Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval

Zhuyun Dai
Carnegie Mellon University
zhuyund@cs.cmu.edu

Jamie Callan
Carnegie Mellon University
callan@cs.cmu.edu

ABSTRACT

Term frequency is a common method for identifying the importance of a term in a query or document. But it is a weak signal, especially when the frequency distribution is flat, such as in long queries or short documents where the text is of sentence/passage-length. This paper proposes a Deep Contextualized Term Weighting framework that learns to map BERT's contextualized text representations to context-aware term weights for sentences and passages.

When applied to passages, `DeepCT-Index` produces term weights that can be stored in an ordinary inverted index for passage retrieval. When applied to query text, `DeepCT-Query` generates a weighted bag-of-words query. Both types of term weight can be used directly by typical first-stage retrieval algorithms. This is novel because most deep neural network based ranking models have higher computational costs, and thus are restricted to later-stage rankers.

Experiments on four datasets demonstrate that `DeepCT`'s deep contextualized text understanding greatly improves the accuracy of first-stage retrieval algorithms.

KEYWORDS

Text Understanding, Neural-IR, Term Weighting

1 INTRODUCTION

State-of-the-art search engines use ranking pipelines in which an efficient *first-stage* uses a query to fetch an initial set of documents from the document collection, and one or more *re-ranking* algorithms improve and prune the ranking. Typically the first stage ranker is a Boolean, probabilistic, or vector space bag-of-words retrieval model that fetches information from an inverted index. One key characteristic of first-stage ranking algorithms is how they quantify the contribution of each query or document term. Most retrieval methods use frequency-based signals such as document and query term frequency (*tf, qtf*) to determine the context-specific

Table 1: Two passages that mention 'stomach' twice. Only the first passage is about the topic 'stomach'. This paper proposes a method to weight terms by their roles in a specific text context, as shown by the heatmap over terms.

In some cases, an upset **stomach** is the result of an allergic reaction to a certain type of food. It also may be caused by an irritation. Sometimes this happens from consuming too much alcohol or caffeine. Eating too many fatty foods or too much food in general may also cause an upset **stomach**. All parts of the body (muscles , brain, heart, and liver) need energy to work. This energy comes from the food we eat. Our bodies digest the food we eat by mixing it with fluids(acids and enzymes) in the **stomach**. When the **stomach** digests food, the carbohydrate (sugars and starches) in the food breaks down into another type of sugar, called glucose.

importance of a term, and and inverse document frequency (*idf*) or a similar value to determine its importance globally.

Frequency-based term weights have been a huge success, but they are a crude tool. Term frequency does not necessarily indicate whether a term is important or central to the meaning of the text, especially when the frequency distribution is flat, such as in *sentences and short passages*. Table 1 shows two passages from the MS MARCO machine reading comprehension dataset [21]. If a user searches for 'stomach', the two passages will be considered similarly relevant by frequency-based retrieval models because both mention the query word 'stomach' twice; but the second passage is actually off-topic. To identify which terms are central requires a deeper understanding that considers the meaning of a word, the meaning of the entire text, and the role the word plays in the text.

Recently, there has been rapid progress in text understanding with the introduction of *deep contextualized word representations* such as ELMo [25] and BERT [11]. These methods learn a neural language model from large text corpora. Each token is assigned a representation that is a function of not only the token itself, but also of the entire text. These neural language models have been shown to capture the characteristics of a word's semantics and syntax, and more importantly, how they vary across linguistic contexts [32].

This paper shows how to improve first-stage retrieval models using the contextualized word representations generated by BERT [11]. We present the Deep Contextualized Term Weighting framework (DeepCT). DeepCT is trained in a supervised manner to learn a BERT-based contextualized word representation model and a mapping function from representations to term weights. As a word's representation depends on its specific context, the estimated importance for the same term varies with the context. DeepCT can take up to 512 text tokens [1]. It can makes use of the linguistic context in sentence/passage-long text to generate more representative

---

[1] This is limited by the input length limitation of BERT [11]

term weights, helping the cases where the original term frequency distribution is flat, such as retrieving passages or searching long queries.

One use of DeepCT is to identify **essential passage terms for passage retrieval**. As shown in Table 1, passages usually have flat term weight distribution, making term-frequency based retrieval less effective. We develop a novel DeepCT-Index that offline weights and indexes terms in passage-long documents. It trains a DeepCT model to predict whether a passage term is likely to appear in relevant queries. The trained model is applied to every passage in the collection. This inference step is query-independent, allowing it to be done offline during indexing. The context-based passage term weights are scaled to *tf*-like integers that are stored in an ordinary inverted index that can be searched efficiently by common first-stage retrieval models.

Another use of DeepCT is to identify **important query terms in long queries**. For long queries that mention many terms and concepts, it is important to identify which are central. We follow a query term weighting framework proposed by Zheng and Callan [36] to develop DeepCT-Query. It trains DeepCT with signals from relevant query-document pairs, weighting query terms by their possibilities to be mentioned by relevant documents. The predictions are used to generate weighted queries that can be used with widely-used retrieval models such as BM25 [26] and query likelihood [15].

Our experiments demonstrate that DeepCT generates effective representations for both passages and queries that lead to large improvements in first-stage retrieval accuracy. Analysis shows that DeepCT's main advantage is its ability to estimate term importance using the meaning of the context rather than term frequency signals, allowing the retrieval model to differentiate between key terms and other frequently mentioned but non-central terms.

Most of the prior neural-IR research [9, 13], including recent research on leveraging BERT for IR [8, 22], focused on re-ranking stages due to the complexity of neural models. Our work adds the ability to improve existing first-stage rankers. More accurate first-stage document rankings provide better candidates for downstream re-ranking, which improves end-to-end accuracy and/or efficiency.

Section 2 discusses related work. Section 3 describes the Deep Contextualized Term Weighting framework (DeepCT), its use for passage indexing (DeepCT-Index), and its use for for query weighting (DeepCT-Query). Sections 5-6 describe our methodologies and experiments. Section 7 concludes [2].

