Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching

Liu Yang Mingyang Zhang Cheng Li Michael Bendersky Marc Najork
Google Research, Mountain View, CA, USA
{yangliuy,mingyang,chgli,bemike,najork}@google.com

ABSTRACT

Many natural language processing and information retrieval problems can be formalized as the task of semantic matching. Existing work in this area has been largely focused on matching between short texts (e.g., question answering), or between a short and a long text (e.g., ad-hoc retrieval). Semantic matching between long-form documents, which has many important applications like news recommendation, related article recommendation and document clustering, is relatively less explored and needs more research effort. In recent years, self-attention based models like Transformers [30] and BERT [6] have achieved state-of-the-art performance in the task of text matching. These models, however, are still limited to short text like a few sentences or one paragraph due to the quadratic computational complexity of self-attention with respect to input text length. In this paper, we address the issue by proposing the Siamese Multi-depth Transformer-based Hierarchical (SMITH) Encoder for long-form document matching. Our model contains several innovations to adapt self-attention models for longer text input. We propose a transformer based hierarchical encoder to capture the document structure information. In order to better capture sentence level semantic relations within a document, we pre-train the model with a novel masked sentence block language modeling task in addition to the masked word language modeling task used by BERT. Our experimental results on several benchmark datasets for long-form document matching show that our proposed SMITH model outperforms the previous state-of-the-art models including hierarchical attention [34], multi-depth attention-based hierarchical recurrent neural network [14], and BERT. Comparing to BERT based baselines, our model is able to increase maximum input text length from 512 to 2048. We will open source a Wikipedia based benchmark dataset, code and a pre-trained checkpoint to accelerate future research on long-form document matching.[1]

ACM Reference Format:
Liu Yang Mingyang Zhang Cheng Li Michael Bendersky Marc Najork. 2020. Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411908

[1]The code and a pre-trained checkpoint of the proposed SMITH model will be available at https://github.com/google-research/google-research/tree/master/smith. The Wikipedia based benchmark dataset will be available at https://github.com/google-research/google-research/tree/master/gwikimatch. Please note that different from the Wikipedia dataset used in this paper, the released dataset will contain both machine generated document pairs and human annotated document pairs. We hope the dataset can be a useful open benchmark for future research on document matching.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10.
https://doi.org/10.1145/3340531.3411908

1 INTRODUCTION

Semantic matching is an essential task for many natural language processing (NLP) and information retrieval (IR) problems. Research on semantic matching can potentially benefit a large family of applications including ad-hoc retrieval, question answering and recommender systems [17]. Semantic matching problems can be classified into four different categories according to text length, including short-to-short matching, short-to-long matching, long-to-short matching and long-to-long matching. Table 1 shows a classification of different semantic matching tasks with example datasets. Semantic matching between short text pairs is relatively well studied in previous research on paraphrase identification [38], natural language inference [1], answer sentence selection [35], etc. Short-to-long semantic matching like relevance modeling between query/ document pairs has also been a popular research topic in IR and NLP communities [3]. For long-to-short semantic matching, there are also a variety of research on tasks like conversation response ranking, which is to match a conversation context with response candidates [18]. To the best of our knowledge, semantic matching between long document pairs, which has many important applications like news recommendation, related article recommendation and document clustering, is less explored and needs more research effort. Table 2 shows an example of semantic matching between document pairs from Wikipedia. These documents have thousands of words organized in sections, passages and sentences.

Compared to semantic matching between short texts, or between short and long texts, semantic matching between long texts is a more challenging task due to a few reasons: 1) When both texts are long, matching them requires a more thorough understanding of semantic relations including matching pattern between text fragments with long distance; 2) Long documents contain internal structure like sections, passages and sentences. For human readers, document structure usually plays a key role for content understanding. Similarly, a model also needs to take document structure information into account for better document matching performance; 3) The processing of long texts is more likely to trigger practical issues like out of TPU/GPU memories without careful model design. In the recent two years, self-attention based models like Transformers [30] and BERT [6] have achieved the state-of-the-art performance in several natural language understanding tasks like sentence pair classification, single sentence classification and answer span detection. These kinds of models, however, are still limited to the representation and matching of short text sequences like sentences due to the quadratic computational time

Table 1: A classification of different semantic matching tasks. The focus of this paper is *long-to-long* document matching.

Type	Tasks	Example Data	Explanation
Short-to-short	Paraphrase Identification	MRPC [7]	Given two sentences, predict whether they have the same semantic meaning.
	Answer Sentence Selection	WikiQA [35]	Given a question and candidate answer sentences, select a correct answer.
	Textual Entailment	SNLI [1]	Given two sentences, predict whether they have textual entailment relations.
Short-to-long	Document Ranking	TREC 2019 Deep Learning [3]	Given a query and a candidate document set, rank documents according to query/ document relevance.
	Blog Search	TREC 2008 Blog Track [21]	Given a query and a blog collection, rank blog posts according to topic relevance or opinions.
Long-to-short	Response Ranking	UDC [18]	Given a dialog context and candidate responses, select a high quality response.
Long-to-long	Related Article Suggestion	Wikipedia [14]	Given a document pair, predict whether they are relevant with each other.
	Paper Citation Suggestion	AAN [25]	Given a paper pair, predict whether one paper is a good citation of the other.

Table 2: An example to illustrate the document matching task from the Wikipedia data. "Sim" means the similarity estimated based on the Jaccard similarity between the outgoing links of two documents. "Len" means the document length by words.

Type	URL	Title	Document Content	Label	Sim	Len
Source	http://en.wikipedia.org/wiki/Chartered_Engineer_(UK)	Chartered Engineer (UK)	In the United Kingdom, a Chartered Engineer is an Engineer registered with the Engineering Council (the British	\	\	1147
Target	http://en.wikipedia.org/wiki/Engineering_Council	Engineering Council	The Engineering Council (formerly Engineering Council UK; colloquially known as EngC) is the UK's regulatory	1	0.5846	999
Target	http://en.wikipedia.org/wiki/Institute_of_Physics	Institute of Physics	The Institute of Physics (IOP) is a UK-based learned society and professional body that works to advance physics	0	0.0099	2036

and space complexity of self-attention with respect to the input sequence length [30]. To handle this challenge, we would like to design a long-form text encoder, which combines the advantages of sequence dependency modeling with self-attention in Transformers and long text processing with hierarchical structures for document representation learning and matching.

