Context-Aware Document Term Weighting for Ad-Hoc Search

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

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

ABSTRACT

Bag-of-words document representations play a fundamental role in modern search engines, but their power is limited by the shallow frequency-based term weighting scheme. This paper proposes HDCT, a context-aware document term weighting framework for document indexing and retrieval. It first estimates the semantic importance of a term in the context of each passage. These fine-grained term weights are then aggregated into a document-level bag-of-words representation, which can be stored into a standard inverted index for efficient retrieval. This paper also proposes two approaches that enable training HDCT without relevance labels. Experiments show that an index using HDCT weights significantly improved the retrieval accuracy compared to typical term-frequency and state-of-the-art embedding-based indexes.

KEYWORDS

Document Representation, Term Weighting, Neural IR

ACM Reference Format:
Zhuyun Dai and Jamie Callan. 2020. Context-Aware Document Term Weighting for Ad-Hoc Search. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380258

1 INTRODUCTION

The bag-of-words plays a fundamental role in modern search engines due to its efficiency and ability to produce detailed term matching signals. Most bag-of-words representations and retrieval models use term weights based on term frequency (*tf*), for example *tf.idf* and BM25 [32]. However, being *frequent* does not necessarily lead to being *semantically important*. Identifying central words in a text also requires considering the meaning of each word and the role it plays in a specific context.

Recently, there has been rapid progress in text understanding with the introduction of deep contextual word representations such as ELMo [30] and BERT [36]. These methods assign to each word a representation that is a function of the entire text. They were shown to capture the characteristics of a word's semantics and syntax, and more importantly, how they vary across linguistic contexts [37].

This paper proposes a novel framework that uses the contextual word representations from BERT [36] to generate more effective document term weights for bag-of-words retrieval. We present HDCT, a **C**ontext-aware **H**ierarchical **D**ocument **T**erm weighting framework. HDCT first estimates a term's context-specific importance in each passage, by projecting each word's BERT representation into a real-valued term weight. In this way, the term weight estimation can take into account word orders, dependencies, and complex interactions within the local context. Estimating at the passage level also allows HDCT to handle long documents that exceed BERT's length limit [14]. Next, HDCT combines the local passage term weights into a global document bag-of-words representation. The document representation provides the retrieval model with both document-level and passage-level key terms, so that documents can be matched to queries accurately. These representations can be generated offline, stored in an inverted index, and retrieved efficiently with standard bag-of-words retrieval algorithms such as BM25.

Training HDCT requires having ground truth information about a term's importance in a passage. It is impractical to manually label every term in every training document. This paper proposes three strategies that automatically generate training labels using documents, relevance feedback, and pseudo-relevance feedback.

The first strategy solely relies on document contents. An ideal system should be able to automatically build a search engine for any document collection without human labeling effort. Towards this goal, we propose a *content-based weak-supervision* strategy that exploits the internal structure of documents. It mines labels from certain document fields that were shown to provide a high-quality summary of the document (e.g., titles [19] and web inlinks [15]), helping HDCT to recognize essential terms in a passage.

The second and the third strategies leverage search logs and relevance feedback to align HDCT with user search intents. When relevance signals are available, it is intuitive to bias document term weights towards their related queries [10]. We propose a *relevance-based supervision* strategy that trains HDCT using term distribution in a document's relevant queries. Sometimes user queries are available but not relevance signals, for example, if privacy regulations do not permit collecting user clicks. We propose a *PRF-based weak-supervision* strategy that trains HDCT on machine-generated pseudo-relevant feedback (PRF) labels.

Experiments show that the content-trained HDCT significantly improves bag-of-words retrieval models like BM25 and RM3 [22], and can be competitive with some supervised learning-to-rank pipelines. An analysis shows that BERT-based term weights are more effective than term frequencies at the passage level. The hierarchical document modeling approach successfully combines the passage-specific term weights for document retrieval, outperforming other approaches that combine passage retrieval scores.

Section 2 reviews related work. Sections 3 and 4 describe the HDCT framework and the three strategies to generate training labels. Experimental methodologies and results are presented in Sections 5 and 6. Section 7 concludes the paper. Code, data, and hyperparameter study can be found in our virtual appendices. [1]

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380258

[1] http://boston.lti.cs.cmu.edu/appendices/TheWebConf2020-Zhuyun-Dai/

2 RELATED WORK

Document Representations in Ad-Hoc Search. Modern search engines were built upon *bag-of-words document representations*. One of the central problems in bag-of-words is how to quantify the context-specific importance of a term in a particular document. The most widely-used term weighting approaches are term frequency (*tf*) based methods, such as BM25 [32] and Query Likelihood [22].

One alternative is statistical machine translation methods [3]. They weight a document term using its probability of translating into a query term. Statistical machine translation models have not been used much because they are difficult to train due to data sparsity, and were not more effective than *tf*-based retrieval with pseudo-relevance feedback [22] in most situations.

Another alternative is graph-based methods [4, 26, 33]. They construct a graph for each document, where nodes represent terms and edges represent relationships among words. Terms are then weighted using graph ranking methods such as PageRank. Graph-based methods were shown effective for short text [4]; but only marginal improvements over standard BM25 were observed on standard document retrieval datasets [4, 33].

Contextualized neural language models [14, 30] use deep neural networks to capture how a word interacts with other words in its context [37]. Compared to statistical machine translation models and graph-based models, these neural models are easier to train and can model more complicated word relations. They open new possibilities for estimating context-specific term importance. Although much recent work has applied contextualized neural language models like BERT [14] to IR tasks, prior work mainly focused on using BERT as black-box re-ranker [9, 27, 28, 31]. Our previous work [8] preliminarily studies using BERT to weight terms for initial ranking, but it is limited to sentences and short passages. It is still an open question whether these contextualized neural language models can be leveraged to generate better bag-of-words representations for longer documents, which are commonly seen in ad-hoc search.

Besides bag-of-words, recent research also investigates *neural document representations* for ad-hoc search. Most of prior work uses dense text representations that enable a query to match every document to some degree, which makes them impractical for first-stage ranking in large-scale datasets [11, 13, 16, 18, 39]. Zamani et al. [42] proposed a different approach named SNRM. SNRM learns high dimensional but sparse embeddings in which queries and documents are represented by a set of "latent words", so that they can be searched with standard inverted index. SNRM was shown to outperform traditional bag-of-words retrieval and several neural ranking/re-ranking models [42]. One challenge faced by representing a document into a single embedding is that it discards the original words in the document, therefore it may lose accurate term matching signals, which are critical to text retrieval [16].

Passage-Level Evidence. In document retrieval, the most widely-used way to incorporate passage-level evidence is to combine passage scores, which estimates the relevance score between queries and individual passages and aggregate passage scores into a document score. A large number of methods along this line of research have been proposed in the past few decades [20, 21, 23, 34, 38]. A less common approach is to combine passage representations, which uses passage-level term statistics to build document representations, and performs document-level retrieval. For example, recent work from Catena et al. [5] models document *tf* using a weighted sum of term frequencies per passage based on position of the passage in the document. It is an open question how to go beyond these simple statistics and mine deeper signals from passages to better represent documents.