2 RELATED WORK

Two types of work are related: term weighting, and neural approaches for early-stage ranking.

Term Weighting. Bag-of-words retrieval models such as BM25 [26] and query likelihood [17] are the foundation of modern search engines due to their efficiency and effectiveness. Most retrieval models use frequency-based signals such as *tf*, *ctf*, and *df* to estimate how well each term represents a text (query, document). A rich research literature studies how to better estimate term importance (e.g., [2, 3, 5, 18, 27, 36]), however frequency-based signals continue to dominate in both industry and academia.

For document term weighting, the most widely-used alternatives to frequency-based models are graph-based methods [5, 18, 27]. Graph-based approaches build a document graph, where the nodes represent terms and edges represent term co-occurrence within a maximum distance. Terms are ranked by graph ranking algorithms such as PageRank, and scores are indexed for retrieval [5, 27].

Recently, a few neural IR approaches study document term weight from word embeddings [10], because word embeddings have the ability to encode some aspects of word meanings that are useful for determining word importance. They add a term-gating sub-network into neural re-rankers that learn global *idf*-like term weights from word embeddings. It is not clear whether they can improve the initial ranking stage.

For query term weighting, one line of research focuses on feature-based approaches [2–4]. They require running the query against a document collection to generate features. These methods improve search accuracy compared to frequency-based query term weighting, but the use of pseudo-relevant feedback causes extra computational cost. To predict query term weights from just the text content, Zheng and Callan [36] proposed a word-embedding based method called DeepTR. DeepTR constructs a feature vector for each query term using the difference between the term's word2vec [19] embedding and the average query embedding. It then learns a regression model to map the feature vector onto the term's ground truth weight (term recall weight [35]). The estimated weights are used to generate bag-of-words queries that can be searched in first-stage retrieval. Later, Guo et al. [13] used a similar word-embedding based query term weighting method in neural re-rankers.

As discussed above, in both document and query term weighting problems, word embeddings have gained increasing attention due to their ability to encode certain aspects of text meaning [10, 13, 36]. However, word embedding based approaches face a critical issue in capturing the context, as classic word embeddings are context-independent – a word always has the same embedding regardless of the text context. Recently, contextualized neural language models have been proposed to model the linguistic context [11, 25]. In contextualized neural language models, a word's representation is a function of the entire text input. These representations were shown to capture a word's certain syntactic features and semantic features in the text content [32]. Currently, the best performing neural language model is BERT [11]. BERT has received a lot of attention for IR, mainly focused on using it as a black-box re-ranking model to predict query-document relevance scores [8, 22]. This work attempts to leverage the BERT's advantages to explicitly model query/document term importance for efficient first-stage retrieval.

Neural Approaches for Early Stage Ranking. Most neural ranking models use continuous representations that enable a query to match every document to some degree, which makes them impractical for first-stage rankers. Recent research addresses this efficiency problem in two ways, both using latent representations. One approach learns low dimensional dense embeddings of queries and documents that support fast retrieval using approximate nearest neighbor search in the embedding space [1], however there is an efficiency vs. accuray tradeoff [6]. A second approach is to learn high-dimensional but sparse latent representations in which queries and documents are represented by a set of 'latent words' [28, 34].

Sparse 'latent words' allow inverted index style look up, which is efficient, but also introduces the specificity vs. exhaustiveness tradeoff found in all controlled vocabularies [29].

Recently, Nogueira et al. [23] proposed `Doc2Query` that uses neural models to modify the discrete word-based document representations. It generates potential queries from documents using neural machine translation and indexes those queries as document expansion terms. We are not aware of other first-stage neural models that use the discrete word-based text representations and large vocabularies that have been fundamental to modern search engines.

3 DEEP CONTEXTUALIZED DOCUMENT AND QUERY TERM WEIGHTING

This section presents the Deep Contextualized Term Weighting framework (DeepCT), and how it is used to weight query terms (`DeepCT-Query`) and index documents (`DeepCT-Index`).

3.1 DeepCT Framework

`DeepCT` includes two main components: generating contextualized word embeddings through BERT, and predicting term weights through linear regression.

Contextualized word embedding generation. To estimate the importance of a word in a specific text, the most critical problem is to generate features that characterize a word's relationships to the text context. Recent contextualized neural language models like ELMo [25] and BERT [11] have been shown to capture such properties through a deep neural network. `DeepCT` leverages one of the best performing models, BERT, to extract a word's context features. BERT uses an attention mechanism where a word gradually absorbs context information based on its attention to every other word in the same text. Devlin et al. [11] provide more detail.

Map to target weights. The contextualized word embedding is a feature vector that characterizes the word's syntactic and semantic role in a given context. `DeepCT` linearly combines the features into a word importance score:

$$\hat{y}_{t,c} = \vec{w}T_{t,c} + b \tag{1}$$

where $T_{t,c}$ is token t's contextualized embedding in text c; and, $\vec{w}$ and b are the linear combination weights and bias.

`DeepCT` is trained with a per-token regression task. Given the ground truth term weight for every word in text c, denoted as $\{y_{1,c}, \ldots, y_{N,c}\}$, `DeepCT` aims to minimize the mean square error (MSE) between the predicted weights $\hat{y}$ and the target weights y:

$$loss_{MSE} = \sum_{c} \sum_{t} (y_{t,c} - \hat{y}_{t,c})^2. \tag{2}$$

The range of possible predicted term weights $\hat{y}_{t,c}$ is $(-\infty, \infty)$, but in practice most predictions fall into $[0, 1]$ because, as shown in Sections 3.3 and 3.3, the ground truth weights are $[0, 1]$. Our query/document weighting approaches accept any non-negative weights; terms with negative weights are discarded.

