# Incorporating Semantic Textual Similarity and Lexical Matching for Information Retrieval

**Hiroki Iida** and **Naoaki Okazaki**
Department of Computer Science, School of Computing, Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo, Japan
{hiroki.iida@nlp.c., okazaki@c.}titech.ac.jp

## Abstract

BERT, which has been successfully applied to many types of natural language processing (NLP) tasks, is also effective with various information retrieval (IR) tasks. However, it is not easy to obtain appropriate data for fine-tuning a BERT model. This paper proposes a method that can improve IR performance without fine-tuning the model on the target IR data. Focusing on words appearing in both a query and a document, we introduce local-similarity (LS). LS calculates the similarity of contextualized representations of the common words, encoded using a pre-trained model for the semantic textual similarity task. To incorporate the LS into the IR scoring, we propose local-similarity scoring (LSS) functions. Experimental results show that LSS outperforms BM25 on several representative benchmarks. We also demonstrate that LSS reflects improving the pre-trained model of LS to the higher IR performance. Our code is available at `https://github.com/nlp-titech/rerank_by_sts`.

## 1 Introduction

Semantic matching and relevance matching are closely related tasks with different goals. Mainly studied in natural language processing (NLP), semantic matching infers the similarity/dissimilarity of two pieces of text, which are usually of the same length. By contrast, relevance matching aims at ranking documents for a given query in information retrieval (IR), where a query is mostly much shorter than documents (Guo et al., 2016).

The recent advance of deep learning has achieved success in semantic matching. In addition, the advent of BERT (Devlin et al., 2019) allowed us to solve various semantic matching tasks, for example, paraphrasing and similarity, with the same model. BERT has also been successfully applied to IR tasks (Yang et al., 2019; Nogueira and Cho, 2019). However, it is not easy to obtain appropriate data for fine-tuning. Nogueira and Cho (2019) reported that BERT needs about 10,000 data to outperform BM25 (Robertson and Walker, 1994): a traditional scoring function based on lexical matching.

In this paper, to alleviate the difficulty, we study a method improving the performance without fine-tuning a model on the target IR data. The method incorporates semantic textual similarity (STS) based on words appearing in both a query and a document based on Guo et al. (2016), who reported that matching such words is one of the important factors for relevance matching. Concretely, we propose the local-similarity (LS) method. In essence, LS considers contextualized representations of tokens appearing in the query tokens rather than those of all tokens in a retrieved document. Incorporating LS into scoring functions, we present local-similarity scoring (LSS) functions.

We confirm the effectiveness of the LSS functions on three well-known IR benchmarks: MS MARCO Passage, MS MARCO Document (Bajaj et al., 2016), and Robust04 (Voorhees, 2004). A pre-trained model for LSS is based on Sentence-BERT (SBERT) (Reimers and Gurevych, 2019). To examine whether LSS is effective without fine-tuning SBERT on IR data, we use an SBERT model fine-tuned on natural language inference (NLI) datasets (SBERT-N) and paraphrase datasets (SBERT-P), which are strong baselines for the semantic textual similarity (STS) task (Agirre et al., 2012). The experimental results show that LSS functions with SBERT-N or SBERT-P achieve better re-ranking performances over BM25 (Robertson and Walker, 1994) on all benchmarks.

## 2 Related Work

### 2.1 Neural Ranker

Many neural ranking models have appeared recently. Guo et al. (2016) proposed a deep relevance matching model (DRMM), focusing on the difference between semantic matching and relevance matching. Subsequently, models utilizing both semantic matching and relevance matching have appeard (Hui et al., 2018; Rao et al., 2019). However, all of them are fine-tuned on the target IR data. We studied a method that enhances the IR performance without the target IR data.

Following the advent of BERT, it was shown that BERT is also effective for both ad-hoc retrieval (MacAvaney et al., 2019; Li et al., 2020; Ma et al., 2021) and question answer (QA) retrieval (Nogueira and Cho, 2019; Khattab and Zaharia, 2020). It has now become clear that representations of a query and a document from fine-tuned BERT are also effective (Karpukhin et al., 2020; Lin et al., 2020; Xiong et al., 2021; Gao et al., 2021; Qu et al., 2021). Furthermore, a model fine-tuned on IR data different from the target IR data is transferable (Yang et al., 2019; Yilmaz et al., 2019; Nogueira et al., 2020; Thakur et al., 2021). In particular, Yang et al. (2019) assumed that BERT could capture the relevance matching as long as we can fine-tune BERT on the appropriate data. However, they did not propose the methodology for detecting whether the data for fine-tuning the model are appropriate for the target IR data. Moreover, it is not easy to obtain appropriate data for fine-tuning. Thakur et al. (2021) showed that BM25 still outperforms BERT models on certain IR datasets even if the model is fine-tuned on an IR dataset. Thus, we propose a method that improves the IR performance even if the data for fine-tuning BERT are weakly related to the IR task.