Perhaps the closest prior research to our work is the study on the semantic text matching for long-form documents by Jiang et. al. [14]. They proposed the MASH RNN model to learn document representations from multiple abstraction levels of the document structure including passages, sentences and words. However, the adopted attentive RNN component in the MASH RNN model may suffer from the gradient vanishing and explosion problems on long input sequences. It is difficult for RNN based models to capture the long distance dependencies in long documents, which might lead to sub-optimal performance on long text content modeling compared with self-attention models like Transformers/ BERT where there are direct interactions between every token pair in a sequence. It is the right time to revisit this line of work and further push the boundary of long text content understanding with self-attention models like Transformers. However, as presented before, building Transformer based long text encoder is not trivial because of the quadratic computational time and memory complexity of self-attention with respect to the input sequence length. For example, the maximum input text length of BERT is 512 for single sentence classification, and less than 512 for sentence pair classification.

We address these issues by proposing the Siamese Multi-depth Transformer-based Hierarchical (SMITH) Encoder for document representation learning and matching, which contains several novel design choices to adapt self-attention models like Transformers/ BERT for modeling long text inputs. Our proposed text matching model adopts a two-tower structure of Siamese network, where each tower is a multi-depth Transformer-based hierarchical encoder to learn the document representations. We first split the input document into several sentence blocks, which may contain one or more sentences using our proposed greedy sentence filling method. Then the sentence level Transformers learn the contextual representations for the input tokens in each sentence block. We represent the whole sentence block with the contextual representations of the first token, following the practice in BERT. Given a sequence of sentence block representation, the document level Transformers learn the contextual representation for each sentence block and the final document representation. This model design brings several benefits in terms of model training and serving: 1) The Siamese model architecture is a better choice to serve with efficient similarity search libraries for dense vectors[15, 31], since document representations can be generated independently and indexed offline before online serving. 2) The hierarchical model can capture the document internal structural information like sentence boundaries. 3) Compared with directly applying Transformers to the whole document, the two level hierarchical SMITH model including sentence level and document level Transformers reduces the quadratic memory and time complexity by changing the full self-attention on the whole document to several local self-attentions within each sentence block. The sentence level Transformers capture the interactions between all token pairs within a sentence block, and the document level Transformers maintain the global interaction between different sentence blocks for long distance dependencies.

Inspired by the recent success of language model pre-training methods like BERT, SMITH also adopts the "unsupervised pre-training + fine-tuning" paradigm for the model training. For the model pre-training, we propose the masked sentence block language modeling task in addition to the original masked word language modeling task used in BERT for long text inputs. When the input text becomes long, both relations between words in a

sentence block and relations between sentence blocks within a document becomes important for content understanding. Therefore, we mask both randomly selected words and sentence blocks during model pre-training. The sum of the masked sentence block prediction loss and the masked word prediction loss is the final SMITH model pre-training loss. The model fine-tuning process is similar to BERT, where we remove the word/ sentence level masks and fine-tune the model parameters initialized with pre-trained checkpoints with only the text matching loss. We evaluate the proposed model with several benchmark data for long-form text matching [14]. The experimental results show that our proposed SMITH model outperforms the previous state-of-the-art Siamese matching models including hierarchical attention [34], multi-depth attention-based hierarchical recurrent neural network [14], and BERT for long-form document matching, and increases the maximum input text length from 512 to 2048 when compared with BERT-based baselines.

Our main contributions can be summarized as follows:

- We propose the Siamese Multi-depth Transformer-based Hierarchical (SMITH) Encoder for document matching, which contains several novel design choices to adapt self-attention models for modeling long text inputs.
- For model pre-training, we propose the masked sentence block language modeling task to capture sentence level semantic relations within a document towards better long text content understanding.
- Experimental results on several benchmark data for long-form text matching [14] show that our proposed SMITH model outperforms the previous state-of-the-art models and increases the maximum input text length from 512 to 2048 when comparing with BERT based baselines. We will open source a Wikipedia based benchmark dataset, code and a pre-trained model checkpoint to accelerate future research on document understanding and matching.

2 RELATED WORK

Neural Matching Models. A number of neural matching models have been proposed for information retrieval and natural language processing [8, 12, 13, 20, 22, 32–34, 39]. These models can be classified into the *representation-focused* models and the *interaction-focused* models [8, 9]. The *representation-focused* models learn the representations of queries and documents separately, and then they measure the similarity of the representations with functions like cosine, dot, bilinear or tensor layers. On the other hand, the *interaction-focused* models build a query-document word pairwise interaction matrix to capture the exact matching and semantic matching information between query-document pairs. Then a deep neural network which can be a CNN [12, 22, 39], term gating network with histogram or value shared weighting mechanism [8, 34] is applied to the query-document interaction matrix to generate the final ranking score. There are also neural matching models which combine the ideas of the *representation-focused* and *interaction-focused* models [20, 39]. Since it is difficult to serve *interaction-focused* models for online fast inference due to enormous computational costs of the interaction matrices for all query-document pairs, our proposed model belongs to *representation-focused* models, which are also called Siamese models or "Dual Encoder" models.

Self-Attention Models for Long Text Modeling. Self-attention models like Transformer and BERT show promising performance on several tasks in natural language processing and information retrieval. Most of these models are restricted to the representation and matching of short text sequences like sentences and passages. Our work is also built on top of Transformers with a different focus on effective representation learning and matching of long text. Recently there are some related works on adapting Transformers for long text modeling [2, 5, 11, 16, 24, 27–29, 36, 40]. Zhang et al. [40] proposed the HiBERT model for document summarization and a method to pre-train it using unlabeled data. Our work on the SMITH model is inspired by their research with several differences. First we split the document into sentence blocks with greedy sentence filling for more compact input text structures and less padded words. Second we build a Siamese "Dual Encoder" model for document pair similarity modeling. Third we propose a novel pre-training task based on dynamic masked sentence block prediction instead of predicting the masked sentence one word per step as in [40]. We also consider combining representations from different levels of hierarchical Transformers for richer representations.

Unsupervised Language Model Pre-training. The idea of unsupervised learning from plain text for language model pre-training has been explored in several works like Word2Vec[19], ELMo[23], GPT[26] and BERT[6]. These models can be pre-trained by predicting a word or a text span using other words within the same sentence. For example, Word2Vec can be trained by predicting one word with its surrounding words in a fixed text window and BERT pre-trains a language model by predicting masked missing words in a sentence given all the other words. We also study model pre-training techniques on plain text to improve the downstream document matching task. In addition to the masked word prediction task in BERT, we propose the masked sentence block prediction task to learn the relations between different sentence blocks.