The tokenization step of BERT generates subwords for unseen words (e.g. "DeepCT" is tokenized into "deep" and "##ct"). We use the weight for the first subword as the weight of the entire word; other subwords are masked out when computing the MSE loss.

The `DeepCT` model, from BERT to the regression layer, is optimized end-to-end. The BERT component is initialized with a pretrained BERT model to reduce over-fitting. It is fine-tuned to align the contextualized word embeddings with the term-prediction task. The last regression layer is learned from scratch.

`DeepCT` is a general framework that learns term weights with respect to a linguistic context. `DeepCT` can *learn* different definitions of term importance depending on how ground truth term weights are defined. The predicted term weights can also be *used* differently depending on the task. Below, we describe two approaches to using the `DeepCT` framework to improve first-stage retrieval.

3.2 Index Passages with DeepCT

A novel use of `DeepCT` is to identify terms that are central to the meaning of a passage, or a passage-long document, for efficient and effective passage/short-document retrieval. In this paper, We will refer to both passages in passage retrieval and passage-long documents in short-document retrieval as 'passages'.

As shown in the 'stomach' example in Table 1, classic term frequency signals cannot tell whether the text is centered around a term or just mentions that term when discussing some other topic. This issue is especially difficult in first-stage full-collection ranking, where complex features and models are too expensive to apply. `DeepCT-Index` uses `DeepCT` to weight passage terms and stores the weights in a typical inverted index.

Target Term Weights for Training `DeepCT`. Proper target term weights should reflect whether a term is essential to the passage or not. We propose *query term recall* as an estimation of the ground truth passage term importance:

$$QTR(t, d) = \frac{|Q_{d,t}|}{|Q_d|}. \tag{3}$$

Q_d is the set of queries for which passage,$= d$ is judged relevant. $Q_{d,t}$ is the subset of Q_d that contains term t, and $QTR(t, d)$ is the query term recall weight for t in d. QTR is in the range of $[0, 1]$. Query term recall is based on the assumption that search queries can reflect the key idea of a document. Words that appear in relevant queries are more important than other words in the document.

The training requires relevant query-passage pairs. The model takes the text content of a passage, make predictions, and compute the loss with target weights generated from its related queries. During inference, the model needs only the passage.

Index with predicted term weights. Once `DeepCT` learns model parameters, it can make estimates for any passage without the need of queries. This allows estimated term weights to be calculated and stored during offline indexing. The index can be searched efficiently by online services.

We apply the trained `DeepCT` model to all passages in the collection. The predicted weights are scaled from a [0..1] range to an integer range that can be used with existing retrieval models. We call this weight $\text{TF}_{\text{DeepCT}}$ to convey that it is an alternate way of representing the importance of t erm t in passage d:

$$\text{TF}_{\text{DeepCT}}(t, d) = round(\hat{y}_{t,d} * N), \tag{4}$$

where $\hat{y}_{t,d}$ is the predicted term weights for term t in passage d. N scales the predicted weights into a integer range. We use $N = 100$ in this work as two digits of precision is sufficient for this task.

$\text{TF}_{\text{DeepCT}}$ is used to *replace* the original TF field in the inverted index. The postings of term t changes from [docid(d), TF(t, d)] into [docid(d), $\text{TF}_{\text{DeepCT}}(t, d)$]. The new index, DeepCT-Index, can be searched by mainstream bag-of-words retrieval model like BM25 or query likelihood model (QL). The context-based term weight $\text{TF}_{\text{DeepCT}}$ is expect to bias the retrieval models to central terms in the pasessag, preventing off-topic passages being retrieved.

Efficiency. DeepCT-Index does not add latency to the search system. The main difference between DeepCT-Index and a typical inverted index is that the term importance weight is based on $\text{TF}_{\text{DeepCT}}$ instead of TF. This calculation is done offline. No new posting lists are created, thus the query latency does not become longer. To the contrary, a side-effect of Eq 4 is that $\text{TF}_{\text{DeepCT}}$ of some terms becomes 0. This may be viewed as a form of index pruning [7]. We leave that aspect of this work for future investigation.

3.3 Query Term Weighting with DeepCT

Another straightforward use of DeepCT for IR is to weight query terms in long queries. For long queries that mention many terms and concepts, it is important to identify which are central. For example, given the query *"Find locations of volcanic activity which occurred within the present day boundaries of the U.S and its territories"*, an ideal system would understand that *"volcanic activity"* is the key concept, and that *"boundaries of the U.S"* maybe not be necessary in some corpora. Zheng and Callan [36] proposed a word2vec-based query term re-weighting framework, called DeepTR, that efficiently re-weights bag-of-words queries. We follow the DeepTR framework, but replace the word2vec-based model with DeepCT. We call the proposed approach DeepCT-Query.

Target Term Weights for Training DeepCT. Inspired by Zheng and Callan [36], DeepCT-Query uses *term recall* [35]:

$$TR(t, q) = \frac{|D_{q,t}|}{|D_q|}. \tag{5}$$

D_q is the set of documents that are relevant to the query. $D_{q,t}$ is the subset of relevant documents that contains term t. Their ratio, $TR(t, q)$, is the term recall weight for term t in query q. Term recall is in the range of [0, 1]. Term recall is based on the assumption that a query term is more important if it is mentioned by more relevant documents.

Training requires relevant query-document pairs. The model takes the text content of a query, makes predictions, and computes the loss with target weights generated from the relevant documents. During inference, the model only needs the query.

Re-weight queries with predicted term weights. When a query is received (*online*), the trained model is used to predict importance weights for each term. Following DeepTR [36], we use the estimated query term weights to generate bag-of-words queries (BOW) and sequential dependency model queries (SDM). For example, in the Indri [31] query language, the original BOW query "apple pie" is re-formulated into #weight(0.8 apple 0.7 pie)[3] for predicted weights apple:0.8, pie:0.7. Sequential dependency model adds bigrams and word co-occurrences within a window to the query. We use the re-weighted BOW query to replace the bag-of-words part of the SDM query. Terms with non-positive weights are

discarded. In terms of efficiency, predicting term weights for a new query is simply feeding-forward the query string through DeepCT. We use Bow-DeepCT-Query and SDM-DeepCT-Query to denote the re-weighted bag-of-words and sequential dependency queries.