### 2.2 Semantic Textual Similarity

The semantic textual similarity (STS) task is a representative semantic matching task that measures the semantic equivalence between two sentences (Agirre et al., 2012). Recently, many unsupervised approaches have been proposed, including pooling approaches (Arora et al., 2017; Ethayarajh, 2018), word-alignment approaches (Kusner et al., 2015; Zhelezniak et al., 2019; Wang et al., 2020; Yokoi et al., 2020), and representation learning approaches (Kiros et al., 2015; Logeswaran and Lee, 2018; Cer et al., 2018). SBERT (Reimers and Gurevych, 2019), in which BERT is trained on SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018) data, belongs to a representation learning approach. In addition, SBERT is one of the present baselines. However, to the best of our knowledge, no studies have applied the recent STS models to IR tasks. This study appears to be the first trial to utilize the recent STS model for IR tasks.

### 2.3 Query Expansion and Document Expansion

Query expansion and document expansion are alternative approaches incorporating semantic aspects into IR based on lexical matching. These approaches expand words from an original query and document. Several approaches currently utilize BERT for query or document expansion. QECE (Naseri et al., 2021) incorporates contextualized representation in a query expansion, and UDCG (Jeong et al., 2021) expands documents with abstractive text summarization models. However, because they rank the documents with lexical-matching scoring functions, such as BM25, they cannot reflect the context of the retrieved documents to the ranking scores.

## 3 Proposed Method

In this paper, we tackled the re-ranking task. In the re-ranking task, the top-$N$ retrieved documents are sorted using a scoring function that differs from the retriever. The goal is to assign higher scores to the target documents with a given query.

This paper aims to improve the IR performance with a pre-trained model of STS. However, STS indicates the similarity of two texts as a whole, and the similarity is affected by irrelevant words in the

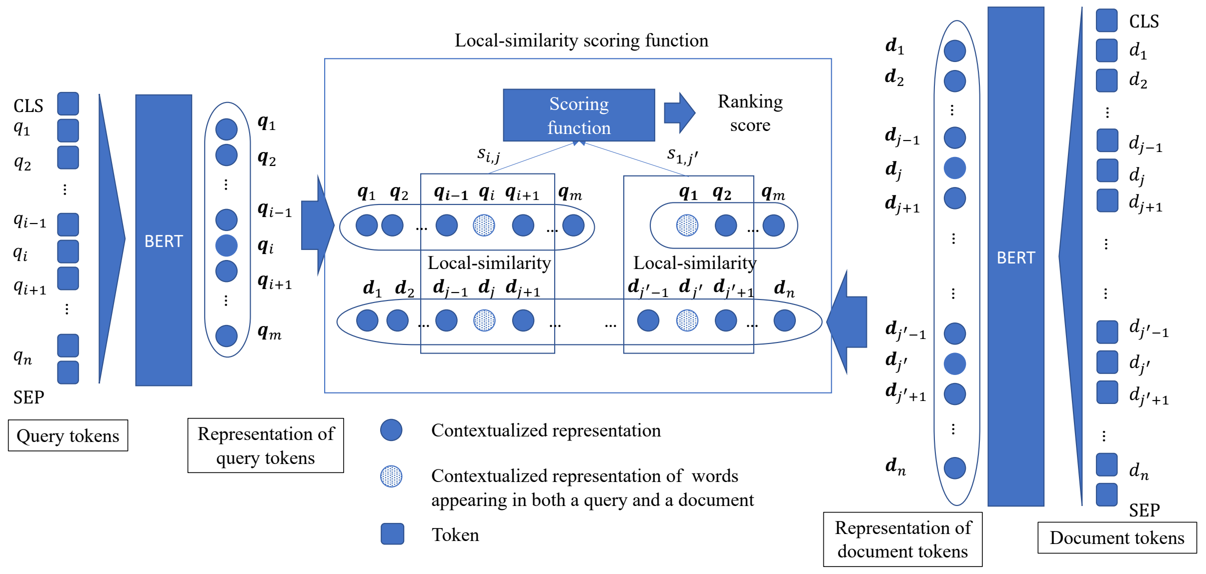
Local-similarity scoring function
Scoring function
Ranking score
$s_{i,j}$
$s_{1,j'}$
Local-similarity
Local-similarity
BERT
BERT
CLS
SEP
Query tokens
Representation of query tokens
Representation of document tokens
Document tokens
Contextualized representation
Contextualized representation of words appearing in both a query and a document
Token

Figure 1: Outline of local-similarity scoring functions (LSS)

retrieved documents. Guo et al. (2016) claimed that words appearing in both a query and a retrieved document play a key role in relevance matching. Thus, we hypothesize that scoring functions incorporating STS and lexical matching can improve the IR performance.

