Learning to Match using Local and Distributed Representations of Text for Web Search

Bhaskar Mitra*[1,2], Fernando Diaz[1], and Nick Craswell[1]

[1]Microsoft, {bmitra, fdiaz, nickcr}@microsoft.com
[2]University College London, {bhaskar.mitra.15}@ucl.ac.uk

ABSTRACT

Models such as latent semantic analysis and those based on neural embeddings learn *distributed* representations of text, and match the query against the document in the latent semantic space. In traditional information retrieval models, on the other hand, terms have discrete or *local* representations, and the relevance of a document is determined by the exact matches of query terms in the body text. We hypothesize that matching with distributed representations complements matching with traditional local representations, and that a combination of the two is favourable. We propose a novel document ranking model composed of two separate deep neural networks, one that matches the query and the document using a local representation, and another that matches the query and the document using learned distributed representations. The two networks are jointly trained as part of a single neural network. We show that this combination or 'duet' performs significantly better than either neural network individually on a Web page ranking task, and significantly outperforms traditional baselines and other recently proposed models based on neural networks.

Keywords

Information retrieval; neural networks; document ranking

1. INTRODUCTION

Neural text embedding models have recently gained significant popularity for both natural language processing (NLP) and information retrieval (IR) tasks. In IR, a significant number of these works have focused on word embeddings [6, 8, 10, 11, 27, 28, 34, 41] and modelling short-text similarities [15, 16, 29, 35–37]. In traditional Web search, the query consists of only few terms but the body text of the documents may typically have tens or hundreds of sentences. In the absence of click information, such as for newly-published or infrequently-visited documents, the body text can be a useful signal to determine the relevance of the document for the query. Therefore, extending existing neural text representation learning approaches to long body text for document ranking is an important challenge in IR. However, as was noted during a recent workshop [4], despite the recent surge in interests towards applying deep neural network (DNN) models for retrieval, their success on ad-hoc retrieval tasks has been rather limited. Some recent papers [30, 34] report worse performance of neural embedding models when compared to traditional term-based approaches, such as BM25 [33].

Traditional IR approaches consider terms as discrete entities. The relevance of the document to the query is estimated based on, amongst other factors, the number of matches of query terms in the document, the parts of the document in which the matches occur, and the proximity between the matches. In contrast, latent semantic analysis (LSA) [5], probabilistic latent semantic analysis (PLSA) [14] and latent Dirichlet allocation (LDA) [2, 39] learn low-dimensional vector representations of terms, and match the query against the document in the latent semantic space. Retrieval models can therefore be classified based on what representations of text they employ at the point of matching the query against the document. At the point of match, if each term is represented by a unique identifiers (*local* representation [13]) then the query-document relevance is a function of the pattern of occurrences of the exact query terms in the document. However, if the query and the document text is first

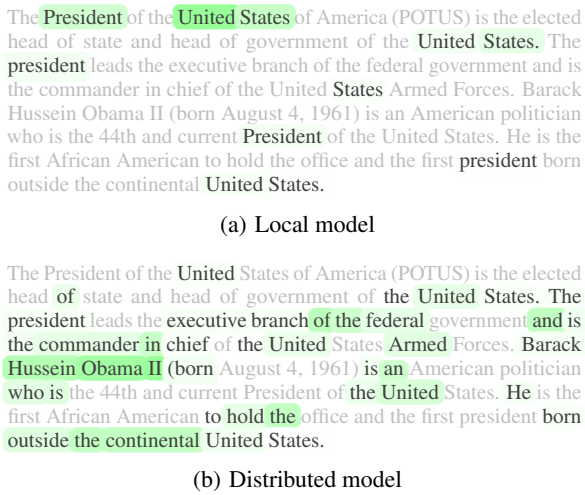


Figure 1: Visualizing the drop in the local and the distributed model's retrieval score by individually removing each of the passage terms for the query "united states president". Darker green signifies a bigger drop. The local model uses only exact term matches. The distributed model uses matches based on a learned representation.

*The author is a part-time PhD student at UCL.

©2017 International World Wide Web Conference Committee (IW3C2), published under Creative Commons CC BY 4.0 License.
WWW 2017, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4913-0/17/04.
http://dx.doi.org/10.1145/3038912.3052579

projected into a continuous latent space, then it is their distributed representations that are compared. Along these lines, Guo et al. [12] classify recent DNN models for short-text matching as either *interaction*-focused [15, 22, 31] or *representation*-focused [15, 16, 35–37]. They claim that IR tasks are different from NLP tasks, and that it is more important to focus on exact matching for the former and on learning text embeddings for the latter. Mitra et al. [27], on the other hand, claim that models that compare the query and the document in the latent semantic space capture a different sense of relevance than models that focus on exact term matches, and therefore the combination of the two is more favourable. Our work is motivated by the latter intuition that it is important to match the query and the document using both local and distributed representations of text. We propose a novel ranking model comprised of two separate DNNs that model query-document relevance using local and distributed representations, respectively. The two DNNs, referred to henceforth as the *local model* and the *distributed model*, are jointly trained as part of a single neural network, that we name as a *duet* architecture because the two networks co-operate to achieve a common goal. Figure 1 demonstrates how each subnetwork models the same document given a fixed query. While the local model captures properties like exact match position and proximity, the distributed model detects synonyms (e.g. 'Obama'), related terms (e.g. 'federal'), and even well-formedness of content (e.g. 'the', 'of').[1]

In this paper, we show that the duet of the two DNNs not only outperforms the individual local and distributed models, but also demonstrates large improvements over traditional baselines and other recently proposed models based on DNNs on the document ranking task. Unlike other recent work [30, 34], our model significantly outperforms classic IR approaches by using a DNN to learn text representation.