4 EXPERIMENTAL METHODOLOGY FOR DeepCT-Index

This section presents the experimental methodology for the second task – passage term weighting and indexing.

Datasets: The current implementation of BERT supports texts of up to 512 tokens, thus we selected two collections that consist primarily of passages: **MS MARCO** [21] and **TREC-CAR** [12]. Passages and short documents are also the case where weighting by term frequency is less effective, because the *tf* distribution tends to be flat. The experimental settings mainly follow the methodology used in previous neural passage ranking/re-ranking studies [22, 23]

MS MARCO [21] is a question-to-passage retrieval dataset with 8.8M passages. Average passage length is around 55 words. The training set contains approximately 0.5M pairs of queries and relevant passages. On average each query has one relevant passage. The development (dev) set contains 6,980 queries and their relevance labels. The test set contains 6,900 queries, but the relevance labels are hidden by Microsoft. Therefore, *the dev set is our main evaluation set.* In a few experiments, we also evaluated on the test set by submitting our rankings to the MS MARCO competition.

TREC-CAR [12] consists of 29.7M English Wikipedia passages with an average length of 61 words. Queries and relevant passages are generated synthetically. A query is the concatenation of a Wikipedia article title with the title of one of its sections. Following prior work [22, 23], we use the automatic relevance judgments, which treats paragraphs within the section as relevant to the query. The training set and validation set have 3.3M query-passage pairs and 0.8M pairs respectively. The test query set contains 1,860 queries with an average of 2.5 relevant paragraphs per query.

Baselines: Experiments were done with done with three baseline and three experimental indexing methods, as described below.

tf index is a standard *tf*-based index, e.g., as used by BM25.

TextRank [18] is a widely-used unsupervised graph-based term weighting approach. We use the open source PyTextRank implementation[4]. Term weights from TextRank are in the range (0, 1); we scale them to integers as described in Eq. 4 for indexing.

Doc2Query [23] is a supervised neural baseline. It trains a neural sequence-to-sequence model to generate potential queries from passages, and indexes the queries as document expansion terms. Doc2Query implicitly re-weights terms because important passage terms are likely to appear in the generated queries. We use the Doc2Query MS MARCO index released by the authors. No such index is available for TREC-CAR, so we use published values for that dataset [23]. Doc2Query was the best-performing first-stage ranking method on MS MARCO at the time this paper was written.

The three experimental indexing methods include the proposed DeepCT-Index and two variants using different embeddings.

DeepCT_W-Index replaces the BERT component in DeepCT with context-independent word embeddings. To provide context to each word, we modeled the passage using the average word embeddings

---

[3]#weight is Indri's probabilistic weighted-AND operator.

[4]https://github.com/DerwenAI/pytextrank

and subtracted the passage embedding from each word's embedding, which was inspired by [36]. The embeddings are initialized by word2vec [19] and fine-tuned during training.

DeepCT$_E$-Index replaces the BERT component in DeepCT with ELMo [25]. ELMo is a pre-trained, context-aware text representation model that differs from BERT in the network architecture and pre-training task. ELMo was initialized with the pre-trained model described by Peters et al. [25] and fine-tuned during training.

Indexing, Retrieval, and Evaluation: First-stage ranking was done by two popular retrieval models: BM25 and query likelihood with Jelinek-Mercer smoothing (QL). We used the Anserini toolkit implementations. We fine-tuned BM25 parameters k_1 and b, and QL smoothing factor λ through a parameter sweep on 500 queries from the training set. Re-ranking was done by two re-rankers: Conv-KNRM [9] and a BERT Re-ranker [22]. We used the trained re-rankers provided by the authors, and applied them to re-rank up to a 1000 passages from a first-stage ranking.

Ranking/re-ranking results were evaluated by Mean Reciprocal Rank at 10 passages (MRR@10), the official MS MARCO evaluation metric. For TREC-CAR, we also report MAP at depth 1,000 following the evaluation methodology used in previous work [22, 23].

DeepCT Settings: The BERT part of DeepCT-Index was initialized with pre-trained parameters. For MS MARCO, we used the official pre-trained BERT (uncased, base model) [11]. TREC-CAR cannot use the official model because its testing documents overlapped with BERT's pre-training documents. We used a BERT model from Nogueira and Cho [22] where the overlapping documents are removed from the pre-training data. After initialization, DeepCT is trained for 3 epochs on the training split of our datasets, using a learning rate of $2e^{-5}$ and a max input text length of 128 tokens.

5 DeepCT-Index RESULTS

The next two subsections describe experiments that investigated first-stage search accuracy using DeepCT-Index indexes, and why DeepCTIndex term weights are effective.

5.1 Retrieval Accuracy of DeepCT-Index

This section examines whether DeepCT-Index improves first-stage retrieval accuracy over baseline term weighting methods. Then it compares the first-stage retrieval accuracy of DeepCT-Index to several single-/multi-stage competition search systems. Finally, it studies the impact of a first-stage ranker using DeepCT-Index on the end-to-end accuracy of multi-stage search systems.

DeepCT-Index Retrieval Performance. Table 2 shows the first-stage retrieval accuracy of BM25 and QL using indexes generated by six methods. The TextRank index failed to beat the common tf index. The Doc2Query index was effective for MS MARCO, but only marginally better for TREC-CAR.

DeepCT-Index outperformed the baselines by large margins. It improved BM25 by 27% on MS MARCO and 46% on TREC-CAR. It produced similar gains for QL, showing that DeepCT-Index is useful to different retrieval models. Win/Loss analysis shows that DeepCT-Index improved 25-35% of the queries and hurt 8-16%.

