clicked queries. The reason is that clicked queries are very effective for web search, especially for the first retrieved document. Adding title to clicked queries, however, significantly improves the search quality for the top 10 documents. The NRM-F model with all fields achieves the highest performance with

[4]We randomly shuffled the documents with equal retrieval scores for a query. This process was repeated for 10 times and the average performance is reported.

Table 3: Comparison of the proposed model with baselines for a single field (Title or Body). The superscripts denote significant improvements over the models specified by the ID column.

ID	Model	Title		Body	
		NDCG@1	NDCG@10	NDCG@1	NDCG@10
1	BM25	0.4039	0.5752	0.3957	0.5693
2	LTR	0.4122	0.5861	0.3996	0.5792
3	DSSM	0.4112	0.5858	0.3961	0.5713
4	C-DSSM	0.4148	0.5874	0.3957	0.5695
5	Duet (distributed)	0.4164	0.5877	0.4066	0.5788
6	NRM-F - Single Field	**0.4226**^{12345}	**0.5883**^{123}	**0.4115**^{12345}	**0.5850**^{12345}

statistically significant margins. This suggests that the proposed framework is able to learn a more accurate document representation for the ad-hoc retrieval task by considering multiple document fields; thus the hypothesis H1 is validated.

H2: NRM-F performs better than competitive baselines, such as term matching and learning to rank. To demonstrate that the proposed instance-level representation model performs reasonably well for both short and long texts, we first evaluate our models against a set of baselines using a single field, title only and body only. We consider the following baselines: BM25 [22], a state-of-the-art learning to rank model with hand-crafted features (LTR), DSSM [8], C-DSSM [23], and the distributed part[5] of the duet model proposed by Mitra et al. [15]. The LTR baseline uses an internal advanced implementation of the LambdaMART algorithm [4] that has been used in the production. We used the features that have been typically extracted from query and document texts. Indeed, from those listed in [20], we used all the features that can be extracted from query and title/body.

To have a fair comparison, we trained all the models using the same training data and pairwise setting.[6] The hyper-parameters in all the models, including the baselines, were optimized for Title and Body, separately. Due to the memory constraints, the C-DSSM and Duet cannot use $\sim 50k$ tri-grams for the word hashing phase (only for Body). Therefore, as suggested in [15], we use top $2k$ popular n-grams for these models. Note that since our model use sparse tensors for word hashing, it is memory-efficient and does not have the same issue.[7]

The results for Title as an example of short text and Body as an example of long text are reported in Table 3. According to this table, the proposed method outperforms all the baselines for both Title and Body. The improvements are statistically significant in nearly all cases. This demonstrates the potential of our model to be used for both short and long texts. The improvements are higher for Body, which makes our model even more suitable for long text. This experiment suggests that our instance-level representation model performs reasonably well.

[5]To have a fair comparison, we only consider the distributed part of the model. Note that all the listed neural models, including NRM-F, can be further enriched by using lexical matching, similar to the local part of the duet model.

[6]The original DSSM and C-DSSM models use binary labels (click data) and random negative sampling for training; however, as suggested by Mitra et al. [15] using explicit judgments leads to a better performance compared to random negative sampling

[7]C-DSSM and Duet perform convolution on top of word hashing layer; thus, the word hashing phase cannot be implemented using sparse tensors (at least not supported by deep learning libraries, such as TensorFlow and CNTK).

Table 4: Performance of the proposed framework with all fields compared to baselines. The superscript * denotes significant improvements over all the other models.