3 METHOD OVERVIEW

3.1 Problem Formulation

We define the task of document matching following previous literature [14]. We are given a source document d_s and a set of candidate documents $\mathcal{D}_c$. The system needs to estimate the semantic similarities $\hat{y} = Sim(d_s, d_c)$, where $d_c \in \mathcal{D}_c$, for every document pair (d_s, d_c) so that the target documents semantically matched to the source document d_s have higher similarity scores. In practice, the documents may contain structural information like passage/ sentence/ word boundaries and different text length characteristics. The task can be formalized as a regression or classification problem depending on the type of data labels. A summary of key notations in this work is presented in Table 3.

3.2 Document Matching with Transformers

The original BERT model proposed by Devlin et. al. [6] supports text matching as the sentence pair classification task. Two input text sequences will be concatenated and separated by a special token [SEP] to feed into the BERT model, in order to learn the contextual representation of each token. Then the contextual representation of the first token, which is the added [CLS] token, will be projected into a probability distribution estimation over different label dimensions to compute the cross-entropy loss. Directly applying this

Table 3: A summary of key notations in this work.

d_s, $\mathcal{D}_s$	The source document and the set of all source documents
d_c, $\mathcal{D}_c$	The candidate document and the set of all candidate documents
$\mathbf{E}(d_s)$, $\mathbf{E}(d_c)$	The learned dense vector representations for d_s and d_c
L_1, L_2	The number of layers in the sentence level Transformers and in the document level Transformers
S_i, $\mathbf{E}(S_i)$	The i-th sentence block in the document and the sequence of word representations for S_i
W_j^i	The j-th word in the i-th sentence block in the document
L_d, L_s	The length of a document by sentence blocks and the length of a sentence block by words
$t(W_j^i)$	The token embedding for W_j^i
$p(W_j^i)$	The position embedding for W_j^i
$\mathbf{T}_j^i$, $\mathbf{S}_i$	The contextual token representation learned by sentence level Transformers for W_j^i and the contextual sentence block representation learned by document level Trnasformers for S_i
b, H, A, L	The batch size, the hidden size, number of attention heads and layers in Transformers

"Single Encoder" BERT model to the document matching task will cause two problems: 1) The input text length for each document will be very limited. On average, we can only feed at most 256 tokens per document into the BERT model to run the model fine-tuning or inference process of document matching. 2) The "Single Encoder" BERT model cannot be served for applications requiring high inference speed. To solve this problem, we learn query independent document representations and index them offline to serve with efficient similarity search libraries [15, 31]. Offline indexing of document representations requires generating dense vector representations for the two documents independently without expensive interactions in the earlier stage. This motivates us to focus on designing "Dual Encoder" BERT model with a Siamese network architecture, where each tower is to encode one document separately. The two towers can share model parameters. In the following sections, we introduce a basic Siamese matching model with Transformers MatchBERT (Section 3.2.1) and the Siamese hierarchical matching model SMITH (Section 3.2.2).

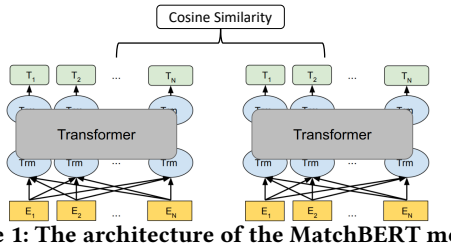

Figure 1: The architecture of the MatchBERT model.

3.2.1 ***MatchBERT: A Basic Siamese Matching Model with Transformers.*** Figure 1 shows the architecture of MatchBERT model for text matching adapted from the BERT model proposed by Devlin et. al. [6]. There are two text encoders in MatchBERT, where each encoder is a BERT model to learn the representation of the source document d_s or the candidate document d_c. Then we compute the cosine similarity between the pooled sequence output of two documents $cos(\mathbf{E}(d_s), \mathbf{E}(d_c))$. The text matching loss is the cross-entropy loss when we compare the document pair similarity scores with document pair labels. To handle long document input, MatchBERT will only model the first N tokens of each document, where the max value of N can be 512. To train the MatchBERT model, we initialize the model parameters with the open source pre-trained BERT checkpoints [2] and then fine tune the model with the text matching loss. MatchBERT will be a strong baseline model.

3.2.2 ***SMITH: Siamese Hierarchical Matching Model with Transformers.*** The SMITH model, which refers to **S**iamese **M**ult**I**-depth **T**ransformer-based **H**ierarchical Encoder, is an extension of the MatchBERT model. It also adopts a Siamese network architecture, where each tower is a transformer-based hierarchical encoder to learn representations in different levels like sentence and document level of long documents. In this way, it combines the advantages of long distance dependency modeling of self-attention in Transformer encoders and hierarchical document structure modeling for long text representation learning. Figure 2 shows the Transformer-based hierarchical encoder in the SMITH model.

4 SIAMESE HIERARCHICAL MATCHING MODEL WITH TRANSFORMERS

4.1 Hierarchical Modeling for Document Representation

4.1.1 ***Splitting Documents with Greedy Sentence Filling.*** In order to represent one document with hierarchical Transformers, we need to split the document into multiple smaller text units. A natural way to perform this step is to split a document into sentences with some off-the-shelf sentence boundary detection libraries. However, as sentence length varies a lot, padding all sentences to the same length will introduce a lot of padded tokens for short sentences, which will make the usage of the model capacity unnecessarily inefficient. We want to preserve each sentence's semantics so methods which may break down sentences like fixed length sliding window[4] are not good options either. We propose a "greedy sentence filling" method to reduce the number of padded words and increase the actual text length the model can take as its input. Specifically, we split a document into multiple sentence blocks of predefined length so that each sentence block can contain one or more natural sentences. We try to fill as many sentences as possible into one sentence block until the block reaches the predefined max block length. When the last sentence cannot fill in the current block, we move it to the next block. When an individual sentence alone is longer than the max sentence block length, we truncate it to fit in the current block. Figure 3 shows an example of how we split a document into sentence blocks. We can see that greedy sentence filling greatly reduces the number of padded tokens given a fixed maximum sentence block length.