Weak Supervision for IR. IR research mainly focuses on two types of weak relevance signals: content-based signals [1–3, 19, 25] and pseudo-relevance feedback based signals [13, 41, 42]. Content-based approaches are motivated by the observation that the document content often exhibits some relevance relations between pieces of text. Research on this topic dates back at least 20 years when Jin et al. [19] used title-document pairs to train statistical translation models. Recently, MacAvaney et al. [25] revisited this topic to train a neural ranking model. Pseudo-relevance feedback (PRF) based approaches make use of the rankings from a search engine to generate pseudo-relevant labels [13, 41, 42]. One limitation of PRF-based approaches is that they rely on the availability of queries and the quality of the pseudo-relevance labels [25]. In general, recent research on both of the above two types of weak supervision focused on embeddings [41, 42] or neural ranking models [13, 25]. Their effectiveness in learning discrete bag-of-words document representations remains to be studied.

3 THE HDCT FRAMEWORK

This section presents the hierarchical document term weighting framework, HDCT, as shown in Figure 1(a).

Given a document d, HDCT estimates passage-level term weights using contextual term representations generated by BERT [36]. Next, HDCT combines the passage-level term weights into document-level term weights. The output is a document bag-of-words representation that can be stored in a standard inverted index and retrieved by common bag-of-words retrieval models like BM25 .

3.1 Passage-Level Term Weighting

The first step of HDCT estimates a term's importance in a passage. Unlike traditional term weighting methods that use term frequency signals, we aim to consider the specific meanings and roles of a term in the passage using BERT.

Given a document d, HDCT first splits it into a sequence of passages $P_d = \{p_1, ..., p_n\}$. The max input text length of BERT is 512 tokens after tokenization – about 300 to 400 English words before tokenization. Meanwhile, prior research shows that fixed-size passages of 200-300 words more are effective than natural passages [20]. Therefore, passages in HDCT consist of consecutive sentences that make up to about 300 words.

Next, HDCT estimates term importance in each passage. Figure 1(b) shows the details of this step. Given a passage p, HDCT generates contextual token embeddings using BERT, which transforms a token into a contextual embedding based on its attention to every other word in the passage. Tenney et al. [37] show that these embeddings can characterize a token's syntactic features (e.g., word dependencies) and semantic features (e.g., named entity labeling), which can help estimate a term's importance.

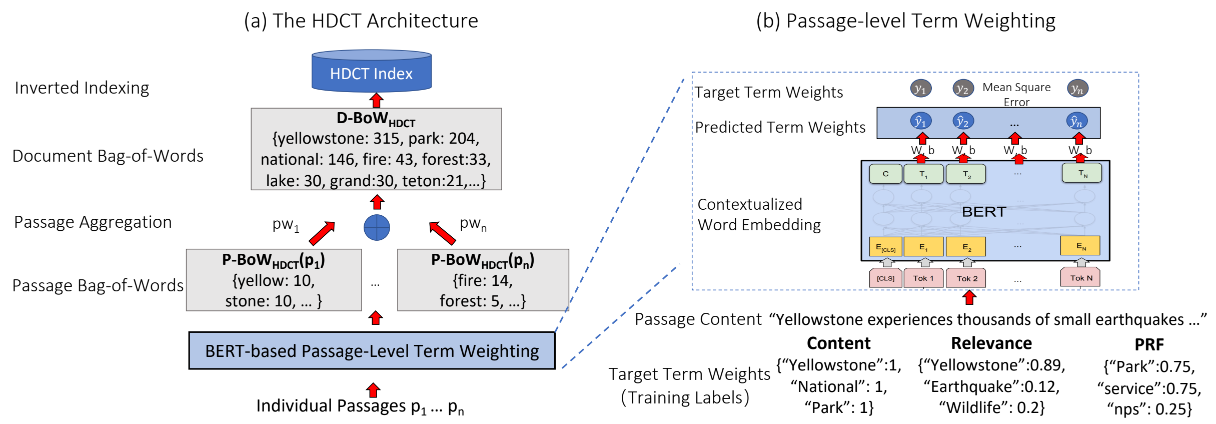
(a) The HDCT Architecture
(b) Passage-level Term Weighting
Inverted Indexing
HDCT Index
Document Bag-of-Words
D-BoW_HDCT {yellowstone: 315, park: 204, national: 146, fire: 43, forest:33, lake: 30, grand:30, teton:21,…}
Passage Aggregation
pw_1
pw_n
Passage Bag-of-Words
P-BoW_HDCT(p_1) {yellow: 10, stone: 10, … }
P-BoW_HDCT(p_n) {fire: 14, forest: 5, …}
BERT-based Passage-Level Term Weighting
Individual Passages p_1 … p_n
Target Term Weights
Predicted Term Weights
Mean Square Error
Contextualized Word Embedding
BERT
Passage Content "Yellowstone experiences thousands of small earthquakes …"
Target Term Weights (Training Labels)
Content {"Yellowstone":1, "National": 1, "Park": 1}
Relevance {"Yellowstone":0.89, "Earthquake":0.12, "Wildlife": 0.2}
PRF {"Park":0.75, "service":0.75, "nps": 0.25}

Figure 1: The HDCT architecture.

HDCT feeds these contextualized token embeddings into a linear layer. It maps a token's embedding into a real-number weight:

$$\hat{y}_{t,p} = w \cdot T_{\text{BERT}}(t,p) + b. \quad (1)$$

$T_{\text{BERT}}(t,p)$ is token t's contextualized embedding in passage p; w and b are the linear combination weights and bias; and, $\hat{y}_{t,p}$ is the predicted weight for token t in the passage p. $\hat{y}_{t,p}$ are mostly in the range of 0-1. This is because our training labels are in 0-1, so the model learns to generate predictions also in that range.

HDCT then scales the real-valued predictions into a tf-like integer that can be used with existing retrieval models. We call this weight tf_{BERT} to convey that it is an alternate way of representing the importance of term t in passage p using BERT:

$$tf_{\text{BERT}}(t,p) = round(N * \sqrt{\hat{y}_{t,p}}). \quad (2)$$

$\hat{y}_{t,p}$ is the prediction from Eq (1). N scales the prediction into an integer range, e.g. $N = 100$ keeps two digit precision. The square-root function is used for smoothing – it brings low predicted weights up, e.g. $\sqrt{0.01} = 0.1$, preventing the document representation from being dominated by a few highly-weighted terms. Two post-processing steps are applied: 1) To handle BERT's subwords, we use the weight of the first subwords for the entire word, and 2) When a term occurs multiple times in the passage, we take the maximum weight across the multiple occurrences.

After the scaling and post-processing, we generate a bag-of-words vector representation of the passage p:

$$\text{P-BoW}_{\text{HDCT}}(p) = [tf_{\text{BERT}}(t_1,p), .., tf_{\text{BERT}}(t_m,p)]. \quad (3)$$

Its terms are from the original text of the passage; the term weights are a tf-like integer based on predictions from BERT.

The above steps are applied to every passage $p_1, ..., p_n$ in the document d. At the end, HDCT generates a sequence of bag-of-words passage vectors.

$$\{\text{P-BoW}_{\text{HDCT}}(p_1), ..., \text{P-BoW}_{\text{HDCT}}(p_n)\}. \quad (4)$$

Although called bag-of-words, the term weights are based on the *linguistic context* of the passage, which is very different from traditional tf based bag-of-words. As shown in Table 1, HDCT emphasize terms that are topical to a passage, even if they have low tf (e.g. 'troll'); they also depress non-topical terms in the passage even if they have high tf (e.g. 'boyle').

Table 1: Visualization of an HDCT weighted passage. Deeper color represents higher weights.

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.

3.2 Document-Level Term Weighting

The previous step generates a sequence of passage bag-of-words representations. The next question is how to combine the passages for document retrieval.