Deep neural network models are known to benefit from large training data, achieving state-of-the-art performance in areas where large scale training corpora are available [19, 20]. Some of the lack of positive results from neural models in ad-hoc retrieval is likely due to the scarce public availability of large quantity of training data necessary to learn effective representations of text. In Section 6, we will present some analysis on the effect of training data on the performance of these DNN models. In particular, we found that–unsurprisingly–the performance of the distributed model improves drastically in the presence of more data. Unlike some previous work [16, 36, 37] that train on clickthrough data with randomly sampled documents as negative examples, we train our model on human-judged labels. Our candidate set for every query consists of documents that were retrieved by the commercial search engine Bing, and then labelled by crowdsourced judges. We found that training with the documents that were rated non-relevant by the human judges as the negative examples is more effective than randomly sampling negative examples from the corpus. To summarize, the key contributions of this work are:

1. We propose a novel duet architecture for a model that jointly learns two deep neural networks focused on matching using local and distributed representations of text, respectively.
2. We demonstrate that this architecture out-performs state-of-the-art neural and traditional non-neural baselines.
3. We demonstrate that training with documents judged as non-relevant as the negative examples is more effective than randomly sampling them from the corpus.

[1]While surprising, this last property is important for detecting quality web content [42].

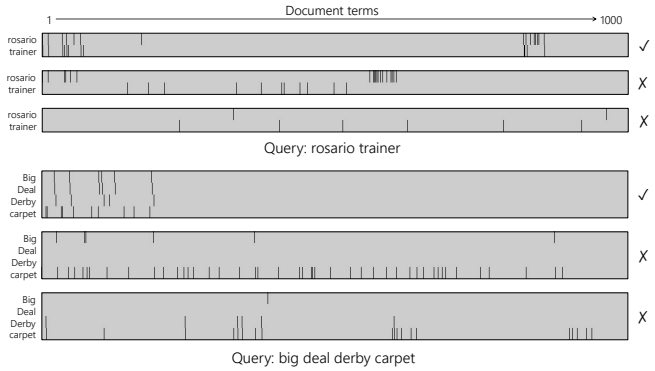


Figure 2: Visualizing patterns of query term matches in documents. Query terms are laid out along the vertical axis, and the document terms along the horizontal axis. The short vertical lines correspond to exact matches between pairs of query and document terms. For both queries, the first document was rated relevant by a human judge and the following two as non-relevant. The query term matches in the relevant documents are observed to be more clustered, and more localized near the beginning of the document.

2. DESIDERATA OF DOCUMENT RANKING

Before describing our ranking model, we first present three properties found across most effective retrieval systems. We will then operationalize these in our architecture in Section 3.

First, *exact term matches* between the query and the document are fundamental to all information retrieval models [7]. Traditional IR models, such as BM25 [33], are based on counts of exact matches of the query terms in the document text. They can be employed with minimal (or no) need for training data, sometimes directly on new tasks or corpora. Exact matching can be particularly important when the query terms are new or rare. For example, if new documents appear on the Web with the television model number 'SC32MN17' then BM25 can immediately retrieve these pages containing precisely that model number without adjusting any parameters of the ranking model. A good ranking model needs to take advantage of exact matches to perform reliably on queries containing terms with rare or no occurrences in the data the model is trained on.

Second, *match positions* of the query terms in the document not only reflect where potentially the relevant parts of the document are localized (e.g. title, first paragraph, closing paragraph) but also how clustered the individual query term matches are with each other. Figure 2 shows the position of matches on two different queries and a sample of relevant and non-relevant documents. In the first query, we see that the query term matches in the relevant document are much more clustered than in the non-relevant documents. We observe this behaviour also in the second query but in addition notice that the clustered matches are localized near the beginning of the relevant document. Match proximity serves as a foundation for effective methods such as sequential dependence models [23].

Finally, *inexact term matches* between the query and the document refer to techniques for addressing the vocabulary mismatch problem. The main disadvantage of term matching is that related terms are ignored, so when ranking for the query 'Australia' then only the term frequency of 'Australia' is considered, even though counting terms

like 'Sydney' and 'koala' can be good positive evidence. Mitra et al. [27] anecdotally demonstrate that a distributed representation based retrieval model that considers *all* document terms can better distinguish between a passage that is truly relevant to the query "Cambridge" from a passage on a different topic (e.g., giraffes) with artificially injected occurrences of the term "Cambridge". They claim that any IR model that considers the distribution of non-matching terms is likely to benefit from this additional evidence of relevance, and be able to tell "Cambridge" apart from "an African even-toed ungulate mammal".[2]

In practice, the most effective IR methods leverage combinations of these techniques. Dependence models combine exact matching with proximity [23]. LDA-based document models combine exact matching with inexact matching [39]. Query hypergraphs capture all three [1]. Our method also combines these techniques but, unlike prior work, jointly learns all the free parameters of the different components within a single deep neural network architecture.

3. THE DUET ARCHITECTURE

Figure 3 provides a detailed schematic view of the duet architecture. The distributed model projects the query and the document text into an embedding space before matching, while the local model operates over an interaction matrix comparing every query term to every document term. The final score under the duet setup is the sum of scores from the local and the distributed networks,

$$f(\mathbf{Q}, \mathbf{D}) = f_\ell(\mathbf{Q}, \mathbf{D}) + f_d(\mathbf{Q}, \mathbf{D}) \quad (1)$$

where both the query and the document are considered as ordered list of terms, $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_{n_\mathbf{q}}]$ and $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_{n_\mathbf{d}}]$. Each query term $\mathbf{q}$ and document term $\mathbf{d}$ is a $m \times 1$ vector where m is the input representation of the text (e.g. the number of terms in the vocabulary for the local model).

We fix the length of the inputs across all the queries and the documents such that we consider only the first 10 terms in the query and the first 1000 terms in the document. If either the query or the document is shorter than these target dimensions, then the input vectors are padded with zeros. The truncation of the document body text to the first 1000 terms is performed only for our model and its variants, but not for the baseline models. For all the neural and the non-neural baseline models we consider the full body text.