4.1.2 ***Hierarchical Document Representation Learning.*** Let $\mathcal{D}$ denote the input document. With greedy sentence filling, we split it into a sequence of sentence blocks $\{S_1, S_2, ..., S_{L_d}\}$, where $S_i = \{W_1^i, W_2^i, ..., W_{L_s}^i\}$. S_i is the i-th sentence block in the document. W_j^i is the j-th word in the i-th sentence block. L_d and L_s denote the length of a document by sentence blocks and the length of a sentence block by words respectively. We learn the representation of each sentence S_i with the Transformer encoder described in [30], which consists of the multi-head self-attention and a position-wise fully connected feed-forward network with residual connections

[2] https://github.com/google-research/bert

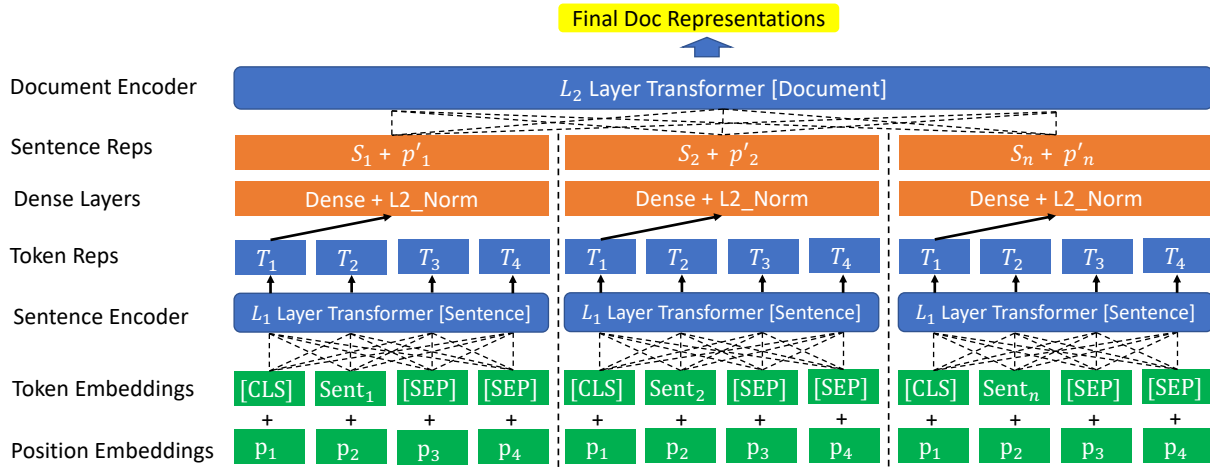

Figure 2: The architecture of the Multi-depth Transformer-based Hierarchical Encoder in the SMITH model for document representation learning and matching. We visualize the sentence level Transformer encoder for the 1st, 2nd and the last sentence block in a document. The output sentence representations of sentence encoders become the inputs of the document level Transformer encoder.

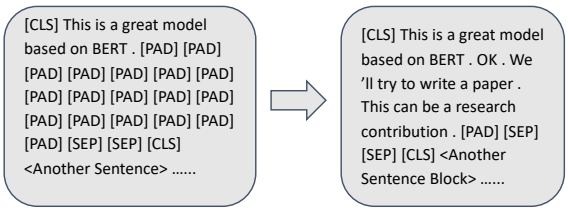

Figure 3: An example of splitting a document into different sentence blocks using the greedy sentence filling method. Left: natural sentence splitting. Right: greedy sentence filling splitting.

[10]. We firstly map the words in S_i to a sequence of dense vector representations

$$\mathbf{E}(S_i) = (\mathbf{e}^i_1, \mathbf{e}^i_2, \ldots, \mathbf{e}^i_{L_s}) \tag{1}$$

where $\mathbf{e}^i_j = t(W^i_j) + p(W^i_j)$ is the sum of the token embedding and position embedding of word W^i_j following the same practice in BERT. The token embedding is initialized randomly during pre-training and the position embedding follows the same setting as in Transformers [30]. The sentence level Transformers will transform $\mathbf{E}(S_i)$ into a sequence of contextualized representations for words in the sentence block $\{\mathbf{T}^i_1, \mathbf{T}^i_2, ..., \mathbf{T}^i_{L_s}\}$. Following the setting in BERT model, we use the contextual representation of the first token, the added [CLS] token, as the learned representation of the whole sentence block. We add another dense layer and perform a L2 normalization on the sentence block representation. The final sentence block representation also adds the sentence block position embedding to model the sentence block location in the document.

With the learned sentence block representations from the sentence level Transformers and the sentence block position embeddings, the document level Transformer encoders will produce a sequence of contextual sentence block representations $\{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_{L_d}\}$. We still use the first contextual sentence block representation as the representation for the whole document. There will be another dense layer added to transform the document representation with L2 normalization before we compute the cosine similarity between the two document representations in the document pair (d_s, d_c).

4.1.3 *Memory and Time Complexity Analysis of Two Level Hierarchical Transformers.*

Next let's analyze memory and time complexity for the two level hierarchical Transformers. The attention mechanism used in Transformer is the scaled dot-product attention, which performs transformation from a query and a set of key-value pairs to an output. The output representation is defined as a weighted sum of the values, where the weight to each value is computed as the interaction score between the query and the corresponding key normalized by the softmax function. Specifically, given the input query embeddings $\mathcal{Q}$, key embeddings $\mathcal{K}$ and value embeddings $\mathcal{V}$, where $\mathcal{Q} \in \mathbb{R}^{b \times l_Q \times H}$, $\mathcal{K} \in \mathbb{R}^{b \times l_K \times H}$, $\mathcal{V} \in \mathbb{R}^{b \times l_V \times H}$, the scaled dot-product attention is defined as:

$$\text{Attention}(\mathcal{Q}, \mathcal{K}, \mathcal{V}) = \text{softmax}\left(\frac{\mathcal{Q}\mathcal{K}^T}{\sqrt{d}}\right)\mathcal{V} \tag{2}$$

where l_Q, l_K, l_V are the number of tokens in each sequence and $l_K = l_V$. b is the batch size and H is the hidden size. To understand the memory cost of Transformers, we can focus on the attention computation in Equation 2. Let us assume $l_K = l_V = l_Q = N$, then the term $\mathcal{Q}\mathcal{K}^T$ has the shape $[b, N, N]$, where N is the maximum input sequence length. Let A and L denote the number of attention heads and layers in Transformers, then the memory complexity of the attention computation in Transformers is $O(b \cdot A \cdot N^2 \cdot L)$. This is why the memory cost of the scaled dot-product attention in Transformers grows quadratically as the increasing of the input sequence length, which makes it difficult to directly apply Transformers to very long input sequences [3]. Similar conclusions also hold for the time complexity of scaled dot-product used in

[3]This is the memory cost of Transformers without considering the feed-forward layers. For the complete memory complexity analysis results including both attention and feed-forward layers in Transformers, we refer the interested readers to [16].