Model	NDCG@1	NDCG@10
BM25-Field Concatenation	0.4281	0.5953
BM25F	0.4431	0.6020
LTR	0.4888	0.6341
NRM-Field Concatenation	0.4582	0.6110
NRM-Score Aggregation-Ind. Training	0.4729	0.6229
NRM-Score Aggregation-Co-training	0.4743	0.6279
NRM-F -Single Query Representation	0.4846	0.6345
NRM-F	**0.4906***	**0.6380***

To evaluate our model with multiple instances, we consider the following baselines: (1) BM25 by concatenating all the fields, (2) BM25F [21] which has been widely used for ad-hoc retrieval with multiple document fields, (3) a learning to rank (LTR) model with hand-crafted features extracted from all the fields, and (4) our neural ranking model with concatenation of all fields as a single input text (i.e., NRM - Field Concatenation). Similar to the last experiments, for LTR we consider all the typical features that can be extracted from text inputs (among those listed in [20] for the LETOR dataset). The features were extracted for all the fields. The learning algorithm for LTR is the same as the one used in the previous experiment. All models were trained on the same training set, and their hyper-parameters were tuned on the same validation set. As shown in Table 4, NRM-F significantly outperforms all the baselines. This suggests that NRM-F not only eliminates the hand-crafted feature engineering for ad-hoc retrieval, but also learns an accurate document representation that leads to higher retrieval performance. The results also validate our second hypothesis.

H3: Learning a multiple-field document representation is superior to scoring based on individual field representations and summing. A simple approach for coping with multiple document fields is to calculate the matching score for the query and each of the document fields and then aggregate the scores. We tried two score aggregation methods, one learns a neural ranking model for each document field individually and then linearly interpolates their scores. Although the other one also interpolates the scores obtained by different fields, the neural networks for different fields are co-trained together. The results in Table 4 show that co-training leads to a better performance compared to isolated training of the

Table 5: Investigating the effectiveness of field-level masking and dropout. The superscripts denote significant improvements over the models specified by the ID column.

ID	Model	All fields		All fields except clicked queries	
		NDCG@1	NDCG@10	NDCG@1	NDCG@10
1	NRM-F (no masking, no dropout)	0.4818	0.6327	0.4577	0.6152
2	NRM-F with masking	0.4856[1]	0.6353[1]	0.4602[1]	0.6174[1]
3	NRM-F with masking & dropout	**0.4906**[12]	**0.6380**[12]	**0.4613**[1]	**0.6181**[1]

Table 6: Performance analysis based on query length, dividing the test queries into three evenly-sized groups.

Model	Short queries		Medium-length queries		Long queries	
	NDCG@1	NDCG@10	NDCG@1	NDCG@10	NDCG@1	NDCG@10
LTR	0.5040	0.6470	0.4753	0.6332	**0.4799**	0.6162
NRM-F	**0.5132**	**0.6584**	**0.4846**	**0.6355**	0.4723	**0.6186**

model for different fields, which is expected. The results also suggest that NRM-F performs better than neural ranking models with score aggregation. The improvements are statistically significant. Therefore, this experiment validates our third hypothesis.

H4: Learning per-field query representations performs better than learning a single query representation. As mentioned in Section 3.7, we believe that different aspects of the query can match different fields, and thus different query representations are needed for different fields. Our empirical results in Table 4 also validate this hypothesis by showing that NRM-F provides a superior performance in comparison with exactly the same neural ranking model, but with single query representation for different fields.

H5: The additional techniques of field-level masking and field-level dropout yield additional performance improvements. To study this hypothesis, we report the results for the following models: (1) our neural ranking model with no field-level masking and dropout, (2) our model with only field-level masking, and eventually (3) our model with both field-level masking and dropout. Note that all the models use conventional dropout [11]. Table 5 reports the results for all fields and for all fields except clicked queries. According to this table, field-level masking is useful to cope with multi-instance fields and significantly improves the performance. The model with both field-level masking and dropout achieves the highest performance; however, the field-level dropout technique is significantly helpful, when at least one of the fields is dominant (i.e., the high-accuracy fields like clicked queries).

4.4 Additional Analysis

Learning curve. It has always been important to know how much data is needed to train the model. We plot the learning curve for our NRM-F model with all fields in Figure 4. The performance is reported in terms of NDCG@10 on the test set. According to this figure, we need approximately two million training instances to have a relatively stable performance.