3.1 Local Model

The local model estimates document relevance based on patterns of exact matches of query terms in the document. To this end, each term is represented by its one-hot encoding in a m_ℓ-dimensional space, where m_ℓ is the size of the vocabulary. The model then generates the $n_\mathbf{d} \times n_\mathbf{q}$ binary matrix $\mathbf{X} = \mathbf{D}^\mathsf{T}\mathbf{Q}$, capturing every exact match (and position) of query terms in the document. This interaction matrix is similar to the visual representation of term matches in Figure 2, and therefore captures both the exact term matches and the match positions. It is also similar to the indicator matching matrix proposed previously by Pang et al. [31]. While the interaction matrix $\mathbf{X}$ perfectly captures every query term match in the document, it does not retain any information about the actual terms themselves. Therefore, the local model cannot learn term-specific properties from the training corpus, nor model interactions between dissimilar terms.

The interaction matrix $\mathbf{X}$ is first passed through a convolutional layer with c filters, a kernel size of $n_\mathbf{d} \times 1$, and a stride of 1. The

[2]A Python implementation of this visualization is available at https://github.com/bmitra-msft/Demos/blob/master/notebooks/DESM.ipynb

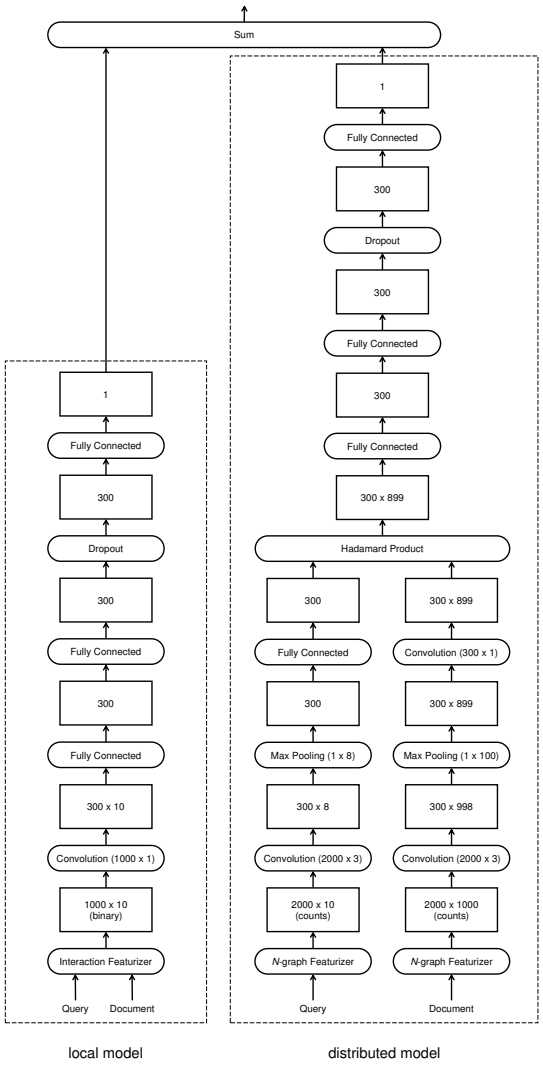


Figure 3: The duet architecture is composed of the local model (left) and the distributed model (right). The local sub-network takes an interaction matrix of query and document terms as input, whereas the distributed sub-network learns embeddings of the query and the document text before matching. The parameters of both models are optimized jointly during training.

output $\mathbf{Z}_i$ corresponding to the i^{th} convolutional window over $\mathbf{X}$ is a function of the match between the $\mathbf{q}_i$ term against all the terms in the document,

$$\mathbf{Z}_i = \tanh\left(\mathbf{X}_i^\mathsf{T}\mathbf{W}\right) \quad (2)$$

where $\mathbf{X}_i$ is the row i of $\mathbf{X}$, $\tanh$ is performed elementwise, and the $n_\mathbf{d} \times c$ matrix $\mathbf{W}$ contains the learnable parameters of the convolutional layer. The output $\mathbf{Z}$ of the convolutional layer is a matrix of dimension $c \times n_\mathbf{q}$. We use a filter size (c) of 300 for all the evaluations reported in this paper. The output of the convolutional layer is then passed through two fully-connected layers, a dropout layer, and a final fully-connected layer that produces a single real-valued output. All the nodes in the local model uses the hyperbolic tangent function for non-linearity.

3.2 Distributed Model

The distributed model learns dense lower-dimensional vector representations of the query and the document text, and then computes the positional similarity between them in the learnt embedding space. Instead of one-hot encoding of terms, as in the local model, we use a character n-graph based representation of each term in the query and document. Our n-graph based input encoding is motivated by the trigraph encoding proposed by Huang et al. [16], but unlike their approach we don't limit our input representation to n-graphs of a fixed length. For each term, we count all the n-graphs present for $1 \leq n \leq \text{G}$. We then use this n-graph frequency vector of length m_d to represent the term.

Instead of directly computing the interaction between the $m_d \times n_\mathbf{q}$ matrix $\mathbf{Q}$ and the $m_d \times n_\mathbf{d}$ matrix $\mathbf{D}$, we first learn a series of nonlinear transformations to the character-based input. For both the query and the document, the first step is convolution. The $m_d \times 3$ convolution window has filter size of 300. It projects 3 consecutive terms to a 300-dimensional vector, then takes a stride by 1 position, and projects the next 3 terms, and so on. For the query, the convolution step generates a tensor of dimensions 300×8. For the document, it generates one of dimensions 300×998.