Transformers. For two level hierarchical Transformers, let L_s denote the max sentence block length by tokens. Then we will split a document into $\frac{N}{L_s}$ sentence blocks. The memory complexity of the attention computation of sentence/ document level Transformers is

$$\begin{aligned} & b \cdot A \cdot {L_s}^2 \cdot L \cdot \frac{N}{L_s} + b \times A \times (\frac{N}{L_s})^2 \cdot L \quad (3) \\ = \quad & ({L_s}^2 \cdot \frac{N}{L_s} + (\frac{N}{L_s})^2) \cdot b \cdot A \cdot L \\ = \quad & (L_s \cdot N + \frac{N^2}{L_s^2}) \cdot b \cdot A \cdot L \end{aligned}$$

Here we assume the number of attention heads and the number of layers are the same for the sentence level Transformers and document level Transformers for simplicity. Thus the memory complexity of two level hierarchical Transformers is $O(\frac{N^2}{L_s^2} \cdot b \cdot A \cdot L)$. Comparing with the original Transformers, we reduce the memory complexity by a factor of L_s^2 with only performing local self-attention over tokens in the same sentence block.

4.1.4 ***Combine Representations from Different Levels.*** In order to integrate learned features from different levels of document structures, we consider several settings for generating the final document representations as follows:

Normal: we only use the output of the document level Transformers as the final document representation.

Sum-Concat: we firstly compute the sum of all sentence level representations and use the concatenation of the sum with the document level representation as the final document representation.

Mean-Concat: we firstly compute the mean of all sentence level representations and use the concatenation of the mean with the document level representation as the final document representation.

Attention: we firstly compute the weighted sum of the sentence level representations with attention mechanism: $\sum_{i=1}^{L_d} \mathbf{h}_i \cdot \text{softmax}(\mathbf{h}_i \mathbf{W} \mathbf{v})$, where $\mathbf{h}_i \in \mathbb{R}^H$ is the learned representation for the i-th sentence block by the sentence level Transformers. $\mathbf{W} \in \mathbb{R}^{H \times V}$ is the projection matrix and $\mathbf{v} \in \mathbb{R}^V$ is the attention model parameter. Then we concatenate the weighted sum with the document level representation as the final document representation.

4.2 SMITH Model Pre-training

For the model training of SMITH, we adopt the "pre-training + fine-tuning" paradigm as in BERT. This approach is to firstly pre-train the model with large unlabeled plain text in an unsupervised learning fashion, and then fine-tune the model with a supervised downstream task so that only few parameters need to be learned from scratch. In addition to the masked word language modeling task proposed by Devlin et. al. [6], we also propose the masked sentence block language modeling task, because one of the basic units in the SMITH encoder for modeling documents is the sentence block. Masked sentence block prediction task can help the model learn the relations between different sentence blocks and hopefully, get a better understanding of whole documents. Our pre-training loss is a sum of masked word prediction loss and the masked sentence block prediction loss. For the details of the masked word prediction task, please refer to Devlin et. al. [6]. For the masked sentence block prediction task, we perform dynamic sentence block masking and masked sentence block prediction as follows.

4.2.1 ***Dynamic Sentence Block Masking.*** Let $\mathbf{D} = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{L_d}\}$ denote a sequence of sentence block representations learned by the sentence level Transformers. For each document in the current batch, we randomly sample m sentence blocks $\mathcal{M} = \{\mathbf{h}_k | \mathbf{h}_k \in \mathbb{R}^H, k \in \mathcal{K}\}$ and replace these sentence blocks with a randomly initialized masked sentence block vector $\hat{\mathbf{h}} \in \mathbb{R}^H$. For example, if we randomly select the 3rd and 5th sentence block for masking, the masked document becomes $\hat{\mathbf{D}} = \{\mathbf{h}_1, \mathbf{h}_2, \hat{\mathbf{h}}, \mathbf{h}_4, \hat{\mathbf{h}}, \mathbf{h}_6, ..., \mathbf{h}_{L_d}\}$. This dynamic sampling process repeats for every document in a batch in each step, so that the same document may get different masked sentence block positions in different steps. The dynamic masking strategy enables the model to predict a larger range of sentence blocks in a document compared with the opposite static masking.

4.2.2 ***Masked Sentence Block Prediction.*** To perform masked sentence block prediction, we consider a multi-class sentence block classification setting similar to the masked word prediction. However, we do not have a global vocabulary for different sentence blocks. [4] Instead, we collect all the masked sentence blocks in a batch as a candidate sentence block pool, from which the model will try to predict the correct sentence block. For each masked sentence block position, the original sentence block in the current position is the positive example. The other co-masked sentence blocks in the current document and in the other documents of the same batch are the negative examples. Specifically, we apply document level Transformers on the masked document $\hat{\mathbf{D}}$ to get a sequence of contextual sentence block representations $\{\hat{\mathbf{S}}_1, \hat{\mathbf{S}}_2, ..., \hat{\mathbf{S}}_{L_d}\}$. Then $\hat{\mathbf{S}}_k$ will be used to predict the original sentence block representation $\mathbf{h}_k$. Given a batch of B masked sentence blocks with the predicted sentence block representation $\hat{\mathbf{S}} \in \mathbb{R}^{B \times H}$ and the ground truth sentence block representation $\mathbf{h} \in \mathbb{R}^{B \times H}$ where $B = b \times m$, we can compute a pairwise similarity matrix for every masked sentence block pair in the current batch as follows:

$$\text{Sim}(\hat{\mathbf{S}}, \mathbf{h}) = \hat{\mathbf{S}} \mathbf{h}^T \quad (4)$$

where $\text{Sim}(\hat{\mathbf{S}}, \mathbf{h}) \in \mathbb{R}^{B \times B}$ where $\text{Sim}(\hat{\mathbf{S}}_j, \mathbf{h}_i)$ is the predicted similarity between the j-th predicted sentence block representation and the i-th sentence block class. We normalize it with a softmax function to transform it to the predicted probability for the the i-th sentence block class as follows:

$$p(\mathbf{h}_i | \hat{\mathbf{S}}_j) = \frac{\exp(\text{Sim}(\hat{\mathbf{S}}_j, \mathbf{h}_i))}{\sum_{r=1}^{B} \exp(\text{Sim}(\hat{\mathbf{S}}_j, \mathbf{h}_r))} \quad (5)$$