We realize this hypothesis with LSS functions. LS calculates similarity with contextualized representations of tokens appearing in both a query and a document. Figure 1 shows the outline of our proposed method. We present LS in Section 3.1 and several scoring functions in Section 3.2.

Herein, we define notations for explaining the functions. We denote a query set as $\mathcal{Q}$, a document set as $\mathcal{D}$, and a vocabulary set as $\mathcal{V}$. We represent a query $Q$ as a sequence of $m$ tokens, $Q = (q_1, q_2, \dots, q_m) \in \mathcal{V}^m$ and a document $D$ as a sequence of $n$ tokens, $D = (d_1, d_2, \dots, d_n) \in \mathcal{V}^n$. Each token is encoded to a $l$ dimension vector using a BERT model[1]. We express an encoded query $\boldsymbol{Q}$ as a sequence of $m$ encoded representations of query tokens $\boldsymbol{Q} = (\boldsymbol{q}_1, \boldsymbol{q}_2, \dots, \boldsymbol{q}_m) \in \mathbb{R}^{l\times m}$ and an encoded document $\boldsymbol{D}$ as a sequence of $n$ encoded representations of document tokens $\boldsymbol{D} = (\boldsymbol{d}_1, \boldsymbol{d}_2, \dots, \boldsymbol{d}_n) \in \mathbb{R}^{l\times n}$. Let $Q \cap D$ denote the set of tokens that appear both in the query $Q$ and the document $D$. We express the set of pairs of positions where the word $w$ appears in the query $Q$ and the document $D$,

$$P_{Q,D}(w) = \{(i,j) \in \mathbb{N}_m \times \mathbb{N}_n \mid q_i = d_j = w\}. \tag{1}$$

Here, $\mathbb{N}_m$ and $\mathbb{N}_n$ are the set of natural numbers $\{1, 2, \dots, m\}$ and $\{1, 2, \dots, n\}$.

## 3.1 Local-Similarity

LS computes the cosine similarity of contextualized representations of a query token $q_i$ and a document token $d_j$,

$$s_{i,j} = \cos(\boldsymbol{q}_i, \boldsymbol{d}_j). \tag{2}$$

We label Equation (2) as *token-based LS*.

We also explore a variant of LS because average pooling is widely used in the STS task. Equation (3) presents *pooling-based LS*,

$$s_{i,j} = \cos(\boldsymbol{h}_{q_i}, \boldsymbol{h}_{d_j}), \tag{3}$$

$$\boldsymbol{h}_{q_i} = \frac{1}{B} \sum_{k=i-o}^{i+o} \boldsymbol{q}_k, \tag{4}$$

$$\boldsymbol{h}_{d_j} = \frac{1}{B} \sum_{k=j-o}^{j+o} \boldsymbol{d}_k. \tag{5}$$

Here, we denote the window size as $o$ and $B = 2 \times o + 1$. We set a window size of $o = 5$ for all experiments.

[1]We include CLS and SEP tokens in an input, but ignore their representations.

### 3.2 Scoring Function

Scoring functions for LSS integrate LS values of tokens appearing in both the query and document to compute the relevance of a document for a given query. We consider three scoring functions inspired by the previous studies. Here, we denote a scoring function that scores a pair $(Q, D)$ as $f(Q, D)$.

**MAXSIM** MAXSIM directly incorporates the LS values to compute a score of between $Q$ and $D$,

$$f^{\mathrm{MaxSim}}(Q,D) = \sum_{w \in Q \cap D} \max_{i,j \in P_{Q,D}(w)} s_{i,j}. \tag{6}$$

In other words, this scoring function computes the sum of the maximum LS values for the tokens $w$ appearing in both $Q$ and $D$. Gao et al. (2021) introduced this function to improve the IR performance with a fine-tuned BERT model.

**MAXSIMIDF** Because MAXSIM cannot consider the importance of tokens in a query, we propose a variant of MAXSIM that integrates inverse document frequency (IDF),

$$f^{\mathrm{MaxSimIDF}}(Q,D) = \sum_{w \in Q \cap D} \mathrm{idf}(w) \max_{i,j \in P_{Q,D}(w)} s_{i,j} \tag{7}$$

Here, $\mathrm{idf}(w) = \log \frac{|\mathcal{D}|}{\mathrm{df}(w)}$, and $\mathrm{df}(w)$ is the document frequency of $w$. We name this scoring function MAXSIMIDF.