Analysis by query length. In this analysis, we uniformly split the test queries into three buckets based on their query length. Therefore, the number of queries in the buckets are approximately equal. The first bucket includes the shortest and the last one includes the longest queries. The results for NRM-F and the LTR baseline with all fields (the one used in Table 4) are reported in Table 6. According to this table, our improvements over the LTR baseline generally decrease by increasing the query length. In other words, NRM-F performs relatively better for shorter queries. The reason is that long queries are often rare and thus it is likely that the models based on representation learning work much better for shorter queries. On the other hand, the experiment is in a telescoping setting with anchor texts and clicked queries. Therefore, the additional terms and synonyms provided by, let say, clicked queries empower the LTR method that uses term matching features. In addition, for long queries in a telescoping setting, ignoring a query term is relatively likely to do not harm the results.

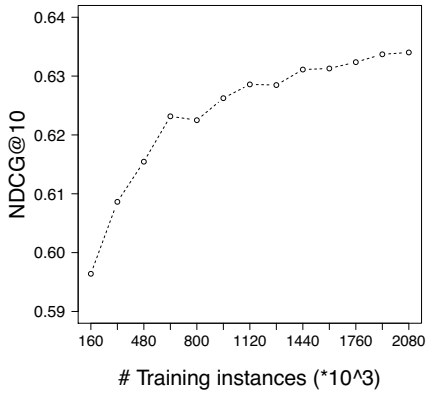

Figure 4: Learning curve demonstrating the performance of NRM-F in terms of NDCG@10 with respect to the size of training set.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed NRM-F, a *general* framework for the task of multiple-field document representation learning for ad-hoc retrieval. NRM-F can consume both short text and long text fields. It can also handle the multi-instance fields, such as anchor text. Since fields vary in coverage, we introduced a field-level masking method to handle missing field instances. We also proposed a field-level dropout technique to prevent the model from over-dependence on high-accuracy fields, such as clicked queries. We performed extensive experiments using a large set of query-document pairs labelled by human judges. Our experiments suggested that incorporating multiple document fields significantly improves the performance

of neural ranking models by a large margin. Our model also outperforms state-of-the-art traditional term matching and learning to rank models, significantly. We showed that multiple query representations are needed for different document fields, based on the intuition that different aspects of a query may be matched against different fields of a document. Our empirical results also demonstrated that learning a multiple-field document representation is superior to aggregating retrieval scores from matching the query with different fields. We further showed that field-level masking and dropout are useful for handling fields with variable number of text instances and avoiding over-dependence on high-accuracy fields, respectively.

This work smooths the path towards pursuing several research directions in the future. For instance, many retrieval tasks based on semi-structured documents, such as academic search, XML retrieval, product search, expert finding, etc. can benefit from the NRM-F framework for improving the retrieval performance. Another possible future direction is to extend the NRM-F framework by considering lexical matching of query and document fields. We can also explore incorporating query independent features, such as, PageRank score, into our framework. Finally, our findings may be applicable to non-ranking tasks, including document classification, spam detection, and document filtering.

Acknowledgements. This work was supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving Web Search Ranking by Incorporating User Behavior Information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*. ACM, Seattle, Washington, USA, 19–26.

[2] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A Neural Click Model for Web Search. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. IW3C2, Montreal, Quebec, Canada, 531–541.

[3] Christopher J. Burges, Robert Ragno, and Quoc V. Le. 2006. Learning to Rank with Nonsmooth Cost Functions. In *Advances in Neural Information Processing Systems 19*. MIT Press, Vancouver, B.C., Canada, 193–200.

[4] Christopher J. C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report. Microsoft Research.

[5] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, Shinjuku, Tokyo, Japan, 65–74.

[6] Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai. 2006. *Advances in XML Information Retrieval and Evaluation: 4th International Workshop of the Initiative for the Evaluation of XML Retrieval*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[7] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, Indianapolis, Indiana, USA, 55–64.