A widely-used approach is to index and retrieve the passages independently, and aggregate passage scores at query time [20, 21, 23, 34, 38]. However, passage-level retrieval often faces the challenge of lacking document-level context [38].

HDCT uses a different approach that aggregates passage representations rather than passage scores. A term's importance in the document is a weighted sum of its importance in each passage:

$$\text{D-BoW}_{\text{HDCT}}(d) = \sum_{i=1}^{n} pw_i \times \text{P-BoW}_{\text{HDCT}}(p_i). \quad (5)$$

pw_i models the importance of the i-th passage p_i to the document d. This work explores two options for determining pw_i. The first one uses $pw_i = 1$ (sum); it weights all passages equally. The second one uses $pw_i = \frac{1}{i}$ (decay); it discounts passages based on the position, as prior research found that passages at the beginning of a document tend to attract more attention from readers and are more important for relevance estimation [5, 38]. Following [38], we use the reciprocal of position as the weight of a passage.

Besides passage position, it is also intuitive to weight passages from their content. This work does not explicitly model this factor. However, as will be discussing in the next section, we can train HDCT to down-weight all terms in a passage, hence implicitly weight passages based on content.

Finally, HDCT stores BoW_{HDCT}, the document bag-of-words representation, into an inverted index, where the new context-aware term weights replace the standard term frequency fields in the inverted lists [7]. We call it the HDCT index.

3.3 Retrieval with HDCT Index

To retrieve documents from HDCT indexes, we use the standard BM25 formula. The *tf* field in BM25 is replaced with the context-aware term weights stored in the HDCT index. HDCT is expected to improve retrieval by identifying key terms in a document.

This work also investigates whether HDCT index is compatible with pseudo-relevance feedback retrieval algorithms. Let $D_R = \{d_1, d_2, ..., d_k\}$ be the top k documents retrieved from a HDCT index in response to the query using standard BM25. An expanded query is generated using RM3 [22]: $P_{RM3}(t, q|R) = (1 - \lambda)P(t|q) + \lambda \sum_{d \in D_R} P(t|d)P(q|d)$, where $P(t|d)$ is estimated with the HDCT term weights:

$$P(t|d) = \frac{\text{D-BoW}_{\text{HDCT}}(d, t)}{\sum_{t_i} \text{D-BoW}_{\text{HDCT}}(d, t_i)}. \tag{6}$$

We than retrieve documents by running the expanded query against the HDCT index.

In terms of efficiency, HDCT does not introduce new words into documents, so the index does not become larger. Usually, the index can be smaller as some terms' weight becomes 0 during the scaling in Eq (2).

3.4 HDCT BoW vs. Classic BoW

Being bag-of-words (BoW) representation, HDCT shares the advantages of classic bag-of-words document representations: efficient retrieval, support for fine-grained term signals [16], and higher interpretability compared to latent topic models and embeddings. On the other hand, unlike traditional *tf* bag-of-words, HDCT does not assume term independence when estimating term weights. Building upon BERT's transformer architecture [36], HDCT takes into account word orders, dependencies, and complex interactions.

Moreover, HDCT investigates the potential of using the passage-level content understanding for document modeling. In most prior work, passages only provide limited signals for bag-of-words retrieval, such as passage term frequencies or passage positions [5, 20, 21, 23, 34, 38]. HDCT uses a deep neural network to extracts richer evidence from passages, leveraging the deep content understanding from passages to build the document representations.

4 TRAINING STRATAGIES FOR HDCT

This section first introduces a general training framework for HDCT. Then it proposes three strategies to generate training labels using documents, relevance feedback, and pseudo-relevance feedback.

HDCT needs to fine-tune the BERT parameters and learn the linear layer parameters w, b. These parameters are learned through a passage-level per-token regression task. As shown in Figure 1(b), assume we have the ground truth term weight for a term t in a passage p, denoted as $y_{t,p}$. We feed the passage p to HDCT, let HDCT predict term weights $\hat{y}_{t,p}$, and tries to minimizes the mean square error between the predictions $\hat{y}_{t,p}$ and the ground truth $y_{t,p}$:

$$MSE = \sum_{p} \sum_{t \in p} (y_{t,p} - \hat{y}_{t,p})^2. \tag{7}$$

However, it is impractical to manually label the importance of every term in every passage ($y_{t,p}$). To automatically generate labels, the key question is, *what evidence do we have that shows a term's importance for document retrieval?*. This paper proposes three training strategies: a content-based approach for cases where only the documents are available, a relevance-based approach for cases where rich query-document relevance assessments are assailable, and a pseudo-relevance based approach for cases where search queries can be collected but the relevance labels or user activities are not accessible.

4.1 Supervision from Document Content

A generally applicable search system should be able to build a good search engine just from the document collection. Towards this goal, the first training strategy mines labels from the documents themselves.

In many domains, the documents are loosely structured with various sources of textual information (fields), such as title, keywords, and inlinks (anchor text). Various researches have shown that these fields behave like real user queries [15, 19]. They provide a short summary of what a document is about and which search intents it may satisfy. These short, highly representative fields provide evidence about which terms bear high importance in the document.

Let F_d be a reference field that we use to train HDCT, e.g., the inlink field. We denote $F = \{f_1, ..., f_m\}$, where each element f_i is a text instance of the reference field. Some fields only have a single instance, e.g., a document usually has one title. Some fields may have multiple instances, e.g., a web page can have thousands of inlinks. The *content-based* strategy mines weak supervision signal about a term's importance by checking if, and how frequently, the term appears in the reference field.

Formally, given a training document d, its passages $\{p_1, ..., p_n\}$, and its reference field $F_d = \{f_1, ..., f_n\}$, the content-based weak-supervision approach generates labels as the following:

$$y_{t,p} = \frac{|F_{d,t}|}{|F_d|}, p \in \{p_1, ..., p_n\}, \tag{8}$$

where t is token from passage p, and $\frac{|F_{d,t}|}{|F_d|}$ is percentage of field instances that contain t. When there is a single instance, e.g., a document title, Eq (8) generates a binary label indicating if term t appears in the field or not. When there are multiple instance, e.g. inlinks, Eq 8 is a real number between 0 and 1. In the latter case, a token is considered more important if it is mentioned by a large portion field instances, reflecting the "collective wisdom".

As shown in Eq (8), the labels $y_{t,p}$ are actually passage independent and only depend on the document's reference field F_d. That means, if the document's title is "Yellowstone National Park", then the target term-weights for "yellowstone" will always be 1 regardless of which passage it appears in. Section 6.3 discusses the effects of such global labels in detail.

The training passage p and its target term weights derived from the reference field, are used to train HDCT by minimizing the MSE loss in Eq (7). These training labels are automatically extracted from the document content; no task-specific data collecting or labeling is required. It makes HDCT applicable to cold-start scenarios.

4.2 Supervision from Relevance and Pseudo-Relevance Feedback

When search queries and their relevant documents are available, they can provide rich information about people's search intents and interests. We would want the document term weights to align with patterns found in these search data. For example, "cast" should have high weight in movie-related documents as many search queries are looking for a movie's cast.

Given a training document d, its passages $P_d = \{p_1, ..., p_n\}$, and its relevant queries $Q_d = \{q_1, ..., q_b\}$, we generate the *relevance-based* training labels as follows:

$$y_{t,p} = \frac{|Q_{d,t}|}{|Q_d|}, p \in \{p_1, ..., p_n\}. \quad (9)$$

t is a term from passage p in document d. $\frac{|Q_{d,t}|}{|Q_d|}$ is the percentage of d's relevant queries that mention term t. If most of d's queries all mention t, then t is likely to be essential for this document. Same as the content-based supervision, the labels are global – they are based on a document's queries rather than passage-specific ones. We leave the discussion to Section 6.3.