**BM25-MAXSIM** Chifu et al. (2015) presented a scoring function that weights a BM25 score with a coefficient $\alpha$[2]. Inspired by this idea, we propose a scoring function, called BM25-MAXSIM,

$$f^{\mathrm{BM25-MaxSim}}(Q,D) = \{1 + \alpha(Q,D)\}R(Q,D), \tag{8}$$

$$R(Q,D) = \sum_{w \in Q \cap D} \frac{\mathrm{idf}(w)\mathrm{tf}_D(w)(1+k_1)}{\mathrm{tf}_D(w) + k_1\{1 + b(\frac{n}{\mathrm{DA}} - 1)\}}. \tag{9}$$

Herein, $R(Q, D)$ presents a BM25 score for the query $Q$ and the document $D$, and $\mathrm{tf}_D(w)$ is the term frequency of $w$ in $D$. DA indicates the mean of the lengths of all documents, and $k_1$ and $b$ are hyper-parameters of BM25.

In this study, we incorporate LS to the weighting coefficient $\alpha$. We use a variant of MAXSIM to compute $\alpha(Q, D)$,

$$\alpha(Q,D) = \frac{1}{|Q \cap D|} f^{\mathrm{MaxSim}}(Q,D). \tag{10}$$

Herein, we divide $f^{\mathrm{MaxSim}}(Q, D)$ with $|Q \cap D|$ because the effect of the number of common tokens $w$ has already been incorporated in Equation (9).

[2]Chifu et al. (2015) firstly conducted word sense discrimination for words in a query by clustering the query and the retrieved documents with the query. Then, they multiplied $1 + \alpha$ to $R(Q, D)$ if the query and documents are in the same cluster. They determined $\alpha$ by grid search on test data. We consider that $\alpha$ can be replaced to LS.

## 4 Experimental Setup

We conducted experiments on the re-ranking task in three IR benchmarks. To confirm the effectiveness of the LSS functions, we compared them with several baseline approaches. We employ SBERT (Reimers and Gurevych, 2019) models as an encoder for obtaining contextualized representations of the queries and documents. We use two fine-tuned SBERT models, i.e., the SBERT model fine-tuned on natural language inference (NLI) datasets (SBERT-N) and the SBERT model fine-tuned on various paraphrase datasets (SBERT-P). According to the web site[3], SBERT-P performs better than SBERT-N in terms of the average score on five downstream tasks. Therefore, a comparison of SBERT-N and SBERT-P as an underlying encoder exhibits the impact of contextualized representations in solving the IR re-ranking task.

### 4.1 Benchmark Datasets

We used three IR benchmarks for the experiment: MS MARCO Passage, MS MARCO Document (Bajaj et al., 2016), and Robust04 (Voorhees, 2004). We chose these datasets to observe the performance when the proposed method and baseline methods handle different task types and different lengths of retrieved documents. Table 1 shows the task types and the average lengths of the three datasets. The top-1000 passages and top-100 documents were the re-ranking candidates of MS MARCO Passage and

[3]`https://github.com/UKPLab/sentence-transformers/blob/v2.0.0/docs/_static/html/models_en_sentence_embeddings.html`

Table 1: Statistics of IR benchmarks

| Dataset | Type | $\lvert Q \rvert$ | Average length of $D$ |
|---|---|---|---|
| MSMARCO Passsage dev | QA | 6980 | 77 |
| MSMARCO Passsage test | QA | 200 | 77 |
| MSMARCO Document test | QA | 200 | 1659 |
| Robust04 | ad-hoc | 250 | 647 |

Document respectively, following TREC 2019 Deep Learning Track. For Robust04, the top-1000 documents were the target, following Yilmaz et al. (2019). All candidates were retrieved using BM25 with the Anserini toolkit (Yang et al., 2017).

**MS MARCO Passage** This dataset is widely used as a QA retrieval benchmark, adopted as the TREC Deep Learning track. The queries and retrieved documents were obtained using Bing[4]. As the evaluation data, we used the dev-small dataset from MS MARCO-official[5] and the test dataset from TREC 2019 Deep Learning track [6]. The document length is the shortest among the three benchmarks because the unit to be retrieved is the passage rather than the document.

**MS MARCO Document** Although this dataset is highly similar to MS MARCO Passage, the dataset consists of documents instead of the passages. The document length of this dataset is the longest among the three benchmarks. We evaluated the performance on the test set of TREC 2019 Deep Learning track.