Following this, we conduct a max-pooling step. For the query the pooling kernel dimensions are 1×8. For the document, it is 1×100. Thus, we get one 300×1 matrix $\tilde{\mathbf{Q}}$ for the query and a 300×899 matrix $\tilde{\mathbf{D}}$ for the document. The document matrix $\tilde{\mathbf{D}}$ can be interpreted as 899 separate embeddings, each corresponding to different equal-sized spans of text within the document. Our choice of a window-based max-pooling strategy, instead of global max-pooling as employed by CDSSM [37], is motivated by the fact that the window-based approach allows the model to distinguish between matches in different parts of the document. As posited in Section 2, a model that is aware of match positions may be more suitable when dealing with long documents, especially those containing mixture of many different topics.

The output of the max-pooling layer for the query is then passed through a fully-connected layer. For the document, the 300×899 dimensional matrix output is operated on by another convolutional layer with filter size of 300, kernel dimensions of 300×1, and a stride of 1. The combination of these convolutional and max-pooling layers enable the distributed model to learn suitable representations of text for effective inexact matching.

To perform the matching, we conduct the element-wise or Hadamard product between the embedded document matrix and the extended or broadcasted query embedding,

$$\tilde{\mathbf{X}} = (\underbrace{\tilde{\mathbf{Q}} \ldots \tilde{\mathbf{Q}}}_{\text{899 times}}) \circ \tilde{\mathbf{D}} \tag{3}$$

After this, we pass the matrix through fully connected layers, and a dropout layer until we arrive at a single score. Like the local model, we use hyperbolic tangent function here for non-linearity.

3.3 Optimization

Each training sample consists of a query $\mathbf{Q}$, a relevant document $\mathbf{D}^*$ and a set of non-relevant documents $\mathcal{N} = \{\mathbf{D}_0, \ldots, \mathbf{D}_\text{N}\}$. We use a softmax function to compute the posterior probability of the positive document given a query based on the score.

$$p(\mathbf{D}^*|\mathbf{Q}) = \frac{\exp(f(\mathbf{Q}, \mathbf{D}^*))}{\sum_{\mathbf{D} \in \mathcal{N}} \exp(f(\mathbf{Q}, \mathbf{D}))} \tag{4}$$

and we maximize the log likelihood $\log p(\mathbf{D}^*|\mathbf{Q})$ using stochastic gradient descent.

Table 1: Statistics of the three test sets randomly sampled from Bing's search logs. The candidate documents are generated by querying Bing and then rated using human judges.

	queries	documents	$\frac{\text{documents}}{\text{query}}$
training	199,753	998,765	5
weighted test	7,741	171,302	24.9
unweighted test	6,808	71,722	10.6

4. MATERIALS AND METHODS

We conducted three experiments to test: (1) the effectiveness of our duet model compared to the local and distributed models separately, and (2) the effectiveness of our duet model compared to existing baselines for content-based web ranking, (3) the effectiveness of training with judged negative documents compared to random negative documents. In this section, we detail our experiment setup and baseline implementations.

4.1 Data

The training dataset consisted of 199,753 instances in the format described in Section 4.2. The queries in the training dataset were randomly sampled from Bing's search logs from a period between January 2012 and September 2014. Human judges rated the documents on a five-point scale (*perfect*, *excellent*, *good*, *fair*, and *bad*). The document body text was retrieved from Bing's Web document index. We used proprietary parsers for extracting the body text from raw HTML content. All query and document text were normalized by down-casing and removing all non-alphanumeric characters.

We considered two different test sets, both sampled from Bing search logs. The *weighted* set consisted of queries sampled per their frequency in the search logs. Thus, frequent queries were well-represented in this dataset. Queries were sampled between October 2014 and December 2014. The *unweighted* set consisted of queries sampled uniformly from the entire population of unique queries. The queries in this samples removed the bias toward popular queries found in the weighted set. The unweighted queries were sampled between January 2015 and June 2015.

Because all of our datasets were derived from sampling real query logs and because queries will naturally repeat, there was some overlap in queries between the training and testing sets. Specifically, 14% of the testing queries in the weighted set occurred in the training set, whereas only 0.04% of the testing queries in the unweighted set occurred in the training set. We present both results for those who may be in environments with repeated queries (as is common in production search engines) and for those who may be more interested in cold start situations or tail queries. Table 1 summarizes statistics for the two test sets.

4.2 Training

Besides the architecture (Figure 3), our model has the following free parameters: the maximum order of the character-based representation for the distributed model (G), the number of negative documents to sample at training time (N), the dropout rate, and the learning rate.

We used a maximum order of five for our character n-graphs in the distributed model. Instead of using the full 62,193,780-dimensional vector, we only considered the top 2,000 most popular n-graphs, resulting 36 unigraphs (a-z and 0-9), 689 bigraphs, 1149 trigraphs, 118 4-graphs, and eight 5-graphs.

When training our model (Section 3.3), we sampled four negative documents for every one relevant document. More precisely, for

each query we generated a maximum of one training sample of each form, (1) One *excellent* document with four *fair* documents (2) One *excellent* document with four *bad* documents (3) One *good* document with four *bad* documents. Pilot experiments showed that treating documents judged as *fair* or *bad* as the negative examples resulted in significantly better performance, than when the model was trained with randomly sampled negatives. For training, we discarded all documents rated as *perfect* because a large portion of them fall under the navigational intent, which can be better satisfied by historical click based ranking signals.

The dropout rate and the learning rate were set to 0.20 and 0.01, respectively, based on a validation set. We implemented our model using CNTK [40] and trained the model with stochastic gradient descent based optimization (with automatic differentiation) on a single GPU.[3] It was necessary to use a small minibatch size of 8 to fit the whole data in GPU memory.

4.3 Baselines

Our baselines capture the individual properties we outlined in Section 2. Exact term matching is effectively performed by many classic information retrieval models. We used the Okapi BM25 [33] and query likelihood (QL) [32] models as representative of this class of model. We used Indri[4] for indexing and retrieval.