In some cases, the search queries are available, but the relevance feed-backs such as clicks are not accessible (for example, in privacy-sensitive scenarios). Inspired by Zamani et al. [42], we propose a *pseudo-relevance feedback based (PRF-based)* weak-supervision strategy for HDCT, which collects pseudo-relevant documents for the queries instead of using true relevant documents.

It takes an existing retrieval system, e.g., BM25, to retrieve documents for the queries. For each query, the top K retrieved documents are considered to be pseudo-relevant to the query. It then collects a document's pseudo-relevant queries, PRF-Q_d, and generates the PRF-based training labels using the same way as Eq (9):

$$y_{t,p} = \frac{|\text{PRF-}Q_{d,t}|}{|\text{PRF-}Q_d|}, p \in \{p_1, ..., p_n\}. \quad (10)$$

The relevance-based and PRF-based labels relies on some existing relevance assessments or a query log. The labels take time and effort to obtain, but they are expected to improve the accuracy of HDCT by tailoring HDCT for the retrieval task.

4.3 Global Labels for Local Term Weighting

In the above three approaches, the target term weights (labels) are derived globally from the entire document, rather than being locally dependent on specific passages. One would expect these global labels to be less effective for passage term weighting. However, in practice, as the contextualized word representation for a word always varies with the passage, HDCT can still generate local term weights even though the training labels are global.

Moreover, these global labels let HDCT capture the global importance of passages. Some passages introduce noise, e.g., advertisements, navigation bars, or large blocks of equations. These passages do have their own locally important words, but they should not have high weights in the document. The document-derived labels teach HDCT to down-weight the entire passage. For example, low-quality passages often do not cover any inlink terms. Rather than trying to find the locally important words, HDCT gives 0 weight to all words in these passages. As a result, the entire passage makes little contribution to the document bag-of-words representation. We will illustrate the passage-weighting effect in Section 6.3.

5 EXPERIMENTAL METHODOLOGIES

This section presents our experimental methodologies, including datasets, baselines, and experimental methods.

5.1 Datasets

Experimental evaluation of HDCT used 4 document retrieval datasets with different characteristics.

ClueWeb09-B is a widely used text retrieval collection. The original collection contains 50 million web pages; we used the spam-filtered subset of 33 million documents. Spam was filtered using the Waterloo spam score [6] with a threshold of 60. The documents were split into a total of 100 million passages using a non-overlapping window of around 300 words. A document consists of 4 fields: title, URL, inlinks, and body.

ClueWeb09-C. Running HDCT over ClueWeb09-B is time consuming, making it slow to experiment with different model configurations. Therefore, we created ClueWeb09-C, a 10% subset of the original corpus. It consists of a 10% random sample of ClueWeb09-B documents, plus all documents that were in the original TREC judgment pool (in the qrels files)[2]. In total, there are 3.4 million documents and 10 million passages.

ClueWeb12-C. ClueWeb12-B13 is another standard text retrieval collection used in IR research. We created the 10% subset, called ClueWeb12-C, using the same method described above. Spam filter was not applied as suggested in [12]. In total, there are 5 million documents and 13 million passages. A document consists of four fields (title, URL, inlinks, and body).

The 2009-2014 TREC Web Tracks provided 200 queries with relevance assessments for ClueWeb09, and 100 queries for ClueWeb12. They were used for evaluating HDCT. Two versions of queries were evaluated: a short keyword query (title query) and a longer natural language query (description query). Evaluation used NDCG@20, the main metric for the TREC Web Tracks; MAP@1000, to show the effectiveness at deeper rankings; and MRR, to be consistent with MS-MARCO-Doc.

The **MS-MARCO Document Ranking dataset** (MS-MARCO-Doc)[3] is a benchmark dataset for web document retrieval recently released in the TREC 2019 Deep Learning Track. The dataset has 4 million documents, which produced 12 million passages. A document consists of 3 fields (title, URL, and body). The dataset comes with a training set of 0.37 million queries and the corresponding relevant documents. Evaluation was on the dev set, which contains 5193 queries[4]. Evaluation used the mean reciprocal rank (MRR) as suggested in the official instructions.

[2]If not included, many queries ended up with few or no relevant documents, making the evaluation results unstable.
[3]https://microsoft.github.io/TREC-2019-Deep-Learning/
[4]The relevance assessments for the official test set were not public at the time the paper was written.

5.2 Baselines and Experimental Methods

Our main baseline is *tf*, the standard term frequency based document index, e.g., as used by Luccene and Indri. We compare *tf* to several experimental HDCT methods.

On the ClueWeb datasets (ClueWeb09-B/C and ClueWeb12-C), we tested three experimental HDCT methods:

- HDCT-title was trained with the content-based weak supervision strategy, using titles as the reference field. It generated term weights for every document in the collection, and built an inverted index that were used for retrieval.
- HDCT-inlink was trained with the content-based weak supervised strategy, using inlinks as the reference field. We removed URL inlinks, and the most frequent inlinks like "next page" that has been linked to more than 10,000 documents in the collection.
- HDCT-PRFaol was trained with the PRF-based weak supervision strategy using the AOL query log [13, 25] and pseudo-relevance labels. We removed queries that are URLs and the 100 most frequent ones. We randomly sampled 500K unique queries, which is about the same scale as the MS-MARCO-Doc training query set used in HDCT-PRFmarco. It was also of the same scale as used in prior research [24]. 10 documents [25, 41] were retrieved for each query using BM25FE, a strong baseline that ensembles BM25 scores on each field (see **Retrieval Models** for more details); we sampled 1 from the top 10 documents to reduce computational cost.

Four HDCT methods were tested on MS-MARCO-Doc:

- HDCT-title was trained with document titles. MS-MARCO-Doc do not have inlink data, so HDCT-inlink is not available.
- HDCT-PRFaol was trained with the PRF-based strategy using the AOL query log and pseudo-relevance labels.
- HDCT-PRFmarco was trained with the PRF-based weak supervision strategy using the MS-MARCO-Doc training queries and pseudo-relevance labels. Same as the AOL query logs, we retrieved top 10 documents for each query using BM25FE and sample 1 to generate the training data. HDCT-PRFaol and HDCT-PRFmarco allows us to compare the differences between out-of-domain and in-domain queries.
- HDCT-supervised was a fully-supervised model trained with the relevance-based supervision. It used 0.37 million relevance assessments in the MS-MAROC-Doc training set.

All models were trained for 100K steps with a batch size of 16 and a learning rate of $2e-5$; training over 100K steps did not lead to significant improvements. BERT parameters was initialized with the official pre-trained BERT (uncased, base model) [36]. Max input length was set to 512 tokens. The scaling coefficient N in Eq (2) and the passage weights pw in Eq (5) were selected based on the dataset. We chose N from $\{10, 100\}$, and the passage weights from {sum, decay}. Unless otherwise specified, the rest of paper reports the best configuration of each dataset, that is $N = 10$ with sum for ClueWeb datasets, and $N = 100$ with decay for MS-MARCO-Doc.

Training HDCT took about one day on 4 TPUs. Indexing needs to run HDCT over the entire corpus, so the cost depends on the corpus size. It took HDCT less than 1 day $\times$ 4 TPUs to index ClueWeb09-C, ClueWeb12-C, and MS-MARCO-Doc (3-5 million documents, 10-13 million passages), and 6 days $\times$ 4 TPUs to index ClueWeb09-B (33 million documents, 100 million passages).