**Robust04**[7] This is a well-known ad-hoc retrieval dataset. TREC-Disk4 and 5 are the targets of the retrieval in this task. They mainly include news articles and government documents. This dataset has 250 test topics, and the titles of the topics were used as the queries. Although Robust04 can be considered as a document-level task, the lengths of the documents are shorter than those of the MS MARCO Document.

[4]`https://www.bing.com/`

[5]`https://microsoft.github.io/msmarco/`

[6]`https://microsoft.github.io/msmarco/TREC-Deep-Learning-2019.html`

[7]`https://trec.nist.gov/data/robust/04.guidelines.html`

## 4.2 Evaluation Metrics

We used MRR@10 as an evaluation metric for the MS MARCO Passage dev dataset, following the official MS MARCO leader-board. In addition, we used nDCG@10 for the test sets of MS MARCO Passage and MS MARCO Document, following TREC 2019 Deep Learning track. For Robust04, we adopted nDCG@20 to compare the proposed method with previous studies. The MRR evaluation was executed using MS MARCO-evaluation scripts in the Anserini toolkit, and the evaluations of nDCG were executed using `trec_eval`[8]. Tests of statistical significance were conducted using the paired-sample t-test.

## 4.3 Baseline Methods

We adopted the following methods as the baseline.

**BM25** This is the most famous scoring function based on lexical matching, and is still a strong baseline (Thakur et al., 2021). We used the implementation of the Anserini toolkit. The BM25 parameters are taken from the Anserini toolkit[9].

**Cos-sim** This method represents a query and a document as a vector, respectively. These vectors are calculated through average pooling[10]. The cosine similarity of the vectors is used as a score,

$$f(Q, D) = \cos(\boldsymbol{q}, \boldsymbol{d}) \tag{11}$$

$$\boldsymbol{q} = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{q}_i, \quad \boldsymbol{d} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{d}_j. \tag{12}$$

Note that Equation (11) is a part of the loss function when SBERT models were trained. Cos-sim is a popular function in the STS task because the similarity of this function considers two texts as a whole.

**ColBERT** This scoring function was adopted by Khattab and Zaharia (2020). We can consider it as a type of word-alignment approach in the STS task. The scoring of this function also indicates the

[8]`https://github.com/usnistgov/trec_eval`

[9]We used $(k_1, b) = (0.82, 0.68)$ for MS MARCO Passage, $(k_1, b) = (4.46, 0.82)$ for MS MARCO Document, and $(k_1, b) = (0.9, 0.4)$ for Robust04.

[10]We experimented with CLS-pooling, max-pooling, and average-pooling. The result of average-pooling showed the best performance.

similarity of two texts as a whole, such as Cos-sim. Formally, the scoring function is described as,

$$f(Q, D) = \sum_{i=1}^{m} \max_{j}(\cos(\boldsymbol{q}_i, \boldsymbol{d}_j)). \tag{13}$$

**Transferred BERT (TBERT)** As the baseline performance of the BERT model fine-tuned on IR data, we quoted the results of Yilmaz et al. (2019). We named their approach Transferred BERT (TBERT) because they transferred a BERT model fine-tuned on IR data to the target IR data, which is different from the IR data for fine-tuning. Depending on fine-tuning data, the authors experimented with two types of TBERT: the model fine-tuned on MS MARCO Passage (TBERT-M) and the model fine-tuned on MB dataset (Lin et al., 2014) and MS MARCO Passage (TBERT-B). The authors evaluated TBERT on Robust04, and TBERT-B achieved the best performance. The scoring function of TBERT is the weighted sum of the score from the retriever and fine-tuned BERT model. The weight is tuned through cross-validation on the target IR data.

**CEQE** CEQE (Naseri et al., 2021) is one of the best performing unsupervised IR methods. It utilizes contextualized word embedding from BERT for query expansion. We referred to the results from their paper.

### 4.4 Embeddings and Tokenizer

We tested LSS using two SBERT models: SBERT-N and SBERT-P. We used MPNet (Song et al., 2020) tokenizer implemented in huggingface transformers[11] (Wolf et al., 2020). On document-level tasks, the lengths of the documents are usually longer than 512. Thus, we encoded every segment of 512 tokens and ignored those after the 16,384th token because 99% of the length of documents is under 16,384.

**SBERT-N** SBERT-N is a SBERT model trained on NLI data, i.e. SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018). Concretely, we adopted the nli-mpnet-base-v2 model, which is the best model in terms of average on five benchmark tasks for SBERT[12] within the models whose training data does not include IR datasets. This model allows us to confirm whether LSS can improve IR performance with a pre-trained model for the STS task.