DeepCT-Index also surpassed the neural baseline Doc2Query by large margins. DeepCT-Index and Doc2Query were trained using the same data, demonstrating the advantage of DeepCT-Index's

explicit term-weighting over Doc2Query's implicit term-weighting through passage expansion.

Results for DeepCT$_W$-Index and DeepCT$_E$-Index demonstrate the importance of context. Non-contextual word2vec embeddings produced term weights that were less effective than *tf*. ELMo produced more effective term weights, but BERT's stronger use of context produced the most effective weights. These results also show the generality of the DeepCT framework.

First-stage search with DeepCT-Index vs. re-ranking. We further compared the DeepCT-Index-based BM25 retrieval to several competition systems by participating in the MS MARCO competition. Table 3 shows results from the MS MARCO leaderboard[6]. It first lists 2 important first-stage rankers: the official standard BM25 and Doc2Query BM25 [23]. DeepCT-Index BM25 outperformed both. Table 3 also lists representative re-ranking approaches for feature-based learning-to-rank, previous state-of-the-art neural re-rankers (*non-ensemble version*), and BERT-based re-rankers. FastText+Conv-KNRM [14] was the best non-BERT model on the leaderboard. BERT Re-Ranker [22] uses BERT as a black-box model that takes a query-passage pair and outputs a relevance scores; most recently-proposed re-rankers in the BERT family [8, 22] are based on this approach. All the multi-stage systems used the Official BM25 for first-stage ranking, and applied a re-ranker to fine-tune the rankings of the top 1000 passages.

DeepCT-Index BM25 was better than several re-ranking pipelines. It is more accurate than feature-based learning-to-rank, a widely used re-ranking approach in modern search engines. It is also more accurate than a popular neural re-ranking model K-NRM [33]. Compared to Duet V2 (the official re-ranking baseline) and Conv-KNRM [9], DeepCT-Index BM25 achieves similar accuracy while being more efficient as it does not need the re-ranking stage. These results indicate that it is possible to replace some pipelined ranking systems with a single-stage retrieval using DeepCT-Index.

Finally, strong neural re-rankers like the BERT Re-Ranker were much more effective than DeepCT-Index BM25. These models generate high quality soft-match signals between query and passage words (e.g. "hotel" to "motel"). In contrast, DeepCT-Index BM25 only matches terms exactly, which provides much less evidence.

DeepCT-Index as the first-stage of re-ranking systems. The next experiment examines whether a first-stage ranking produced by DeepCT-Index BM25 can improve later-stage re-rankers. Table 4 reports the performance of two re-rankers applied to candidate passages retrieved from DeepCT-Index and the tf index. The re-rankers were selected based on their performance on the MS-MARCO leaderboard (Table 4): Conv-KNRM [9], which has medium accuracy, and BERT Re-Ranker [22], which has high accuracy. We tested various re-ranking depths. Re-ranking at a shallower depth has higher efficiency but may miss more relevant passages.

The recall values show the percentage of relevant passages in the re-ranking passage set. DeepCT-Index had higher recall at all

Table 2: Ranking accuracy of `BM25` and `QL` using indexes built with three baselines and three `DeepCT-Index` methods. Win/Tie/Loss are the number of queries improved, unchanged, or hurt, compared to `tf` index on MRR@10. * and † indicates statistically significant improvements over `tf` index and `Doc2Query`. `Doc2Query` results for TREC-CAR are from Nogueira et al. [23]; statistically significance for these results are unknown.

| Index | MS MARCO dev | | | | TREC-CAR | | | | | |
| | BM25 | | QL | | BM25 | | | QL | | |
	MRR@10	W/T/L	MRR@10	W/T/L	MRR@10	MAP	W/T/L	MRR@10	MAP	W/T/L
`tf index`	0.191	-/-/-	0.189	-/-/-	0.233	0.174	-/-/-	0.211	0.162	-/-/-
`TextRank`	0.130	662/4556/1762	0.134	702/4551/1727	0.160	0.120	167/1252/441	0.157	0.118	166/1327/367
`Doc2Query`	0.221*	1523/4431/1026	0.224*	1603/4420/957	-	0.178	-/-/-	-	-	-/-/-
`DeepCT-Index`	**0.243***†	2022/3861/1097	**0.230***	1843/4027/1110	**0.332***	**0.246***	615/1035/210	**0.330***	**0.247***	645/1071/144
`DeepCT`$_W$`-Index`	0.174	931/4804/1245	0.168	867/4793/1320	0.205	0.147	250/1311/309	0.192	0.139	245/1345/280
`DeepCT`$_E$`-Index`	0.234*†	1891/4139/950	0.220*	1726/4210/1044	0.280*	0.201*	516/1144/200	0.276*	0.197*	540/1190/130

Table 3: Retrieval accuracy of BM25 with `DeepCT-Index` compared with several representative ranking/re-ranking systems on the MS MARCO dataset using official evaluation on dev and hidden test set.[5]

| | Ranking Method | dev | | test | |
		MRR@10		MRR@10	
Single-Stage	Official BM25	0.167	-30%	0.165	-31%
	Doc2Query BM25 [23]	0.215	-12%	0.218	-9%
	DeepCT-Index BM25	0.243	–	0.239	–
Multi-Stage	Feature-based LeToR	0.195	-20%	0.191	-20%
	K-NRM [33]	0.218	-10%	0.198	-17%
	Duet V2 [20]	0.243	+0%	0.245	+2%
	Conv-KNRM [9]	0.247	+2%	0.247	+3%
	FastText+Conv-KNRM [14]	0.277	+14%	0.290	+21%
	BERT Re-Ranker [22]	0.365	+50%	0.359	+50%

Table 4: Re-Ranking accuracy of two re-rankers applied to passages retrieved by BM25 from `DeepCT-Index` and the `tf index`. Dataset: MS MARCO dev.