Retrieval Models. The *tf* and HDCT indexes were tested with three widely-used retrieval models.

BM25. The BM25 retrieval model [32] is a widely-used well-performing bag-of-words retrieval model.

BM25FE. BM25FE is an ensemble of BM25 rankers on different document fields. Field scores are linearly combined in the ensemble, where the weights are searched through a parameter sweep. ClueWeb datasets used title, URL, inlink, and body. MS-MARCO-Doc used title, URL and body; inlink was not available in this dataset. HDCT only weighted terms in the body field.

BM25+RM3. The relevance model RM3 [22] is a popular query expansion technique using pseudo-relevance feedback. BM25+RM3 has been shown to improve the original BM25, and has been considered a strong baseline. We also tested the compatibility between HDCT index and BM25+RM3 as described in Section 3.3.

We used the Anserini [40] implementation of the above retrieval models. We *tuned the parameters* of these retrieval models on the evaluation query sets through 2-fold cross-validation. These include: the k_1 and b parameters in BM25, the field weights in BM25FE, and the number of feedback documents, the number of feedback terms, and the feedback coefficient in BM25+RM3.

Embedding-Based Retrieval Baselines. We compared HDCT, which uses discrete bag-of-words, to two retrieval models that use embeddings: RLM [41] and SNRM [42]. Same as HDCT, they support efficient full-collection retrieval. RLM [41] makes use of word embedding similarities for pseudo-relevance feedback. SNRM [42] is the current state-of-the-art embedding-based index. It converts documents into sparse 20,000-dimension embeddings, and store them in an inverted index. RLM and SNRM are both trained using a PRF-based weak supervision approach [41, 42]. The authors did not release the trained models or indexes, and we were not able to fully optimize our own implementations due to the large amount of training data they require [5]. Therefore, we report the results reported by the authors on the ClueWeb09-B dataset [41, 42].

Supervised Re-Ranking Baselines. HDCT is designed for full-collection retrieval. We also compared it to two strong re-ranking systems, which are more computationally complex and require training data. The first, LeToR, is a popular feature-based learning-to-rank method using Coordinate Ascent [9, 11, 31]. The second, BERT-FirstP, is a neural BERT-based re-ranker [9]. BERT-FirstP has shown improved performance over previous state-of-the-art neural ranking models [9]. Both methods adopt the settings used by Dai et al. [11] and re-ranked the top 100 documents retrieved by Galago SDM.

6 EXPERIMENTAL RESULTS

Four experiments were conducted to study: the retrieval effectiveness of content-trained HDCT; the effects of stronger supervision using relevance-based and PRF-based labels; the effects of different types of hierarchical document modeling; and whether HDCT improves pseudo-relevance feedback.

[5] RLM and SNRM used 6 million queries and 6×10^7 to 6×10^{13} training examples [41, 42].

Table 2: Effectiveness of content-trained HDCT indexes on the ClueWeb09-C dataset. $*$ indicates statistically significant improvements over *tf*, the standard inverted index using term frequency.

ClueWeb09-C		Title Query						Description Query					
Retrieval Model	Indexing Term Weight	MRR		NDCG@20		MAP@1000		MRR		NDCG@20		MAP@1000	
BM25	*tf*	0.493	–	0.307	–	0.248	–	0.570	–	0.321	–	0.238	–
	HDCT-title	0.592*	20%	0.342*	11%	0.254	3%	0.608	7%	0.362*	13%	0.257*	8%
	HDCT-inlink	0.586*	19%	0.356*	16%	0.265*	7%	0.625	9%	0.377*	17%	0.264*	11%
BM25FE	*tf*	0.591	–	0.322	–	0.250	–	0.651	–	0.357	–	0.269	–
	HDCT-title	0.604	2%	0.358*	11%	0.263*	5%	0.663	2%	0.376*	5%	0.274	2%
	HDCT-inlink	0.615	4%	0.361*	12%	0.270*	8%	0.643	-1%	0.385*	8%	0.280*	4%
BM25+RM3	*tf*	0.563	–	0.350	–	0.278	–	0.581	–	0.351	–	0.257	–
	HDCT-title	0.610*	8%	0.369*	6%	0.280	1%	0.634*	9%	0.386*	10%	0.276*	7%
	HDCT-inlink	**0.630***	12%	**0.397***	14%	**0.298***	7%	**0.663***	14%	**0.399***	14%	**0.285***	11%

Table 3: Effectiveness of content-trained HDCT indexes on the ClueWeb12-C dataset. $*$ indicates statistically significant improvements over *tf*, the standard inverted index using term frequency.

ClueWeb12-C		Title Query						Description Query					
Retrieval Model	Indexing Term Weight	MRR		NDCG@20		MAP@1000		MRR		NDCG@20		MAP@1000	
BM25	*tf*	0.545	–	0.211	–	0.050	–	0.535	–	0.183	–	0.043	–
	HDCT-title	0.607*	11%	0.230*	9%	0.054*	8%	0.621*	18%	0.218*	19%	0.053*	22%
	HDCT-inlink	0.603	10%	0.232*	10%	0.055*	11%	0.602*	13%	0.215*	17%	0.052*	19%
BM25FE	*tf*	0.584	–	0.229	–	0.054	–	0.554	–	0.197	–	0.048	–
	HDCT-title	0.611	5%	0.236	3%	0.058*	6%	0.631*	12%	0.218*	11%	0.053*	9%
	HDCT-inlink	0.613*	5%	**0.241***	5%	**0.060***	11%	0.619*	12%	0.217*	10%	0.053*	10%
BM25+RM3	*tf*	0.567	–	0.216	–	0.051	–	0.503	–	0.186	–	0.043	–
	HDCT-title	**0.642***	13%	0.235*	9%	0.056*	10%	**0.635***	26%	**0.221***	19%	**0.054***	25%
	HDCT-inlink	0.622*	10%	0.241*	12%	0.058*	11%	0.61*0	21%	0.220*	19%	0.053*	21%

Table 4: Effectiveness of content-trained HDCT indexes on MS-MARCO-Doc. $*$: statistically significant improvements over *tf*, the standard inverted index using term frequency.

MS-MARCO-Doc		Dev Query	
Retrieval Model	Indexing Term Weight	MRR	
BM25	*tf*	0.254	–
	HDCT-title	0.287*	13%
BM25FE	*tf*	0.283	–
	HDCT-title	0.300*	6%
BM25+RM3	*tf*	0.250	–
	HDCT-title	0.288*	15%

6.1 Performance of Content-Trained HDCT

When building a search system for a new document collection, it is often the case that there are no relevance labels to train machine learning models. Typically, people would build a *tf*-based inverted index and use out-of-the-box retrieval models like BM25. Our goal is to construct a better index using content-trained HDCT without relying on any additional labels. The first experiment tests if content-trained HDCT can outperform standard *tf*-based retrieval models, strong supervised re-ranking models, and competitive embedding-based retrieval models.

Comparison to Standard *tf* Index. Tables 2-4 show the retrieval effectiveness of several content-trained HDCT indexes on the ClueWeb09-C, ClueWeb12-C and MS-MARCO-Doc datasets. HDCT-title and HDCT-inlink use document title/inlinks as the reference field to generate training labels. The baseline is a typical term-frequency (*tf*) based inverted index.

Significant and robust gains from HDCT over *tf* were observed on all datasets and query sets under various retrieval models.

When BM25 was used, HDCT indexes were 10%-20% more accurate than the *tf* index.It shows that HDCT weights are more effective than simply counting term frequencies in the document.