**SBERT-P** SBERT-P is an SBERT model trained on various paraphrase tasks. Concretely, the model is paraphrase-mpnet-base-v2, which is the best model among all SBERT models in terms of the average score of the benchmarks for SBERT. The training datasets of SBERT-P include not only NLI datasets but also MS MARCO Passage and yahoo_answers_title_question[13] among others[14]. Note that SBERT-P outperforms SBERT-N on the SBERT benchmarks. For this reason, we can claim that improving SBERT leads to the higher IR performance on LSS if LSS with SBERT-P performs better than LSS with SBERT-N on all IR benchmarks.

## 5 Result

This section first confirms whether the LSS function can outperform BM25 on all three benchmarks. Next, we investigate whether improving the pre-trained model can contribute to the IR performance on LSS. Finally, we analyze some cases qualitatively.

### 5.1 Effectiveness of LSS

Table 2 shows the results of LSS with SBERT-N and baseline methods. First, LSS outperforms Cos-sim and ColBERT. Thus, LSS appears effective for IR. LSS with *pooling-based LS* also outperforms BM25 on MS MARCO Passage and Document. In addition, BM25-MAXSIM improves the performance of Robust04 over that of BM25. In particular, BM25-MAXSIM with *pooling-based LS* performs the best within LSS on MS MARCO Document and Robust04, and the second-best on MS MARCO Passage. Therefore, BM25-MAXSIM with *pooling-based LS* can improve the IR performance most sta-

[11] https://github.com/huggingface/transformers

[12] https://github.com/UKPLab/sentence-transformers/blob/v2.0.0/docs/_static/html/models_en_sentence_embeddings.html

[13] https://www.kaggle.com/soumikrakshit/yahoo-answers-dataset

[14] All datasets are listed on https://www.sbert.net/examples/training/paraphrases/README.html

Table 2: The results of the baseline methods and LSS functions with SBERT-N on IR benchmarks. The best results are labeled in **bold** and the best result of LSS is expressed in *italics*. * denotes the statistical significance over BM25 (p-value $\leq$ 0.05).

| | | MS MARCO Passage | | MS MARCO Document | Robust04 |
|---|---|---|---|---|---|
| | | Dev MRR@10 | Test NDCG@10 | Test NDCG@10 | NDCG@20 |
| BM25 | | 0.1874 | 0.4973 | 0.5234 | 0.4240 |
| Cos-sim | | 0.1453 | 0.4461 | 0.4865 | 0.3576 |
| ColBERT | | 0.1149 | 0.3705 | 0.4007 | 0.2924 |
| CEQE | | – | – | 0.5614 | 0.4587 |
| TBERT-M | | – | – | – | 0.4512 |
| TBERT-B | | – | – | – | **0.5325** |
| **Local-similarity scoring function** | | | | | |
| **Scoring funciton** | **Local-similarity** | | | | |
| MaxSim | *token-based LS* | 0.1914 | 0.4896 | 0.5002 | 0.4087 |
| MaxSim | *pooling-based LS* | 0.2045* | 0.5193 | 0.5269 | 0.4246 |
| MaxSimIDF | *token-based LS* | 0.2145* | 0.5254 | 0.5278 | 0.4150 |
| MaxSimIDF | *pooling-based LS* | ***0.2318**** | ***0.5617*** | 0.5702 | 0.4334 |
| BM25-MaxSim | *token-based LS* | 0.2055* | 0.5256* | 0.5664* | 0.4567* |
| BM25-MaxSim | *pooling-based LS* | 0.2150* | 0.5541* | ***0.5792**** | *0.4649** |

Table 3: Comparison with SBERT-P and SBERT-N on LSS functions and Cos-sim. The best results are labeled in **bold**. * denotes statistical significance over BM25 (p-value $\leq$ 0.05).

| | MS MARCO Passage | | MS MARCO document | Robust04 |
|---|---|---|---|---|
| | Dev MRR@10 | Test NDCG@10 | Test NDCG@10 | NDCG@20 |
| **SBERT-P** | | | | |
| Cos-sim | 0.2819* | **0.6772*** | 0.6331* | 0.4236 |
| BM25-MaxSim with *pooling-based LS* | 0.2503* | 0.6053* | 0.6231* | **0.4691*** |
| MaxSimIDF with *pooling-based LS* | **0.2865*** | 0.6491* | **0.6383*** | 0.4588* |
| **SBERT-N** | | | | |
| Cos-sim | 0.1453 | 0.4461 | 0.4865 | 0.3576 |
| BM25-MaxSim with *pooling-based LS* | 0.2150* | 0.5541* | 0.5792* | 0.4649* |
| MaxSimIDF with *pooling-based* | 0.2318* | 0.5617 | 0.5702 | 0.4334 |

bly. By contrast, the performance of MaxSimIDF is the best on MS MARCO Passage. A difference between MaxSimIDF and BM25-MaxSim is whether the Term Frequency (TF) is considered. The TF may be important for a document-level IR.