| | Recall | | Conv-KNRM Re-Ranker MRR@10 | | BERT Re-Ranker MRR@10 | |
Depth	tf	DeepCT	tf	DeepCT	tf	DeepCT
10	40%	49%	0.234	0.270	0.279	0.320
20	49%	58%	0.244	0.277	0.309	0.343
50	60%	69%	0.253	0.278	0.336	0.361
100	68%	76%	0.256	0.274	0.349	0.368
200	75%	82%	0.256	0.269	0.358	0.370
500	82%	88%	0.256	0.269	0.366	0.374
1000	86%	91%	0.256 [1]	0.264	0.371 [1]	0.376

[1] The values are not exactly the same as in Table 4 due to differences in the initial rankings generated from our BM25 and the official BM25 from MS MSARCO.

depths, meaning a ranking from `DeepCT-Index` provided more relevant passages to a re-ranker. Both re-rankers consistently achieved higher MRR@10 by using `DeepCT-Index` compared to using `tf` index. For Conv-KNRM, the best MRR@10 improved from 0.256 to 0.278, and the required re-ranking depth decreased from 100 to 50. In other words, a first-stage ranking from `DeepCT-Index` helped the re-ranker to be over 8% more accurate and 2× more efficient. For BERT re-ranker, `DeepCT-Index` enabled the re-ranker to achieve similar accuracy using much fewer passages. Re-ranking the top 100-200 passages from `DeepCT-Index` produced similar MRR@10 as re-ranking the top 1,000 passages from `tf` index.

The high computational cost of deep neural-based re-rankers is one of the biggest concerns about adopting them in online services. Nogueira et al. [23] reported that adding a BERT Re-Ranker, with a re-ranking depth of 1000, introduces 10× more latency to a BM25 first-stage ranking even using GPUs or TPUs. `DeepCT-Index` reduces the re-ranking depth by 5× to 10×, making deep neural-based re-rankers practical in latency-/resource-sensitive systems.

Summary. There is much prior research about passage term weighting, but it has not been clear how to effectively model a word's syntax and semantics in specific passages. Our results show that a deep, contextualized neural language model is able to capture some of the desired properties, and can be used to generate effective term weights for passage indexing. A BM25 retrieval on `DeepCT-Index` can be 25% more accurate than classic *tf*-based indexes, and are more accurate than some widely-used multi-stage retrieval systems. The improved first stage ranking further benefits the effectivenss and efficiency of downstream re-rankers.

5.2 Understanding Sources of Effectiveness

This section aims to understand the sources of effectiveness of `DeepCT-Index` through several analyses.

Table 5 visualizes `DeepCT-Index` weights on a few cases. Each case has a query, a relevant passage, and a non-relevant passage that mentions query concepts but is actually off-topic. The heatmap visualizes the term weights in `DeepCT-Index`. Weights are normalized by the sum of all term weights in the passage, to reflect relative term importance in the passage.

Emphasize central terms and suppress non-central terms. DeepCT is able to identify terms that are central to the topic of the text. In the first query in Table 5, both passages mention *"susan boyle"*. In the relevant passage, `DeepCT-Index` recognized that the topic is *"susan boyle"* and put almost all weight on these two terms. The off-topic passage is about the definition of *"troll"*, while *"susan boyle"* is used in an example. `DeepCT-Index` managed to identify the central concept *"troll"* and suppress other non-topical terms; less than 10% of weight goes to *"susan boyle"*. With the new weights, the

Table 5: Visualization of `DeepCT-Index` passage term weights. Red shades reflect the normalized term weights – the percentage of total passage term weights applied to the term. White indicates zero weight. Query terms are bold.

	Percentage of weights a term takes in the passage: 0 10% 20% 30% 40% >50%
Query	**who is susan boyle**
On-Topic	Amateur vocalist **Susan Boyle** became an overnight sensation after appearing on the first round of 2009's popular U.K. reality show Britain's Got Talent.
Off-Topic	Best Answer: a troll is generally someone who tries to get attention by posting things everyone will disagree, like going to a **susan boyle** fan page and writing **susan boyle** is ugly on the wall. they are usually 14-16 year olds who crave attention.
Query	**what values do zoos serve**
On-Topic	**Zoos serve** several purposes depending on who you ask. 1) Park/Garden: Some **zoos** are similar to a botanical garden or city park. They give people living in crowded, noisy cities a place to walk through a beautiful, well maintained outdoor area. The animal exhibits create interesting scenery and make for a fun excursion.
Off-topic	There are NO purebred Bengal tigers in the U.S. The only purebred tigers in the U.S. are in AZA **zoos** and include 133 Amur (AKA Siberian), 73 Sumatran and 50 Malayan tigers in the Species Survival Plan. All other U.S. captive tigers are inbred and cross bred and do not **serve** any conservation **value**.
Query	**do atoms make up dna**
On-Topic	**DNA** only has 5 different **atoms** - carbon, hydrogen, oxygen, nitrogen and phosphorous. According to one estimation, there are about 204 billion **atoms** in each **DNA**.
Off-Topic	Genomics in Theory and Practice. What is Genomics. Genomics is a study of the genomes of organisms. It main task is to determine the entire sequence of **DNA** or the composition of the **atoms** that make up the **DNA** and the chemical bonds between the **DNA atoms**.

BM25 score between the query and the off-topic passage is greatly decreased. Similar behavior can be seen on other cases in Table 5.

Figure 1 compares the term weight distribution of `DeepCT-Index` and `tf` index. It plots the average weight of each passage's highest-weighted term, the average weight of the second highest-weighted term, and so on. The original *tf* distribution is flat. `DeepCT-Index` assigns high weights to a few central terms, resulting in a skewed term weight distribution. Such skewed distribution confirms our observations from the case study that `DeepCT-Index` aggressively emphasizes a few central terms and supresses the others.