When BM25FE was used, the gap between HDCT and *tf* was smaller, but HDCT still outperformed *tf* in most cases. It shows that the content-trained HDCT can provide new information not covered by titles and inlinks. Titles and inlinks are often short and incomplete. Sometimes they have low text quality. Learning from a large number of titles/inlinks of different styles and qualities helps HDCT to capture general patterns of term importance, generating smoother and cleaner term weights than the original text fields.

RM3 is a pseudo-relevance feedback retrieval model originally designed for *tf* weights. Our results show that HDCT weights also fits into RM3. HDCT brought significantly improvements to the original *tf*-based BM25+RM3. The combination of HDCT-inlink index and BM25+RM3 retrieval achieved the best accuracy on ClueWeb09-C. Its

Table 5: Effectiveness of `HDCT-inlink` on the ClueWeb09-B dataset. We report MAP@100 because `LeToR` and `BERT-FirstP` re-ranked the top 100 documents. Superscripts 1-8 indicate statistically significant improvements over the corresponding methods, as labeled in the first column, e.g., ^{14} means that this result is statistically significantly better than methods 1 and 4.

ClueWeb09-B	Title Query						Description Query					
Method	MRR		NDCG@20		MAP@100		MRR		NDCG@20		MAP@100	
1 `BM25`, *tf*	0.477	-12%	0.272	-8%	0.154	-4%	0.471	-6%	0.234	-7%	0.134	-7%
2 `BM25FE`, *tf*	0.530^{1}	-2%	0.268	-9%	0.157	-3%	0.511^{13}	3%	0.250^{1}	-0%	0.139^{1}	-4%
3 `BM25+RM3`, *tf*	0.520^{1}	-4%	0.294^{12}	-0%	0.164^{1}	+2%	0.473	-6%	0.249^{1}	-1%	0.138	-5%
4 `LeToR`	0.543	–	0.295^{12}	–	0.161^{1}	–	0.503^{12}	–	0.251^{1}	–	0.145^{123}	–
5 `BERT-FirstP`	0.538^{14}	-1%	0.286^{12}	-3%	0.166^{12}	+3%	$\mathbf{0.532}^{1236}$	+6%	0.272^{1234}	+8%	$\mathbf{0.151}^{1236}$	+4%
6 `BM25, HDCT`	0.543^{1234}	+0%	0.303^{1235}	+3%	0.163^{1}	+1%	0.510^{13}	+1%	0.267^{1234}	+6%	0.143^{13}	-1%
7 `BM25FE, HDCT`	0.543^{1234}	+0%	0.303^{1235}	+3%	0.163^{1}	+1%	0.521^{1234}	+3%	0.271^{1234}	+8%	0.145^{123}	+0%
8 `BM25+RM3,HDCT`	$\mathbf{0.597}^{1-7}$	+10%	$\mathbf{0.326}^{1-7}$	+11%	$\mathbf{0.180}^{1-7}$	+12%	0.525^{12346}	+4%	$\mathbf{0.274}^{12346}$	+9%	0.148^{123}	+2%

Table 6: Comparison between `HDCT` and embedding-based retrieval models. Dataset: ClueWeb09-B, title queries. Results for RLM are from [41]; results for SNRM are from [42]. `HDCT` were evaluated using the methodology in [41, 42] to be directly comparable. Statistic significance test are not available as [41, 42] reported aggregated results.

	`RLM` [41]		`SNRM` [42]		`HDCT-inlink`	
tf Baseline	0.224	–	0.229	–	0.246	–
Retrieval	–	–	0.235	+3%	0.270	+10%
Retrieval+PRF	0.236	+5%	0.248	+8%	0.289	+17%

NDCG@20 has a 14% improvement from *tf*-based `BM25+RM3`, 22% from the the *tf*-based `BM25FE`, and 29% from the *tf*-based `BM25`.

Comparison to Supervised Re-ranking Systems. Next, we test `HDCT` on the ClueWeb09-B dataset. ClueWeb09-B is a standard test collection widely used in IR research. It allowed us to compare `HDCT` to other published results. Table 5 compares `HDCT-inlink`, the best model found on `ClueWeb09-C`, to the standard *tf* retrieval models and two supervised re-ranking systems: `LeToR` [9, 11, 31]; and `BERT-FirstP` [9]. They were trained on around 10,000 relevant query-document pairs. `HDCT-inlink` was not trained on any relevance labels.

`HDCT-inlink` outperformed the standard *tf* on ClueWeb09-B, which has 10× irrelevant documents as ClueWeb09-C. It demonstrates the robustness of `HDCT` to larger and noisier collections.

`HDCT-inlink` was also as good as or better than some of the re-ranking systems. On query titles, retrieval using `BM25+RM3` from `HDCT-inlink` was significantly more accurate than both re-rankers. On query descriptions, it outperformed `LeToR` and was on-par with `BERT-FirstP` at the top of the ranking. Neural ranking models like `BERT-FirstP` need a large amount of training data [9, 31]. ClueWeb09-B is a realistic condition in which only a moderate amount of training data is available. These results indicate that under such conditions, `HDCT` can be very competitive with some state-of-the-art supervised re-ranking pipelines without using any relevance labels.

In terms of efficiency, re-ranking models like `BERT-FirstP` are computationally expensive at the query time [27, 29]. `HDCT` index is built offline and uses simple bag-of-words retrieval at query time, making it preferable in efficiency-sensitive scenarios.

Comparison to Embedding-Based Retrieval. `HDCT` only uses exact term matching signals for retrieval. Embedding-based retrieval models, on the other hand, can match text fuzzily using embeddings. We compare `HDCT` to two embedding-based retrieval models: `RLM` [41] and `SNRM` [42]. Results are shown in Table 6. As discussed in Section 5, we report their performance in the original papers [41, 42], and re-evaluated `HDCT` using their methodology to be directly comparable [6].

As shown in Table 5, `RLM` was used in a pseudo-relevant back scenario (`Retrieval+PRF`) [41]. Its retrieval accuracy was lower than the other two methods. `SNRM` and `HDCT` can be used for stand-alone retrieval (`Retrieval`, where `HDCT` used `BM25`), or be combined with psuedo-relevance back (`Retrieval+PRF`, where `HDCT` used `BM25+RM3`). `HDCT` achieved higher absolute NDCG@20 values, and larger relative improvements from the corresponding *tf* baseline. In `SNRM`, documents are semantically matched by latent topics in embeddings, but may lose term specificity. This is a common issue in controlled vocabularies [35] and representation-based neural ranking models [16, 17]. `HDCT` represents documents using the free-text vocabulary. It has higher precision as it preserves the exact term matching signals which are critical in IR [16].

Summary. The analysis in this section shows that one can train effective `HDCT` models solely from the content of documents. `HDCT` is more accurate than strong *tf*-based baselines and state-of-the-art weakly-supervised embedding baselines. Being an efficient full-collection retrieval model, `HDCT` can be same or more accurate than the supervised and more complex re-ranking models under low-resource or cold-start conditions.

6.2 Effects of Relevance and Pseudo-Relevance Labels

The previous experiment demonstrates the effectiveness of content-trained `HDCT`. The next experiment studies `HDCT`'s performance when it was trained with stronger supervision from real search queries and relevance labels provided. Table 7 shows the effectiveness of `HDCT` models training using relevance labels and pseudo-relevance labels on the MS-MARCO-Doc dataset. We examined two types of pseudo-relevance labels that reflect what people might use

[6]NDCG@20 values of `HDCT-inlink` in Table 6 are different from Table 5 due to different evaluation methodologies. [41, 42] used the ClueWeb09-A qrels files. We followed the settings in [9, 11] and used ClueWeb09-B subset.