Match positions are handled by substantially fewer models. Metzler's dependence model (DM) [23] provides an inference network approach to modeling term proximity. We used the Indri implementation for our experiments.

Inexact term matching received both historic and modern treatments in the literature. Deerwester *et al.* originally presented latent semantic analysis (LSA) [5] as a method for addressing vocabulary mismatch by projecting words and documents into a lower-dimension latent space. The dual embedding space model (DESM) [27, 28] computes a document relevance score by comparing every term in the document with every query term using pre-trained word embeddings. We used the same pre-trained word embeddings dataset that the authors made publicly available online for download[5]. These embeddings, for approximately 2.8M words, were previously trained on a corpus of Bing queries. In particular, we use the $\text{DESM}_{\text{IN-OUT}}$ model, which was reported to have the best performance on the retrieval task, as a baseline in this paper. Both the deep structured semantic model (DSSM) [16] and its convolutional variant CDSSM [37] consider only the document title for matching with the query. While some negative results have been reported for title-based DSSM and CDSSM on the *ad hoc* document retrieval tasks [12, 30], we included document-based variants appropriately retrained on the same set of positive query and document pairs as our model. As with the original implementation we choose the non-relevant documents for training by randomly sampling from the document corpus. For the CDSSM model, we concatenated the trigraph hash vectors of the first T terms of the body text followed by a vector that is a sum of the trigraph hash vectors for the remaining terms. The choice of T was constrained by memory requirements, and we pick 499 for our experiments.

The DRMM model [12] uses a DNN to perform term matching, with few hundred parameters, over histogram-based features. The histogram features, computed using exact term matching and pre-trained word embeddings based cosine similarities, ignoring the actual position of matches. We implemented the $\text{DRMM}_{\text{LCH}\times\text{IDF}}$ variant of the model on CNTK [40] using word embeddings trained on a corpus of 341,787,174 distinct sentences randomly sampled from Bing's Web index, with a corresponding vocabulary of 5,108,278 words. Every training sample for our model was turned into four corresponding training samples for DRMM, comprised of the query, the positive document, and each one of the negative documents. This guaranteed that both models observed the exact same pairs of positive and negative documents during training. We adopted the same loss function as proposed by Guo *et al.*

Table 2: Performance on test data. All duet runs significantly outperformed our local and distributed model ($p < 0.05$). All duet runs also outperformed non-neural and neural baselines. The difference between the duet model and the best performing baseline per dataset and position (italics) is statistically significant ($p < 0.05$). The best NDCG performance on each dataset and position is highlighted in bold.

(a) weighted

	NDCG@1	NDCG@10
Non-neural baselines		
LSA	22.4	44.2
BM25	24.2	45.5
DM	24.7	46.2
QL	24.6	46.3
Neural baselines		
DRMM	24.3	45.2
DSSM	25.8	48.2
CDSSM	*27.3*	48.2
DESM	25.4	*48.3*
Our models		
Local model	24.6	45.1
Distributed model	28.6	50.5
Duet model	**32.2**	**53.0**

(b) unweighted

	NDCG@1	NDCG@10
Non-neural baselines		
LSA	31.9	62.7
BM25	34.9	63.3
DM	35.0	63.4
QL	34.9	63.4
Neural baselines		
DRMM	*35.6*	*65.1*
DSSM	34.3	64.4
CDSSM	34.3	64.0
DESM	35.0	64.7
Our models		
Local model	35.0	64.4
Distributed model	35.2	64.9
Duet model	**37.8**	**66.4**

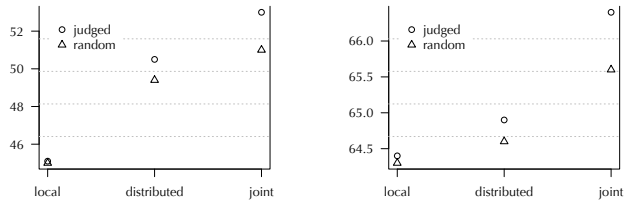
4.4 Evaluation

All evaluation and empirical analysis used the normalized discounted cumulative gain (NDCG) metric computed at positions one and ten [18]. All performance metrics were averaged over queries for each run. Whenever testing for significant differences in performance, we used the paired t-test with a Bonferroni correction.

[3] A CNTK implementation of the Duet model is available at https://github.com/bmitra-msft/NDRM/blob/master/notebooks/Duet.ipynb

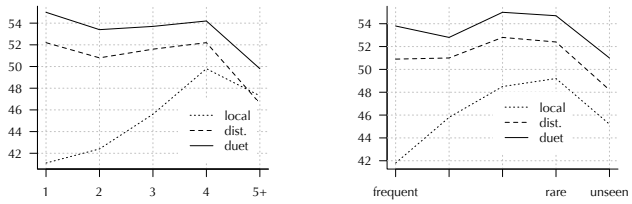
[4] http://www.lemurproject.org/indri/

[5] https://www.microsoft.com/en-us/download/details.aspx?id=52597



(a) Weighted set (b) Unweighted set

Figure 4: The duet model demonstrates significantly better NDCG performance ($p < 0.05$) on both test sets when trained with judged non-relevant documents as the negative examples, instead of randomly sampling them from the document corpus. The distributed model also shows statistically significant NDCG gain ($p < 0.05$) on the weighted set, and a non-statistically significant NDCG gain on the unweighted set.



(a) Model performance by query length (b) Model performance by term rarity

Figure 5: NDCG performance of different models by length of query and how rare the rarest query term is in the training data. For the rare term analysis, we place all query terms into one of five categories based on their occurrence counts in the training data. Then we then categorize each query in the test dataset based on the frequency of the rarest term belongs in the query. We include a category for queries with at least one term which has no occurrences in the training data.