Thus all the sentence blocks $\{\mathbf{h}_r\}$, where $r \in [1, B], r \neq j$ can be treated as randomly sampled negative classes for $\hat{\mathbf{S}}_j$. Finally we can compute the cross-entropy loss over all masked sentence blocks and the pre-training joint loss:

$$\mathcal{L}_{\text{sp}} = -\frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{B} \mathbb{1}\{j = i\} \log p(\mathbf{h}_i | \hat{\mathbf{S}}_j) \quad (6)$$

$$\mathcal{L}_{\text{pretrain}} = \mathcal{L}_{\text{sp}} + \mathcal{L}_{\text{wp}} \quad (7)$$

where $\mathcal{L}_{\text{sp}}$ and $\mathcal{L}_{\text{wp}}$ denote the masked sentence block prediction loss and the masked word prediction loss respectively.

[4] In fact, the number of all unique sentence blocks can be unlimited considering different composition of words into sentence blocks.

Table 4: The statistics of experimental datasets. # of DocPairs denotes the number of document pairs. AvgSPerD, AvgWPerD, AvgWPerS denote the average number of sentences per document, average number of words per document and average number of words per sentence respectively.

Data	Wiki65K			AAN104K		
Items	Train	Valid	Test	Train	Valid	Test
# of DocPairs	65,948	8,166	8,130	104,371	12,818	12,696
AvgSPerD	92.4	92.0	91.0	111.6	111.4	111.1
AvgWPerD	2035.3	2041.7	1992.3	3270.1	3251.2	3265.9
AvgWPerS	22.0	22.2	21.9	29.3	29.2	29.4

4.3 SMITH Model Fine-tuning and Inference

After model pre-training, we fine-tune SMITH model on the downstream document matching task with only the binary cross-entropy loss between the estimated matching probability and the ground truth matching label. Note that the word level and sentence block level language modeling masks added during the pre-training stage need to be removed during the fine-tuning stage to avoid losing document content information. After model pre-training and fine-tuning, the trained SMITH model can be used for document representation inference. The document representations inferred by SMITH model offline can be served online with fast similarity search libraries [15, 31].

5 EXPERIMENTS

5.1 Dataset Description

Following [14], we evaluate our proposed methods with two datasets: Wikipedia relevant document recommendation data (Wiki65K) and ACL Anthology Network paper citation suggestion data (AAN104K) from the previous related work. The statistics of the experimental datasets are shown in Table 4. Note that we do not use any TREC datasets like TREC 2019 Deep Learning Track data [3]. This is because these datasets focus on short-to-long matching like document ranking, aiming to match a short query to a set of documents, whereas our task is more on long-to-long document matching.

5.1.1 ***Relevant Document Recommendation Data.*** Relevant document recommendation can be useful in many real word applications such as news recommendation, related Web page recommendation, related QA posts recommendation, etc. We use the Wikipedia relevant document recommendation data from Jiang et. al. [14] as the evaluation set. The ground truth of document similarity is constructed based on the Jaccard similarity between the outgoing links of two Wikipedia documents with the assumption that similar documents have similar sets of outgoing links. The document pairs with similarities greater than 0.5 are considered as positive examples. For each positive document pair, the document with the lexicographical smaller URL is defined as the source document of the pair. Then a mismatched document from the outgoing links of the source document is sampled to generate a negative document pair. This negative sampling approach is better than random sampling from the entire corpus, since random sampled documents may be too irrelevant to make the task challenging enough to evaluate the performance of different methods. For more details of this dataset, we refer interested readers to [14].

Note that there are around six thousands training examples and less than one thousand validation/testing examples in the Wikipedia data used in [14] [5]. We produce a Wikipedia document matching dataset of ten times larger size as shown in the data statistics in Table 4. Thus the training/validation/testing data partition and statistics are different from the data in [14] and we report the results from running the model implementation on these datasets, which are different from the numbers reported in [14]. We refer to this data as Wiki65K since there are 65K training document pairs.

5.1.2 ***Paper Citation Suggestion Data.*** Paper citation suggestion can help researchers find related works and finish paper writing in a more efficient way. Given the content of a research paper and the other paper as a candidate citation, we would like to predict whether the candidate should be cited by the paper. We use the ACL Anthology Networks (AAN) Corpus [25] processed by Jiang et. al. [14] for the evaluation. The AAN data consists of 23,766 papers written by 18,862 authors in 373 venues related to NLP. For each paper with available text, the paper with each of its cited paper in the corpus is treated as a positive example. For each positive example, an irrelevant paper is randomly sampled to generate a negative example. The reference sections were removed to prevent the leakage of ground truth. The abstract sections were also removed to increase the difficulty of the task. We also filter document pairs with no section content or invalid UTF-8 text based on the processed data in [14] . We refer to this data as AAN104K since there are 104K training document pairs.

5.1.3 ***Unsupervised Language Model Pre-training Data.*** For the SMITH model pre-training, we create a randomly sampled Wikipedia collection, containing 714,800 documents with 43,532,832 sentences and 956,169,485 words. We pre-train SMITH model with unsupervised masked word and masked sentence block language modeling loss on this data and fine-tune the model on Wiki65K and AAN104K for the downstream document matching task.

5.2 Experimental Setup

5.2.1 ***Competing Methods.*** We consider different types of methods for comparison as follows: 1) **Hierarchical Attention Network (HAN).** The HAN model [37] is a hierarchical attention model to learn document representations. For each sentence in the document, it applied an attention-based RNN to learn the sentence representation and to attend differentially to more or less important document content. 2) **SMASH.** The SMASH model [14] is the state-of-the-art model for long document matching. It adopts a Siamese multi-depth attention-based hierarchical recurrent neural network (SMASH RNN) to learn long document representations for matching. 3) **MatchBERT.** The MatchBERT model has been presented in Section 3.2.1. 4) **SMITH.** This is our proposed method. We fixed the document level Transformers as 3 layers and tried several SMITH model variations as follows:

- **SMITH-Short**: the SMITH model with loading the pre-trained BERT checkpoint released by Devlin et al. [6]. We load the BERT-Base checkpoint pre-trained on uncased text and then fine-tune the model with only the document matching loss.