Table 7: Effectiveness of HDCT when trained with relevance labels and pseudo-relevance labels. Dataset: MS-MARCO-Doc. Superscripts 1-5 indicate statistically significant improvements over the corresponding methods, as labeled in the second column.

MS-MARCO-Doc		Dev Query	
Retrieval Model	Indexing Term Weight	MRR	
BM25FE	1 *tf*	0.283	–
	2 HDCT-title	0.300^{13}	+6%
	3 HDCT-PRFaol	0.291^{1}	+3%
	4 HDCT-PRFmarco	0.307^{123}	+8%
	5 HDCT-supervised	0.320^{1234}	+13%

in different settings: generated using in-domain and out-of-domain queries. All indexes were retrieved with BM25FE, the strongest retrieval model on MS-MARCO-Doc.

HDCT-supervised is a fully-supervised model that used in-domain queries and true relevance labels. As shown in Table 7 , it is the most effective, outperforming the title-trained one by 7%. Document titles are not necessarily aligned with user search intents [41]. Relevance-based labels can bias HDCT towards the retrieval task.

HDCT-PRFaol used out-of-domain pseudo-relevance labels from AOL queries. Results in Table 7 show that they were not effective. We observed similar results on ClueWeb datasets (results were not shown due to space limitations). Domain difference is a common challenge in weak-supervision [25]; this experiment reveals that it has a significant impact on HDCT.

HDCT-PRFmarco was trained on in-domain pseudo-relevance labels. It was the closest to the fully-supervised model, demonstrating that our PRF-based weak-supervision strategy is useful for in-domain queries. User activities, such as clicks, are not always accessible due to privacy regulations. In this case, the search queries alone can provide important evidence to build better document representations.

Summary. This experiment shows that the relevance-based supervision strategy can align HDCT with the search tasks, making it more effective than the content-trained models. Collecting relevance labels or user activities may be expensive or not permitted. Our PRF-based weak-supervision strategy provides a way to improve with the queries alone, which are often easier to collect.

6.3 Effects of Hierarchical Document Modeling

HDCT uses a two-level hierarchy that first estimates term weights in passages, and then combine them into document-level term weights. This section first studies the effectiveness of HDCT's term weighting at the passage-level. It then compares different ways of combining passages. Finally, it analyzes its behavior through a case study.

Table 8 shows the results of several passage-based document retrieval approaches. TruncateDocument truncates a document into a single passage of 512 tokens to fit BERT. PassageRetreival indexes and retrieves individual passages, and combines passage scores at the query time. A document's score is the average or maximum of its passage scores [9, 20, 38]. Both methods were

Table 8: Different ways of combining passages. *ptf* stands for passage term frequency. pHDCT stands for the passage-level term weights from HDCT. MARCO and CW09-B stand for MS-MARCO-Doc (dev queries) and ClueWeb09-B (title queries). HDCT was trained using titles on MARCO, and inlinks on CW09-B. Retrieval model: BM25. Superscripts 1-9 indicate statistically significant improvements over the corresponding methods.

Method	MARCO	CW09-B
	MRR	NDCG@20
1 *ptf*+TruncateDocument	0.240^{3}	0.189^{34}
2 pHDCT+TruncateDocument	0.265^{13457}	0.200^{134}
3 *ptf*+PassageRetrievalAvg	0.212	0.131
4 pHDCT+PassageRetrievalAvg	0.243^{3}	0.148^{3}
5 *ptf*+PassageRetrievalMax	0.236^{3}	0.212^{134}
6 pHDCT+PassageRetrievalMax	0.261^{1345}	0.233^{1-5}
7 *tf*	0.245^{35}	0.272^{1-6}
8 HDCT, sum	0.280^{1-7}	$\mathbf{0.303}^{1-7,9}$
9 HDCT, decay	$\mathbf{0.287}^{1-8}$	0.286^{1-7}

tested with term frequencies in passages (*ptf*), or HDCT's BERT-based passage-level term weights (pHDCT).

Passage Level Effectiveness. The BERT-based passage term weights (pHDCT) were more effective than term frequencies (*ptf*) in retrieving passages. In TruncateDocument, a document only has one passage. Ranking these passages using pHDCT had significantly higher accuracy than using *ptf*, showing that pHDCT can better capture the key terms in a passage. The results from the PassageRetrieval approaches also show HDCT's passage-level effectiveness. pHDCT generated more accurate query-passage relevance scores, so its final document rankings were also more accurate than *ptf*.

Alternative Ways to Combine Passages. Next, we compare various ways to combine the passage-specific term weights. A simple way is to truncate a document into a single passage. However, as shown in Table 8, there is a large gap between pHDCT + TruncatedDocument to HDCT, showing that the first few hundred words is not sufficient for effective document retrieval, and it is necessary to consider all passages in the document.

PassageRetrieval makes use of all passages combining passage retrieval scores at the query time. As shown in Table 8, none of the passage retrieval approaches were as good as HDCT. They focus on the local content but lose the global context. For example, a passage discussing "the act to protect Yellowstone" is likely to be mistakenly retrieved by a query that looks for general "act" (Table 9, P6).

HDCT combines passage representations at the index time. Our results show that it is significantly better than truncating documents or combining passage scores. The effectiveness of passage weighting depend on the dataset. The simple unweighted sum is robust across dataset. The position-decayed sum (decay) is less effective on ClueWeb09-B, probably due to that the position decay is too strong for the longer documents in ClueWeb09-B.

Case Study. Table 9 illustrates HDCT's hierarchical document modeling process. The Wikipedia web page of Yellowstone National

Table 9: An example of `HDCT-inlink` weighted passages of a ClueWeb09-B document. This document has 36 passages. P1-P36 show 5 terms with highest `HDCT` weights from 6 passages. The first and the last rows show 10 terms from the document with highest *tf* and `HDCT` weights.

Yellowstone National Park - Wikipedia		
tf	Doc	yellowstone: 247, park: 243, national: 147, 2007: 96, http:89, fire: 84, retrieve: 82, service:67, 03:47, year: 41
HDCT	P1	yellow:10, stone:10, park:9, national:9, yellowstone:3
	P6	park:8, act:8, public: 3, 1872: 4, superintendent: 3
	P18	yellowstone:10, bison:6, herd:5, park:5, animal:4
	P23	fire:14, yellowstone:5, 1988:5, forest:8, rockies:4
	P32	yellowstone:10, volcano:7, lake:6, national:4, dome:2
	P36	wikipedia:1
	Doc (sum)	yellowstone:315, park:204, national:146, lake:31, grand:30, us:22, montana:22, fire:43, forest:33, teton:21

Park from ClueWeb09-B contains over 10,000 words, and covers a wide range of topics including the park's history, geology, and recreation. As shown in Table 9, the original *tf* correctly favors important concepts like "yellowstone", but also favors non-content words such as "2007" and "retrieve".

HDCT successfully identifies essential terms in each passage, e.g., the act that created the park (P6), wildlife (P18), fires (P23), and scenes (P32). HDCT also recognized that these are different aspects of the central topic "Yellowstone". These examples show that HDCT is able to identify passage-specific key terms although it is not trained on passage-specific labels (Section).