MaxSimIDF performs better than MaxSim on all the benchmarks. To this point, the importance of each query term seems a main factor to improve the IR performance as claimed by Guo et al. (2016).

Comparing TBERT, the performance of BM25-MaxSim is higher than TBERT-M on Robust04 despite TBERT-M being trained on MS MARCO Passage, which is an IR dataset, and tuned hyperparameter on Robust04. By contrast, TBERT-B outperforms LSS. Thus, in spite of using a pre-trained model for the STS task, LSS can achieve the performance of TBERT when TBERT is fine-tuned on less appropriate IR data. Focusing on CEQE, BM25-MaxSim with *pooling-based LS* outperforms CEQE on MS MARCO Document and Robust04.

Finally, the *pooling-based LS* is better than *token-based LS* as the contextualized representation of words on all datasets; however, this may be because SBERT was trained through average pooling.

### 5.2 Effect of Model Improvement for Local-Similarity

Table 3 shows the results of Cos-sim and LSS with SBERT-P and SBERT-N. We used MaxSimIDF and BM25-MaxSim with *pooling-based LS* because they achieved the best result in LSS in the previous section.

Now, we investigate whether improving the underlying encoder leads to a higher IR performance on LSS. Tabel 3 shows that LSS with SBERT-P out-

Table 4: Examples of pairs of a query and a correct document from MS MARCO Passage. The ranking rows show the ranking of the correct document using BM25, Cos-sim, MAXSIMIDF and BM25-MAXSIM. The top example is what LSS ranked the correct document most highly. The middle example is where Cos-sim outperforms the other functions. On the bottom example, BM25 achieved the best result. The best function on each example is expressed in *italics*. Words appearing in the query and the document are labeled in **bold**.

| | |
|---|---|
| Query | umeclidinium cost |
| Correct Documnt | The average price of **umeclidinium** is $315 per month. It is less expensive than other long-acting anticholinergic inhalers (e.g., tiotro-pium, $360) and long-acting beta agonists (e.g., salmeterol [Serevent Diskus], $340). However, **umeclidinium** may be a higher-tier medication on certain health plans (typically tier 3 or 4), resulting in a higher co-pay, and it is not covered by all insurance plans. SIMPLICITY The recommended dosage of **umeclidinium** is one 62.5-mcg inhalation daily. The admin-istration device is preloaded with blisters |
| Ranking | BM25: 21 Cos-sim: 24 *MaxSimIDF*: 6 BM25-MAXSIM: 9 |
| Query | airplane definition of drag |
| Correct Documnt | **Drag** is the aerodynamic force that opposes an aircraft's motion through the air. **Drag** is generated by every part **of** the **airplane** (even the engines! How is drag generated? |
| Ranking | BM25: 11 *Cos-sim*: 5 MAXSIMIDF: 18 BM25-MAXSIM: 12 |
| Query | united home life insurance phone number |
| Correct Documnt | **United Home Life Insurance** Company(UHL) 225 S. East St Indianapolis, IN 46202 : *Phone*: 1-800-428-3001 : Website: www.unitedhomelife.com * A.M. Best: A- (Excellent) BBB: A-About: **United Home Life Insurance** Company was formed in 1948. They specialize in **life insurance** policies with simplified underwriting. This means there are no medical exam and no bodily fluids required for underwriting your policy. |
| Ranking | *BM25*: 4 Cos-sim: 326 MAXSIMIDF: 28 BM25-MAXSIM: 14 |

performs LSS with SBERT-N. Recall that SBERT-P outperforms SBERT-N on average with the benchmarks for SBERT. Therefore, the enhancement of the performance on the IR benchmarks suggests that LSS can reflect improving the model to IR tasks.

The improvement is more significant on MS MARCO Passage and Document than Robust04 because the data for training SBERT-P include MS MARCO Passage. Even though SBERT-P is trained using the loss function utilizing Cos-sim, MAXSIMIDF with *pooling-based LS* on SBERT-P can outperform Cos-sim on MS MARCO Passage dev and MSMARCO Document. Thus, LSS is effective if training data for SBERT include the target IR task.

## 5.3 Case Study

We then conducted a case study on MS MARCO Passage for a qualitative analysis. Table 4 shows examples.

The top example shows the reason why LSS can be successful. In the top example, *cost* appearing in the query did not appear in the correct document, but the context of the document was related to cost. LSS ranked the document higher, catching the context of the document. However, top-ranked documents using Cos-sim did not tend to include *umeclidinium*. This example illustrates the advantage of LSS, incorporating STS and lexical matching.