[5] https://research.google/pubs/pub47856/

Table 5: Comparison of different models over Wiki65K and AAN104K datasets. The best performance is highlighted in boldface. SMITH-WP+SP shows significant improvements over all baseline methods with p < 0.05 measured by Student's t-test. Note that SMITH-Short with input documents with maximum length larger than 512 and MatchBERT with input documents with maximum length larger than 256 will trigger the out-of-memory (OOM) issues on TPU V3. There is $\times 2$ in the BestDocLen, which denotes the best setting of the maximum input document length, since all the compared models are "Dual-Encoder"/ "Siamese" models. Note that all the compared models are for the document/ document matching task. Models designed for the query/ document matching task are not comparable.

	Data	Wiki65K				AAN104K			
Method	BestDocLen	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
HAN (NAACL16)	2048×2	0.8875	0.8571	0.9317	0.8928	0.8219	0.7895	0.8654	0.8257
SMASH (WWW19)	2048×2	0.9187	0.9177	0.9177	0.9177	0.8375	0.8224	0.8333	0.8278
MatchBERT	256×2	0.9316	0.9272	0.9366	0.9319	0.8355	0.8387	0.8201	0.8293
SMITH-Short	512×2	0.9415	0.9178	0.9699	0.9431	0.8212	0.8169	0.8161	0.8165
SMITH-NP	1536×2	0.9054	0.8911	0.9237	0.9071	0.7725	0.8106	0.7062	0.7548
SMITH-WP	1536×2	0.9492	0.9307	0.9707	0.9503	0.8400	0.8408	0.8354	0.8381
SMITH-WP+SP	1536×2	**0.9585**‡	**0.9466**‡	**0.9720**‡	**0.9591**‡	**0.8536**‡	**0.8431**‡	**0.8657**‡	**0.8543**‡
Δ over SMASH	NA	+4.33%	+3.15%	+5.92%	+4.51%	+1.92%	+2.52%	+3.89%	+3.19%
Δ over MatchBERT	NA	+2.89%	+2.09%	+3.78%	+2.92%	+2.17%	+0.52%	+5.56%	+3.01%

The maximum input text length is 512 (4 sentence blocks with 128 tokens per sentence block).

- **SMITH-NP**: the SMITH model without language modeling pre-training stage and trained from randomly initialized model parameters. We only train SMITH-NP model with the document matching data using text matching loss.
- **SMITH-WP**: the SMITH model pre-trained with masked word prediction loss in the pre-training collection and then fine-tuned with document matching loss on the downstream matching task data.
- **SMITH-WP+SP**: the SMITH model pre-trained with both masked word prediction loss and masked sentence block prediction loss on the pre-training collection and then fine-tuned with document matching loss on the downstream matching task data.

Note that we do not compare with any interaction-focused models like DRMM [8], K-NRM [33], Duet [20] or MatchPyramid [22]. These models either do not scale to long documents or require heavy interactions between word pairs in two text sequences, which will lead to long inference latency in practice. Thus all the compared methods belong to representation-focused models or "Dual-Encoder" models where the document representation can be learned offline in advance before serving online with fast similarity search libraries [15, 31]. Models like DRMM, KNRM, Duet, etc. are proposed for short-to-long text matching like query/document matching instead of long-to-long text matching that we focus on in this paper.

5.2.2 ***Evaluation Methodology***. We formalize the document matching task as a classification task where we would like to predict whether two documents are relevant or not given a document pair. Thus we consider standard classification metrics including accuracy, precision, recall and F1-score for the evaluation.

5.2.3 ***Parameter Settings and Implementation Details***. All models are implemented with TensorFlow[6]. We use TPU V3[7] for the model pre-training and fine-tuning. For the model pre-training stage, we pre-train SMITH on the Wikipedia collection presented in Section 5.1.3 around 68 epochs until the validation loss does not decrease significantly. Pre-training of SMITH with 2 layers in the sentence level Transformers and 3 layers in the document level Transformers with max document length 1024 takes 50 minutes for one epoch on the Wikipedia collection and the pre-training stage takes around 57 hours. The pre-training loss depends on the model variation types (masked word prediction loss for SMITH-WP or the sum of masked word prediction loss and masked sentence block prediction loss for SMITH-WP+SP). We dynamically mask 2 sentence blocks per document if we use the masked sentence block prediction loss presented in Section 4.2 during pre-training. The masked word prediction task follows the similar way by Devlin et al. [6]. The sentence block length is 32. We tune max number of sentence blocks with values in $\{32, 48, 64\}$. Thus the max document length can be values in $\{1024, 1536, 2048\}$. We finally set the number of sentence blocks as 48 (max document length is 1536) for both Wiki65K and AAN104K, which achieves the best performance on the validation data. We tune parameters using the validation dataset and report model performance on the test dataset. Let L_1, H_1, A_1 and L_2, H_2, A_2 denote the number of Transformer layers, the hidden size, the number of attention heads in sentence level Transformers and document level Transformers. We fix $L_2 = 3$ and tune L_1 with values in $\{2, 4, 6\}$. We finally set $L_1 = 6$, $H_2 = 256, A_2 = 4$ and $H_1 = 256, A_1 = 4$ for both Wiki65K and AAN104K. Both training and evaluation batch size are 32. We optimize the models using Adam with learning rate 5e-5, $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-6$. The dropout rate in all layers is 0.1.

For the model fine-tuning stage, the hyper-parameters are almost the same to those used in the pre-training stage. The max number of training steps is 100K. The number of learning rate warm up steps is 10K. We remove both masked word prediction loss and masked sentence block prediction loss during fine-tuning, and update the pre-trained model parameters only using the document matching loss. The fine-tuning stage takes much less time ranging from 4 to 12 hours depending on the model and data settings.