5. RESULTS

Table 2 reports NDCG based evaluation results on two test datasets for our model and all the baseline models. Our main observation is that the duet model performs significantly better than the individual local and distributed models. This supports our underlying hypothesis that matching in a latent semantic space can complement exact term matches in a document ranking task, and hence a combination of the two is more appropriate. Note that the NDCG numbers for the local and the distributed models correspond to when these DNNs are trained individually, but for the 'duet' the two DNNs are trained together as part of a single neural network.

Among the baseline models, including both traditional and neural network based models, CDSSM and DESM achieve the highest NDCG at position one and ten, respectively, on the weighted test set. On the unweighted test set DRMM is our best baseline model at both rank positions. The duet model demonstrates significant improvements over all these baseline models on both test sets and at both NDCG positions.

We also tested our independent local and distributed models against their conceptually closest baselines. Because our local model captures both matching and proximity, we compared performance to dependence models (DM). While the performance in terms of NDCG@1 is statistically indistinguishable, both NDCG@10 results are statistically significant ($p < 0.05$). We compared our distributed model to the best neural model for each test set and metric. We found no statistically significant difference except for NDCG@10 for the weighted set.

We were interested in testing our hypotheses that training with labeled negative documents is superior to training with randomly sampled documents presumed to be negative. We conducted an experiment training with negative documents following each of the two protocols. Figure 4 shows the results of these experiments. We found that, across all our models, using judged nonrelevant documents was more effective than randomly sampling documents from the corpus and considering them as negative examples.

6. DISCUSSION

Our results demonstrated that our joint optimization of local and distributed models provides substantial improvement over all baselines. Although the independent models were competitive with existing baselines, the combination provided a significant boost.

We also confirmed that using judged negative documents should be used when available. We speculate that training with topically-similar (but non-relevant) documents allows the model to better discriminate between the confusable documents provided by an earlier retrieval stage. This sort of staged ranking, first proposed by Cambazoglu et al. [3], is now a common web search engine architecture.

In Section 4.3 we described our baseline models according to which of the properties of effective retrieval systems, that we outlined in Section 2, they incorporate. It is reasonable to expect that models with certain properties are better suited to deal with certain segments of queries. For example, the relevant Web page for the query "what channel are the seahawks on today" may contain the name of the actual channel (e.g., "ESPN" or "FOX") and the actual date for the game, instead of the terms "channel" or "today". A retrieval model that only counts repetitions of query terms is likely to retrieve less relevant documents for this query – compared to a model that considers "ESPN" and "FOX" to be relevant document terms. In contrast, the query "pekarovic land company", which may be considered as a tail navigational intent, is likely to be better served by a retrieval model that simply retrieves documents containing many matches for the term "pekarovic". A representation learning model is unlikely to have a good representation for this rare term, and therefore may be less equipped to retrieve the correct documents. These anecdotal examples agree with the results in in Table 2 that show that on the weighted test set all the neural models whose main focus is on learning distributed representations of text (duet model, distributed model, DESM, DSSM, and CDSSM) perform better than the models that only look at patterns of term matches (local model and DRMM). We believe that this is because the DNNs can learn better representations for more popular queries, and perform particularly well on this segment. Figure 5 provides further evidence towards this hypothesis by demonstrating that the distributed model has a larger NDCG gap with the local model for queries containing more popular terms, and when the number of terms in the query is small. The duet model, however, is found to perform better than both the local and the distributed models across all these segments.

To better understand the relationship of our models to existing baselines, we compared the per-query performance amongst all models. We conjecture that similar models should perform similarly for the same queries. We represented a retrieval model as a vector where each position of the vector contains the performance of the model on a different query. We randomly sample two thousand queries from our weighted test set and represent all ranking models

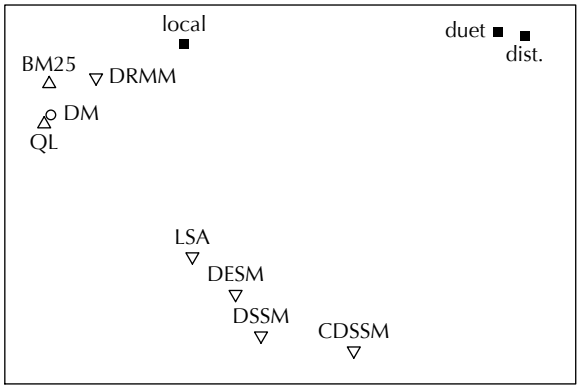


Figure 6: Principal component analysis of models based on retrieval performance across testing queries. Models using exact term matches (△), proximity (○), and inexact matches (▽) are presented. Our models are presented as black squares.

as vectors of their NDCG values against these two thousand queries. We visualized the similarity between models by projecting using principal component analysis on the set of performance vectors. The two-dimensional projection of this analysis is presented in Figure 6. The figure largely confirms our intuitions about properties of retrieval models. Models that use only local representation of terms are closer together in the projection, and further away from models that learn distributed representations of text. Interestingly, the plot does not distinguish between whether the underlying model is based on a neural network based or not – with neural networks of different retrieval properties appearing in each of the three clusters.

Another interesting distinction between deep neural models and traditional approaches is the effect of the training data size on the performance of the model. BM25 has very few parameters and can be applied to new corpus or task with almost no training. On the other hand, DNNs like ours demonstrate significant improvements when trained with larger datasets. Figure 7 shows that the effect of training data size particularly pronounced for the duet and the distributed models that learns representations of text. The trends in these plots indicate that training on even larger datasets may result in further improvements in model performance over what is reported in this paper. We believe this should be a promising direction for future work.