DeepCT's strong bias towards central terms explains its effectiveness: promoting on-topic terms and suppressing off-topic terms. It is also a source of *error*. In many failing cases, `DeepCT-Index` identified the central passage words correctly, but the answer to the query is mentioned in a non-central part of the passage. `DeepCT-Index` down-weighted or even ignored these parts, causing relevant passage to be ranked lower. It is worth exploring how to reduce the risk of losing information that is useful but not central.

Context-based weights vs. frequency-based weights. The examples in Table 5 also show that the term weights produced by `DeepCT-Index` are based on the meaning of the context rather than frequency. A term may get low weight even it is frequent (e.g. in the last *"Genomics"* passage, *"DNA"* is considered unimportant even though it is mentioned 3 times). The same term receives very different weights in different passage even when *tf* is the same. This extent of independence from frequency signals is uncommon in previous term weighting approaches; even semantic text representations such as Word2Vec [19] and Glove [24] are reported to have high correlation with term frequency [30].

6 EXPERIMENTAL METHODOLOGY AND RESULTS FOR DeepCT-Query

This section presents the experimental methodology and results for the query term weighting task. The methodolgy follows the settings used in [36].
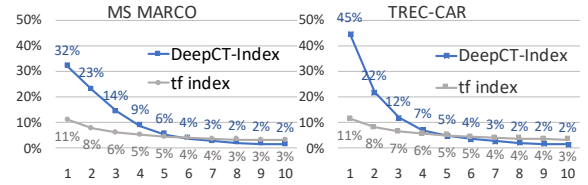

Figure 1: Term weight distribution of the top-10 terms in passages with highest weights. The X-axis shows the term's rank ordered by weight. The Y-axis shows the average term weight normalized by total passage term weight.

Datasets: We used the **Robust04** and **Gov2** TREC collections. Robust04 is a news corpus with 0.5M documents and 249 test topics. Gov2 is a web collection with 25M web pages and 150 test topics. Each test topic has 3 types of query: a short title query, a sentence-long query description, and a passage-long query narrative.

Baselines: Experiments were done with 6 baselines that use two forms of query structure (BOW, SDM) and three types of term weights (`tf`, DeepTR, *Oracle*). BOW and SDM stand for bag-of-word queries and sequential dependency queries [17]. `tf` is the classic query term frequency weights. DeepTR is a previous state-of-the-art query weighting approach [36]. It extracts word features using the difference between the word's own embedding and the average embedding of the query. The features are linearly combined to produce term weights.DeepTR is supervised trained on term recall (Eq. 3). *Oracle* weights query terms by the ground truth term recall weights; it reflects how much DeepTR and `DeepCT-Query` can achieve if they made perfect predictions, estimating an upper limit.

Indexing, Retrieval and Evaluation: Following Zheng and Callan [36], we use the Indri search engine[7] with standard stemming and stop words filtering. We train and evaluate `DeepCT-Query` and DeepTR with 5-fold cross validation. Zheng and Callan [36] show

---

[7]http://lemurproject.org/indri/

Table 6: Retrieval accuracy of `DeepCT-Query` using `QL` retrieval model on Robust04 and Gov2. * indicates statistically significant improvements of `BOW-DeepCT-Query` over `BOW-DeepTR`. † indicates statistically significant improvements of `SDM-DeepCT-Query` over `SDM-DeepTR`. BOW-/SDM-*Oracle* weight terms by ground truth, estimating an upper limit for `DeepTR` and `DeepCT-Query`.

Query	Robust04						Gov2					
	Title		Description		Narrative		Title		Description		Narrative	
	NDCG@20	MAP	NDCG@20	MAP	NDCG@20	MAP	NDCG@20	MAP	NDCG@20	MAP	NDCG@20	MAP
BOW-tf	**0.410**	**0.243**	0.417	0.248	0.320	0.177	0.442	0.291	0.407	0.253	0.419	0.231
BOW-DeepTR	0.381	0.226	0.427	0.264	0.330	0.183	0.437	0.289	0.427	0.271	0.395	0.211
BOW-DeepCT-Query	0.383	0.229	**0.445***	**0.273***	**0.367***	**0.213***	**0.443**	**0.293**	**0.430**	**0.280**	**0.438***	**0.260***
BOW-*Oracle*	*0.369*	*0.228*	*0.484*	*0.311*	*0.447*	*0.283*	*0.453*	*0.306*	*0.462*	*0.312*	*0.481*	*0.298*
SDM-tf	**0.427**	**0.264**	0.427	0.261	0.332	0.186	**0.483**	**0.324**	0.434	0.270	0.436	0.239
SDM-DeepTR	0.394	0.241	0.452	0.261	0.355	0.186	0.482	0.324	**0.464**	**0.294**	0.432	0.237
SDM-DeepCT-Query	0.394	0.242	**0.462**	**0.288**†	**0.380**†	**0.225**†	0.483	0.323	0.446	0.292	**0.455**†	**0.268**†
SDM-*Oracle*	*0.398*	*0.248*	*0.453*	*0.295*	*0.441*	*0.276*	*0.492*	*0.337*	*0.472*	*0.319*	*0.496*	*0.305*

that query likelihood (QL) model performs slightly better than BM25, so we use QL to search the index. Retrieval results are evaluated using standard TREC metrics: NDCG@20 and MAP@1000.

DeepCT Settings: The BERT part of DeepCT was initialized with the official pre-trained BERT (uncased, base model) [11]. DeepCT, including the BERT layers and the last regression layer, was fine-tuned end-to-end. The model was trained for 10 epochs. We used a learning rate of $2e^{-5}$. Max input text length was set to be 30, 50 and 100 tokens for query titles, descriptions, and narratives.

Results. Results are listed in Table 6. The short title queries did not benefit from the term weighting approaches. Title queries often consist of a few keywords that are all essential, so re-weighting is less important. Besides, there isn't much context for DeepCT to leverage to estimate term importance. External information about the query, such as pseudo-relevance feedback [16], may be necessary to understand short queries.