HDCT then sums up the passages, generating a document bag-of-words that characterizes the entire document. It correctly shows the central terms of the entire document, e.g. "yellowstone". It also provides an overview of the diverse topics discussed in different passages, such as 'fire', 'lake', and 'grand teton'. HDCT's document vector are more representative than *tf*, leading to higher retrieval effectiveness as shown in previous experiments.

Table 9 also shows how HDCT implicitly down-weights unimportant passages. P36 is the Wikipedia disclaimer "All text is available under the term of...". Instead of giving high weights to locally-important terms, `HDCT-inlink` decides that the entire passage is not important. The highest term weight is 1 for 'wikipedia'; all other terms receive 0 weights. As a result, the disclaimer passage contributes little to the final document representation.

Summary. The passage retrieval experiment proves that HDCT better captures key terms in a passage than *tf*. When combining passages for document retrieval, we show that HDCT's approach that aggregates passage representations is more effective than the widely-used approach that aggregates passage scores. The case study demonstrates that HDCT can emphasize a document's recurring theme terms, identify specific topical terms in different passages, down-weight noisy terms, and suppress off-topic passages.

6.4 Effects of HDCT on RM3

Section 6.1 shows that HDCT significantly improved the pseudo-relevance feedback (PRF) retrieval algorithm BM25+RM3. The last experiment studies the effects of HDCT on PRF retrieval. HDCT weights affect PRF in two ways: Through the quality of the initial document ranking; and through the terms and frequencies used by the PRF algorithm. This experiment separates those effects by varying whether *tf* or HDCT weights are used in the document retrieval (BM25) and PRF (RM3) stages. Table 10 presents the results.

Table 10: `BM25+RM3` using *tf* and `HDCT-inlink` term weights. Dataset: ClueWeb09-B (title queries). Superscripts 1-4 indicate statistically significant improvements over the corresponding methods.

Method	BM25	RM3	NDCG@20
1	*tf*	*tf*	0.294
2	*tf*	HDCT	0.295
3	HDCT	*tf*	0.320^{12}
4	HDCT	HDCT	0.326^{12}

HDCT weights improve pseudo relevance feedback by providing a better document ranking to the PRF algorithm (the two BM25 HDCT rows). The HDCT weights neither help nor hurt the PRF algorithm (the two RM3 HDCT rows), perhaps because the HDCT document representation is focused on a small number of central terms.

7 CONCLUSION

This paper presents HDCT, a context-aware document term weighting framework for ad-hoc search. In HDCT, a term's weight is based on its context-specific importance in each individual passage, rather than the widely-used term frequencies. The output of HDCT is in the form of standard bag-of-words, allowing efficient and effective retrieval from an inverted index.

A content-based weak-supervision strategy is presented to train HDCT without using relevance labels. The content-trained HDCT achieved significant and robust improvements over standard *tf*-weighted retrieval models, strong embedding-based approaches, and some supervised learning-to-rank systems. Only relying on the document collection, this training strategy makes HDCT applicable to new collections and low-resource domains.

Further study shows that search-specific labels such as queries and clicks can improve HDCT by aligning it with the user search intents. Our PRF-based weak-supervision strategy provides a way to leverage the queries without using relevance labels or user clicks, which are sometimes harder to collect than the queries.

Analysis demonstrates the advantages of HDCT's hierarchical document modeling approach. A passage retrieval experiment shows that HDCT better captures key terms in a passage than *tf*. The passage-level evidence cannot be directly combined by aggregating their retrieval scores. HDCT successfully translates its passages-level gains into document retrieval by combining passage term weights. HDCT provides the retrieval model with both global and passage-specific key terms, so that documents can be accurately retrieved through an efficient bag-of-word retrieval.

8 ACKNOWLEDGMENTS

This research was supported by NSF grant IIS-1815528. Any opinions, findings, and conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Nima Asadi, Donald Metzler, Tamer Elsayed, and Jimmy Lin. 2011. Pseudo test collections for learning web search ranking functions. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval.*

[2] Richard Berendsen, Manos Tsagkias, Wouter Weerkamp, and Maarten De Rijke. 2013. Pseudo test collections for training and tuning microblog rankers. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval.*

[3] Adam L. Berger and John D. Lafferty. 1999. Information Retrieval as Statistical Translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

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

[5] Matteo Catena, Ophir Frieder, Cristina Ioana Muntean, Franco Maria Nardini, Raffaele Perego, and Nicola Tonellotto. 2019. Enhanced News Retrieval: Passages Lead the Way!. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM.

[6] Gordon V. Cormack, Mark D. Smucker, and Charles L. A. Clarke. 2011. Efficient and Effective Spam Filtering and Re-ranking for Large Web Datasets. *Information retrieval* (2011).

[7] W. Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. *Search Engines - Information Retrieval in Practice.* Pearson Education.

[8] Zhuyun Dai and Jamie Callan. 2019. Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval. *arXiv:1910.10687* (2019).

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

[10] Zhuyun Dai, Chenyan Xiong, and Jamie Callan. 2016. Query-biased Partioning for Selective Search. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management.*

[11] 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 International Conference on Web Search and Web Data Mining.*

[12] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval.* ACM.

[13] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *The 40th International ACM SIGIR Conference on Research & Development in Information Retrieval.*

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805* (2018).

[15] Nadav Eiron and Kevin S McCurley. 2003. Analysis of anchor text for web search. In *The 26th International ACM SIGIR Conference on Research & Development in Information Retrieval.*

[16] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proceedings of the 2016 ACM CIKM International Conference on Information and Knowledge Management.*

[17] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. 2019. A deep look into neural ranking models for information retrieval. *arXiv preprint arXiv:1903.06902* (2019).

[18] Christophe Van Gysel, Maarten De Rijke, and Evangelos Kanoulas. 2018. Neural vector spaces for unsupervised information retrieval. *ACM Transactions on Information Systems (TOIS)* (2018).

[19] Rong Jin, Alexander G. Hauptmann, and ChengXiang Zhai. 2002. Title language model for information retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[20] Marcin Kaszkiel and Justin Zobel. 1997. Passage retrieval revisited, In ACM SIGIR Forum. *The 20th International ACM SIGIR Conference on Research & Development in Information Retrieval.*

[21] Marcin Kaszkiel and Justin Zobel. 2001. Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology* (2001).

[22] 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.

[23] Xiaoyong Liu and W Bruce Croft. 2002. Passage retrieval based on language models. In *Proceedings of the eleventh international conference on Information and knowledge management.* ACM.

[24] Sean MacAvaney, Kai Hui, and Andrew Yates. 2017. An Approach for Weakly-Supervised Deep Information Retrieval. *arXiv preprint arXiv:1707.00189* (2017).

[25] Sean MacAvaney, Andrew Yates, Kai Hui, and Ophir Frieder. 2019. Content-Based Weak Supervision for Ad-Hoc Re-Ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.*

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

[27] Bhaskar Mitra, Corby Rosset, David Hawking, Nick Craswell, Fernando Diaz, and Emine Yilmaz. 2019. Incorporating Query Term Independence Assumption for Efficient Retrieval and Ranking using Deep Neural Networks. *arXiv preprint arXiv:1907.03693* (2019).

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

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

[30] 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.*

[31] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531* (2019).

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

[33] François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and TW-IDF: new approach to ad hoc IR. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management.*

[34] Gerard Salton, James Allan, and Chris Buckley. 1993. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval.*

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

[36] Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950* (2019).

[37] 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).

[38] Zhijing Wu, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. Investigating Passage-level Relevance and Its Role in Document-level Relevance Judgment. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[39] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR.*

[40] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[41] Hamed Zamani and W. Bruce Croft. 2017. Relevance-based Word Embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.*

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