A last consideration when comparing these models is runtime efficiency. Web search engines receive tens of thousands of queries per second. Running a deep neural model on raw body text at that scale is a hard problem. The local sub-network of our model operates on the term interaction matrix that should be reasonable to generate using an inverted index. For the distributed model, it is important to note that the 300×899 dimensional matrix representation of the document, that is used to compute the Hadamard product with the query, can be pre-computed and stored as part of the document cache. At runtime, only the Hadamard product and the subsequent part of the network needs to be executed. Such caching strategies, if employed effectively, can mitigate large part of the runtime cost of running a DNN based document ranking model at scale.

7. RELATED WORK

Representations of data can be local or distributed. In a local representation, a single unit represents an entity, for example there is a particular memory cell that represents the concept of a grandmother. That cell should be active if and only if the concept of a grandmother is present. By contrast, in a distributed representation, the concept of grandmother would be represented by a pattern of active cells. Hinton et al. [13] provides an overview contrasting distributed and local representations, listing their good and bad points. In a distributed representation, an activation pattern that has some errors or other differences from past data can still be mapped to the entity in question and to related entities, using a similarity function. A local representation lacks this robustness to noise and ability to generalize, but is better at precisely storing a large set of data.

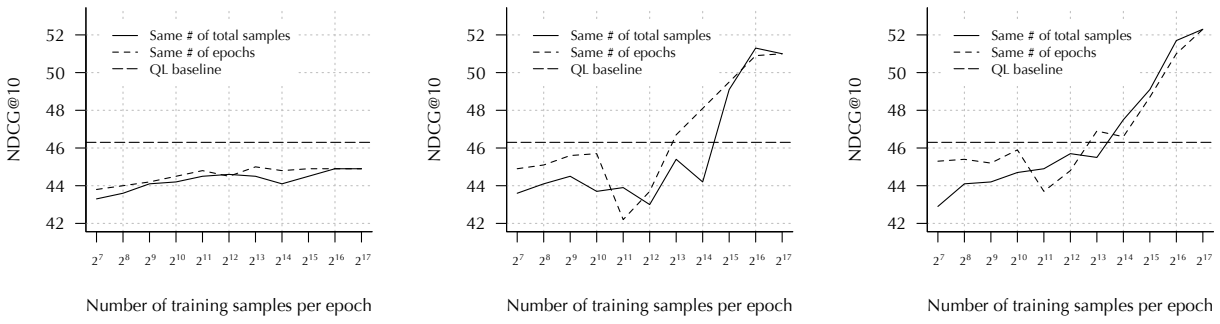
This paper considers local and distributed representations of queries and documents for use in Web page ranking. Our measure of ranking quality is NDCG [17], which rewards a ranker for returning documents with higher gain nearer to the top, where gain is determined based on labels from human relevance assessors. We describe different ranking methods in terms of their representations and how this should help them achieve good NDCG.

Exact term matching models such as BM25 [33] and query likelihood [32] tend to rank a document higher if it has a greater number of query term matches, while also potentially employing a variety of smoothing, weighting and normalization approaches. Such exact matching is done with a local representation of terms. Exact match systems do not depend on a large training set, since they do not need to learn a distributed representation of queries and documents. They are useful in cases where the relevant documents contain exactly the query terms entered by the user, including very rare or new vocabulary, since new terms can be incorporated with no adjustments to the underlying model. They can also be extended to reward matches of query phrases and proximity [23].

To deal with the vocabulary mismatch problem that arises with local representations, it is possible to do document ranking using a distributed representation of terms. Mikolov et al. [24] developed the popular word2vec embedding approach that has been used in several retrieval studies. Zheng and Callan [41] use term embeddings as evidence for term weighting, learning regression models to optimize weighting in a language modeling and a BM25 retrieval model. Ganguly et al. [8] used term embeddings for smoothing in the language modeling approach of information retrieval. Nalisnick et al. [28] used dual embeddings, one for document terms and one for query terms, then ranked based on the all-pairs similarity between vectors. Diaz et al. [6] used term embeddings to generate query expansion candidates in the language modelling retrieval framework, also finding better performance when training a specialized term embedding. Other papers incorporating word embeddings include [10, 11, 34].

Pang et al. [31] propose the use of matching matrices to represent the similarity of short texts, then apply a convolutional neural network inspired by those in computer vision. They populate the matching matrix using both local and distributed term representations. In the local representation, an exact match is used to generate binary indicators of whether the ith term of one text and jth term of the other are the same, as in our local model. In the distributed representation, a pre-trained term embedding is used instead, populating the match matrix with cosine or inner product similarities. The method works for some problems with short text, but not for document ranking [30]. However, by using the match matrix to generate summary statistics it is possible to make the method work well [12], which is our DRMM baseline.

These term embeddings are a learned representation of language, but in most cases, they are not learned on query-document relevance labels. More often they are trained based on a a corpus, where a term's representation is learned from its surrounding terms or other document context. The alternative, learning a representation based on NDCG labels, is in keeping with recent progress in deep learning.



(a) Local model (b) Distributed model (c) Duet model

Figure 7: We study the performance of our model variants when trained with different size datasets. For every, dataset size we train two models – one for exactly one epoch and another one with multiple epochs such that the total number of training samples seen by the model during training is 131,072.

Deep models have multiple layers that learn distributed representations with multiple levels of abstraction. This kind of representation learning, along with other factors such as the availability of large labelled data sets, has yielded performance improvements on a variety of tasks such as speech recognition, visual object recognition and object detection [20].

This paper learns a text representation end-to-end based on query-document ranking labels. This has not been done often in related work with document body text, but we can point to related papers that use short text such as title, for document ranking or related tasks. Huang et al. [16] learn a distributed representation of query and title, for document ranking. The input representation is character trigraphs, the training procedure asks the model to rank clicked titles over randomly chosen titles, and the test metric is NDCG with human labels. Shen et al. [36] developed a convolutional version of the model. These are our DSSM and CDSSM baselines. Other convolutional models that match short texts using distributed representations include [15, 35], also showing good performance on short text ranking tasks.