The middle example is a case in which the device of LSS was ineffective. Top-ranked documents by LSS of this example tended to include *definition* despite their definition not being about the drag of an airplane. However, the correct document does not include *definition*. Thus, this is the case lexical matching has a negative effect.

For the bottom example, both Cos-sim and LSS cannot outperform BM25. They scored high for a passage about the phone number of other insurance companies. This example suggests the importance of lexical matching on the relevance matching problems, as Guo et al. (2016) pointed out.

## 6 Conclusion and Future work

This paper proposed LSS, which can improve the IR performance without fine-tuning and even though the pre-training tasks for the LS model are semantic matching tasks. We showed that LSS can also reflect the improvement of the LS model to IR tasks.

Integrating other zero-shot approaches such as a query-generation and query expansion was out of the scope of this research. Future work will include the effect of the integration.

## Acknowledgments

We are grateful to the anonymous reviewers and Tatsuya Hiraoka who commented our paper. This paper is based on results obtained from a project, JPNP18002, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

## References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In *SemEval*, pages 385–393.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough to beat baseline for sentence embeddings. *ICLR*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A human generated MAchine reading COmprehension dataset. arXiv:1611.09268.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, pages 632–642.

Daniel Cer, Yinfei Yang, Sheng-Yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *EMNLP*, pages 169–174.

Adrian-Gabriel Chifu, Florentina Hristea, Josiane Mothe, and Marius Popescu. 2015. Word sense discrimination in information retrieval: A spectral clustering-based approach. *Information Processing Management*, 51(2):16–31.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, pages 4171–4186.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *RepL4NLP*, pages 91–100.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit exact lexical match in information retrieval with contextualized inverted list. In *NAACL*, pages 3030–3042.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *CIKM*, volume 24-28, pages 55–64.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A Context-Aware neural IR model for ad-hoc retrieval. In *WSDM*, pages 279–287.

Soyeong Jeong, Jinheon Baek, Chaehun Park, and Jong Park. 2021. Unsupervised document expansion for information retrieval with stochastic text generation. In *Workshop on Scholarly Document Processing*, pages 7–17.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-Tau Yih. 2020. Dense passage retrieval for Open-Domain question answering. In *EMNLP*, pages 6769–6781.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *SIGIR*, pages 39–48.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought vectors. *NIPS*, 28:3294–3302.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In Francis Bach and David Blei, editors, *ICML*, volume 37, pages 957–966.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. PARADE: Passage representation aggregation for document reranking. arXiv:2008.09093.

Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. 2014. Overview of the TREC-2014 microblog track. In *TREC*.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling dense representations for ranking using Tightly-Coupled teachers. arXiv:2010.11386.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *ICLR*.

Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2021. PROP: Pre-training with representative words prediction for ad-hoc retrieval. In *WSDM*, pages 283–291.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *SIGIR*, page 1101–1104.

Shahrzad Naseri, Jeffrey Dalton, Andrew Yates, and James Allan. 2021. CEQE: Contextualized embeddings for query expansion. In *ECIR*, pages 467–482.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. arXiv:1901.04085.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained Sequence-to-Sequence model. In *EMNLP*, pages 708–718.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for Open-Domain question answering. In *NAACL*, pages 5835–5847.

Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi, and Jimmy Lin. 2019. Bridging the gap between relevance matching and semantic matching for short text similarity modeling. In *EMNLP-IJCNLP*, pages 5370–5381.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-Networks. *EMNLP-IJCNLP*, pages 3980–3990.

Stephen E Robertson and Stephen Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. In *NeulIPS*, volume 33, pages 16857–16867.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. arXiv:2104.08663.

Ellen M Voorhees. 2004. Overview of the TREC 2004 robust retrieval track. In *TREC*.

Zihao Wang, Yong Zhang, and Hao Wu. 2020. Structural-Aware sentence similarity with recursive optimal transport. arXiv:2002.00745.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage challenge corpus for sentence understanding through inference. In *NAACL*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*, pages 38–45.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of lucene for information retrieval research. In *SIGIR*, pages 1253–1256.

Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Simple applications of BERT for ad hoc document retrieval. arXiv:1903.10972.

Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-Domain modeling of Sentence-Level evidence for document retrieval. In *EMNLP-IJCNLP*, pages 3490–3496.

Sho Yokoi, Ryo Takahashi, Reina Akama, Jun Suzuki, and Kentaro Inui. 2020. Word rotator's distance. In *EMNLP*, pages 2944–2960.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, Francesco Moramarco, Jack Flann, and Nils Y Hammerla. 2019. Don't settle for average, go for the max: Fuzzy sets and Max-Pooled word vectors. *ICLR*.