[8] 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, San Francisco, California, USA, 2333–2338.

[9] Jinyoung Kim, Xiaobing Xue, and W. Bruce Croft. 2009. A Probabilistic Retrieval Model for Semistructured Data. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval (ECIR '09)*. Springer-Verlag, Toulouse, France, 228–239.

[10] Jin Young Kim and W. Bruce Croft. 2012. A Field Relevance Model for Structured Document Retrieval. In *Proceedings of the 34th European Conference on Advances in Information Retrieval (ECIR'12)*. Springer-Verlag, Barcelona, Spain, 97–108.

[11] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *Proceedings of the third International Conference on Learning Representations (ICLR '14)*. Banff, Canada.

[12] Mounia Lalmas and Anastasios Tombros. 2007. Evaluating XML Retrieval Effectiveness at INEX. *SIGIR Forum* 41, 1 (June 2007), 40–57.

[13] Zhengdong Lu and Hang Li. 2013. A Deep Architecture for Matching Short Texts. In *Advances in Neural Information Processing Systems 26 (NIPS '13)*. Lake Tahoe, CA, USA, 1367–1375.

[14] Bhaskar Mitra and Nick Craswell. 2018. An Introduction to Neural Information Retrieval. *Foundations and Trends® in Information Retrieval (to appear)* (2018).

[15] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. IW3C2, Perth, Australia, 1291–1299.

[16] Sung Hyon Myaeng, Don-Hyun Jang, Mun-Seok Kim, and Zong-Cheol Zhoo. 1998. A Flexible Model for Retrieval of SGML Documents. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*. ACM, Melbourne, Australia, 138–145.

[17] Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving Document Ranking with Dual Word Embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web (WWW '16 Companion)*. IW3C2, Montreal, Quebec, Canada, 83–84.

[18] Paul Ogilvie and Jamie Callan. 2003. Combining Document Representations for Known-item Search. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval (SIGIR '03)*. ACM, Toronto, Canada, 143–150.

[19] Benjamin Piwowarski and Patrick Gallinari. 2003. A machine learning model for information retrieval with structured documents. *Machine Learning and Data Mining in Pattern Recognition* (2003), 425–438.

[20] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. *Inf. Retr.* 13, 4 (Aug. 2010), 346–374.

[21] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple BM25 Extension to Multiple Weighted Fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management (CIKM '04)*. ACM, Washington, D.C., USA, 42–49.

[22] S. E. Robertson, E. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1994. Okapi at TREC-3. In *Proceedings of the third Text Retrieval Conference (TREC '94)*.

[23] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14)*. ACM, Shanghai, China, 101–110.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958.

[25] Krysta M. Svore and Christopher J.C. Burges. 2009. A Machine Learning Approach for Improved BM25 Retrieval. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. ACM, Hong Kong, China, 1811–1814.

[26] Ross Wilkinson. 1994. Effective Retrieval of Structured Documents. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)*. Springer-Verlag New York, Inc., Dublin, Ireland, 311–317.

[27] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, Shinjuku, Tokyo, Japan, 55–64.

[28] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Yong Yu, Wei-Ying Ma, WenSi Xi, and WeiGuo Fan. 2004. Optimizing Web Search Using Web Click-through Data. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management (CIKM '04)*. ACM, Washington, D.C., USA, 118–126.

[29] Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. 2016. aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, Indianapolis, Indiana, USA, 287–296.

[30] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. 2017. Situational Context for Ranking in Personal Search. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. IW3C2, Perth, Australia, 1531–1540.

[31] Hamed Zamani and W. Bruce Croft. 2016. Estimating Embedding Vectors for Queries. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval (ICTIR '16)*. ACM, Newark, Delaware, USA, 123–132.

[32] Hamed Zamani and W. Bruce Croft. 2017. Relevance-based Word Embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, Shinjuku, Tokyo, Japan, 505–514.