Outside of document ranking, learning text representations for the target task has been explored in the context of other IR scenarios, including query classification [21], query auto-completion [26], next query prediction [25, 38], and entity extraction [9].

8. CONCLUSION

We propose a novel document ranking model composed of two separate deep neural network sub-models, one that matches using a local representation of text, and another that learns a distributed representation before matching. The duet of these two neural networks demonstrated a higher performance than the solo models on the document ranking task as well as significant improvements over all baselines, including both traditional IR baselines and other recently proposed models based on shallow and deep neural networks. Our analysis indicate that these models may achieve even more substantial improvements in the future with much larger datasets.

Acknowledgements. The authors are grateful to Rich Caruana, Abdelrahman Mohamed, Pushmeet Kohli, Emine Yilmaz, Filip Radlinski, David Barber, David Hawking, and Milad Shokouhi for the insightful discussions and feedback during this work, and to Frank Seide and Dong Yu for their incredible support with CNTK.

References

[1] M. Bendersky and W. B. Croft. Modeling higher-order term dependencies in information retrieval using query hypergraphs. In *Proc. SIGIR*, pages 941–950. ACM, 2012.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3: 993–1022, 2003.

[3] B. B. Cambazoglu, V. Plachouras, and R. Baeza-Yates. Quantifying performance and quality gains in distributed web search engines. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 411–418, New York, NY, USA, 2009. ACM. doi: 10.1145/1571941.1572013.

[4] N. Craswell, W. B. Croft, J. Guo, B. Mitra, and M. de Rijke. Report on the sigir 2016 workshop on neural information retrieval (neu-ir). 50(2):96–103, 2016.

[5] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[6] F. Diaz, B. Mitra, and N. Craswell. Query expansion with locally-trained word embeddings. In *Proc. ACL*, 2016.

[7] H. Fang and C. Zhai. An exploration of axiomatic approaches to information retrieval. In *Proc. SIGIR*, pages 480–487. ACM Press, 2005.

[8] D. Ganguly, D. Roy, M. Mitra, and G. J. Jones. Word embedding based generalized language model for information retrieval. In *Proc. SIGIR*, pages 795–798. ACM, 2015.

[9] J. Gao, P. Pantel, M. Gamon, X. He, L. Deng, and Y. Shen. Modeling interestingness with deep neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2014.

[10] M. Grbovic, N. Djuric, V. Radosavljevic, and N. Bhamidipati. Search retargeting using directed query embeddings. In *Proc. WWW*, pages 37–38, 2015.

[11] M. Grbovic, N. Djuric, V. Radosavljevic, F. Silvestri, and N. Bhamidipati. Context-and content-aware embeddings for query rewriting in sponsored search. In *Proc. SIGIR*, pages 383–392. ACM, 2015.

[12] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In *Proc. CIKM*, 2016.

[13] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Distributed Representations, pages 77–109. MIT Press, Cambridge, MA, USA, 1986.

[14] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. SIGIR*, pages 50–57. ACM, 1999.

[15] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *Proc. NIPS*, pages 2042–2050, 2014.

[16] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proc. CIKM*, pages 2333–2338. ACM, 2013.

[17] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM TOIS*, 20(4):422–446, 2002.

[18] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *TOIS*, 20(4):422–446, 2002.

[19] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[21] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. *Proc. NAACL, May 2015*.

[22] Z. Lu and H. Li. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375, 2013.

[23] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *Proc. SIGIR*, pages 472–479. ACM, 2005.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*, pages 3111–3119, 2013.

[25] B. Mitra. Exploring session context using distributed representations of queries and reformulations. In *Proc. SIGIR*, pages 3–12. ACM, 2015.

[26] B. Mitra and N. Craswell. Query auto-completion for rare prefixes. In *Proc. CIKM*. ACM, 2015.

[27] B. Mitra, E. Nalisnick, N. Craswell, and R. Caruana. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137*, 2016.

[28] E. Nalisnick, B. Mitra, N. Craswell, and R. Caruana. Improving document ranking with dual word embeddings. In *Proc. WWW*, 2016.

[29] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(4):694–707, 2016.

[30] L. Pang, Y. Lan, J. Guo, J. Xu, and X. Cheng. A study of matchpyramid models on ad-hoc retrieval. In *Neu-IR '16 SIGIR Workshop on Neural Information Retrieval*, 2016.

[31] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. Text matching as image recognition. In *Proc. AAAI*, 2016.

[32] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. SIGIR*, pages 275–281. ACM, 1998.

[33] S. Robertson and H. Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[34] D. Roy, D. Paul, M. Mitra, and U. Garain. Using word embeddings for automatic query expansion. In *Neu-IR '16 SIGIR Workshop on Neural Information Retrieva*, 2016.

[35] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proc. SIGIR*, pages 373–382. ACM, 2015.

[36] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proc. CIKM*, pages 101–110. ACM, 2014.

[37] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proc. WWW*, pages 373–374, 2014.

[38] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, and J.-Y. Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proc. CIKM*, pages 553–562. ACM, 2015.

[39] X. Wei and W. B. Croft. Lda-based document models for ad-hoc retrieval. In *Proc. SIGIR*, pages 178–185. ACM, 2006.

[40] D. Yu, A. Eversole, M. Seltzer, K. Yao, Z. Huang, B. Guenter, O. Kuchaiev, Y. Zhang, F. Seide, H. Wang, et al. An introduction to computational networks and the computational network toolkit. Technical report, Tech. Rep. MSR, Microsoft Research, 2014, http://codebox/cntk, 2014.

[41] G. Zheng and J. Callan. Learning to reweight terms with distributed representations. In *Proc. SIGIR*, pages 575–584. ACM, 2015.

[42] Y. Zhou and W. B. Croft. Document quality models for web ad hoc retrieval. In *Proc. CIKM*, pages 331–332. ACM Press, 2005.