Description and narrative queries mention many terms and concepts; it is important to identify which are central during retrieval. Weighted queries are more effective than the un-weighted queries. DeepCT-Query is more accurate than DeepTR in most cases. DeepCT-Query differs from DeepTR in how they represent a term with respect to the context. The results demonstrate that DeepCT-Query better reflect a word's role in the query. Larger improvements were observed on narrative queries than on description queries. The results indicate that for short sentences, simple context modeling may be effective. But for more complex queries, a deep language modeling component like BERT can lead to improved search results.

7 CONCLUSION

Recently much research has focused on the later stages of multi-stage search engines. Most first-stage rankers are older-but-efficient bag-of-words retrieval models that use term frequency signals. However, frequency-based term weighting does not necessarily reflect a term's importance in queries and documents, especially when the frequency distribution is flat, such as in sentence-long queries or passage-long documents. More accurate term importance estimates require the system to understand the role each word plays in each specific context. This paper presents DeepCT, a novel context-aware term weighting approach that better estimates term importance for first-stage bag-of-words retrieval systems.

The DeepCT framework is built on BERT [11], a recent deep neural language model. In BERT, a word's embedding gradually 'absorbs' information from other related words in the input text and generates a new word embedding that characterizes the word in a specific context. DeepCT uses BERT to extract context-based term features and learns to use these features to predict the importance for each term in a supervised per-token regression task. The training signals are mined from relevance-based query-document pairs so that the predicted term weights are aligned with the retrieval task.

One use of DeepCT is DeepCT-Index, which weights document terms. DeepCT produces integer term weights that can be stored in a typical inverted index and are compatible with popular first-stage retrieval models such as BM25 and query likelihood. Another use of DeepCT is DeepCT-Query, which weights query terms. Experimental results show that DeepCT-Query greatly improves the accuracy of longer queries, due to its ability to identify central query terms in a complex context.

Experimental results show that DeepCT-Index improves the accuracy of two popular first-stage retrieval algorithms by up to 50%. Running BM25 on DeepCT-Index can be as effective as several previous state-of-the-art multi-stage search systems that use knowledge bases, machine learning, and large amounts of training data.

The higher-quality ranking enabled by DeepCT-Index improves the accuracy/efficiency tradeoff for later-stage re-rankers. A state-of-the-art BERT-based re-ranker achieved similar accuracy with 5× fewer candidate documents, making such computation-intensive re-rankers more practical in latency-/resource-sensitive systems.

Although much progress has been made toward developing better neural ranking models for IR, computational complexity often limits these models to the re-ranking stage. DeepCT successfully transfers the text understanding ability from a deep neural network into simple signals that can be efficiently consumed by early-stage ranking systems and boost their performance.

Analysis shows the main advantage of DeepCT over classic term-weighting approaches: DeepCT finds the most central words in a text even if they are mentioned only once. Non-central words, even if mentioned frequently in the text, are suppressed. Such behavior is uncommon in previous term weighting approaches. We view DeepCT as an encouraging step from "frequencies" to " meanings".

REFERENCES

[1] Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Francesco Silvestri. 2018. Distance-Sensitive Hashing. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems.*

[2] Michael Bendersky, Donald Metzler, and W Bruce Croft. 2010. Learning concept importance using a weighted dependence model. In *Proceedings of the Third International Conference on Web Search and Web Data Mining.*

[3] Michael Bendersky, Donald Metzler, and W Bruce Croft. 2011. Parameterized concept weighting in verbose queries. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[4] Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2012. Effective query formulation with multiple information sources. In *Proceedings of the Fifth International Conference on Web Search and Web Data Mining.*

[5] Roi Blanco and Christina Lioma. 2012. Graph-based term weighting for information retrieval. *Information retrieval* (2012).

[6] Leonid Boytsov, David Novak, Yury Malkov, and Eric Nyberg. 2016. Off the Beaten Path: Let's Replace Term-Based Retrieval with k-NN Search. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management.*

[7] David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, Yoelle S Maarek, and Aya Soffer. 2001. Static index pruning for information retrieval systems. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[8] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[9] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining.*

[10] 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.*

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

[12] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. 2017. TREC complex answer retrieval overview. In *Proceedings of the Twenty-Seventh Text REtrieval Conference.*

[13] 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 Conference on Information and Knowledge Management.*

[14] Sebastian Hofstätter, Navid Rekabsaz, Carsten Eickhoff, and Allan Hanbury. 2019. On the Effect of Low-Frequency Terms on Neural-IR Models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[15] Victor Lavrenko and W. Bruce Croft. 2001. Relevance-Based Language Models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[16] Victor Lavrenko and W Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 120–127.

[17] Donald Metzler and W. Bruce Croft. 2005. A Markov random field model for term dependencies. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[18] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing.*

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems.*

[20] Bhaskar Mitra and Nick Craswell. 2019. An Updated Duet Model for Passage Re-ranking. *arXiv preprint arXiv:1903.07666* (2019).

[21] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).

[22] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv:1901.04085* (2019).

[23] Rodrigo Nogueira, Kyunghyun Cho, Yang Wei, Lin Jimmy, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *arXiv:1904.08375* (2019).

[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing.*

[25] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

[26] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* (2009).

[27] François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and TW-IDF: new approach to ad hoc IR. In *22nd ACM International Conference on Information and Knowledge Management.*

[28] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.

[29] Gerard Salton and Michael McGill. 1984. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company.

[30] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.*

[31] Trevor Strohman, Donald Metzler, Howard Turtle, and W Croft. 2005. Indri: A language-model based search engine for complex queries. *Information Retrieval - IR* (2005).

[32] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316* (2019).

[33] 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.*

[34] Hamed Zamani, Mostafa Dehghani, W Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management.*

[35] Le Zhao and Jamie Callan. 2010. Term necessity prediction. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010.*

[36] Guoqing Zheng and Jamie Callan. 2015. Learning to Reweight Terms with Distributed Representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.*