[6] https://www.tensorflow.org/
[7] https://cloud.google.com/tpu/

5.3 Evaluation Results

We present evaluation results of different models over Wiki65K and AAN104K data in Table 5. We summarize our observations as follows: 1) Both SMITH-WP and SMITH-WP+SP models outperform all the baseline methods including the stage-of-the-art long text matching method SMASH and MatchBERT based on pre-trained BERT models on both Wiki65K and AAN104K consistently. The comparison between SMITH-WP/ SMITH-Short and MatchBERT shows the effectiveness of introducing hierarchical document structure modeling with sentence level and document level Transformers for long document representation learning and matching. 2) If we compare the SMITH model settings with the pre-training stage (SMITH-Short, SMITH-WP, SMITH-WP+SP) with the SMITH model settings without the pre-training stage (SMITH-NP), we can find that language modeling pre-training can help increase the performance of the downstream document matching task by a large margin. Thus better language understanding via large scale language modeling pre-training will lead to better downstream task performance, which is consistent with the findings by Devlin et al. [6]. 3) Both SMITH-WP and SMITH-WP+SP outperform SMITH-Short, which is initialized by the pre-trained open source BERT model. We think the main reason is that currently SMITH-Short can only process at most 512 tokens due to TPU memory issues, which will hurt the performance. On the other hand, Both SMITH-WP and SMITH-WP+SP can process as long as 2048 tokens, which is a better setting for long document representation learning. 4) If we compare SMITH-WP with SMITH-WP+SP, we can find that adding the masked sentence block prediction task presented in Section 4.2 during the pre-training stage can also be helpful to improve the downstream document matching performance. The masked word prediction task proposed by Devlin et al. [6] can capture the word relations and dependencies in the pre-training corpus, whereas the masked sentence block prediction task can additionally force the model to learn the sentence-level relations and dependencies. Thus combining the masked word prediction task and the masked sentence block prediction task can contribute to a better pre-training language model for long document content understanding and better downstream document matching task performance.

5.4 Impact of Document Length

We further analyze the impact of document length on the document matching performance. We fix the number of layers in the sentence level and the document level Transformers as 4 and 3, the max sentence block length as 32. Then we vary the number of looped sentence blocks per document for SMITH-WP+SP on Wiki65K and AAN104K with different values from 2 to 64. Thus the maximum document length increases from 64 to 2048. The performances of SMITH-WP+SP with different choices of maximum document length is shown in Figure 4. We can find that in general SMITH-WP+SP will achieve better performance as the maximum document length increases. This confirms the necessity of long text content modeling for document matching. The SMITH model which enjoys longer input text lengths compared with other standard self-attention models is a better choice for long document representation learning and matching. We also studied the impact of different sentence block lengths and found it has no major impact on the final performance.

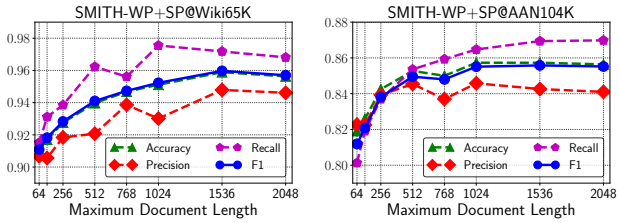

Figure 4: Performance of SMITH-WP+SP on the validation data with different choices of maximum document length.

Table 6: The document matching performance with different choices of the number of layers in the sentence level Transformers on the validation data. L_1 denotes the number of layers in the sentence level Transformers.

Data	L_1	Accuracy	Precision	Recall	F1
Wiki65K	2	0.9537	0.9449	0.9635	0.9541
	4	0.9589	**0.9479**	0.9718	0.9597
	6	**0.9594**	0.9426	**0.9784**	**0.9602**
AAN104K	2	0.8566	0.8470	0.8612	0.8540
	4	0.8573	0.8426	**0.8694**	**0.8558**
	6	**0.8580**	**0.8508**	0.8591	0.8549

5.5 Impact of the Number of Layers in Sentence Level Transformers

Next we analyze the impact of the number of layers in the sentence level Transformers on the final document matching performances. We set the document level Transformer layers as 3, the maximum sentence block length as 32, the number of sentence blocks per document as 48. So the maximum document length is 1536. Then we vary the number of layers in the sentence level Transformers and observe the change of the performances of SMITH-WP+SP. The results are shown in Table 6. We can find that the setting with 4 or 6 layers in the sentence level Transformer layers is slightly better than the setting with only 2 layers in the sentence level Transformers. Increasing the layers in the sentence level Transformers can help the model to learn sentence block semantic representation with more high level interactions. However, it also leads to larger memory cost such as the intermediate activation in each layer. Thus in practice this hyper-parameter has to be tuned with the validation dataset for the trade-off between the model capacity and memory cost.

5.6 Impact of Combining Representations from Different Levels

As presented in Section 4.1.4, we evaluate the performances of SMITH-WP+SP with different methods to combine representations from different levels. Table 7 shows the document matching performance with different choices of document representation combing methods. We can see that the "normal" combing method where we only use the output of the document level Transformers as the final document representation works best. For the other three methods,

Table 7: The document matching performance with different choices of the document representation combining methods presented in Section 4.1.4 on the validation data.

Data	Combine	Accuracy	Precision	Recall	F1
Wiki65K	Normal	**0.9594**	**0.9426**	0.9784	**0.9602**
	Sum-Concat	0.9192	0.9103	0.9301	0.9201
	Mean-Concat	0.9221	0.8924	0.9599	0.9249
	Attention	0.9431	0.9099	**0.9836**	0.9453
AAN104K	Normal	**0.8580**	**0.8508**	0.8591	**0.8549**
	Sum-Concat	0.7632	0.7924	0.6962	0.7412
	Mean-Concat	0.7061	0.6387	**0.9131**	0.7516
	Attention	0.8434	0.8162	0.8758	0.8450

the "attention" method is better than "sum-concat" and "mean-concat". One possible reason is that the weighted combination of sentence blocks can be helpful for generating better document representations as the attention weights can encode the relative importance of different sentence blocks on representing the document content, which is why attention based combining methods work better. The document level Transformers already learn a weighted combination based on the input sentence block representation sequences, which already provide enough signals on the importance scores of different sentence blocks in a document.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose the Siamese Multi-depth Transformer-based Hierarchical (SMITH) Encoder for document representation learning and matching, which contains several novel design choices like two level hierarchical Transformers to adapt self-attention models for long text inputs. For model pre-training, we propose the masked sentence block language modeling task in addition to the original masked word language modeling task in BERT, to capture sentence block relations within a document. The experimental results on several benchmark datasets show that our proposed SMITH model outperforms previous state-of-the-art Siamese matching models including HAN, SMASH and BERT for long-form document matching. Moreover, our proposed model increases the maximum input text length from 512 to 2048 when compared with BERT-based baseline methods.

As a part of this work, we plan to release a large scale benchmark collection for the document matching task so that it is easier for researchers to compare different document matching methods in the future. It is also interesting to investigate how to utilize the learned document representation from Transformer-based hierarchical encoders for other document-level language understanding tasks like document classification, clustering and ranking.

REFERENCES

[1] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP '15*. 632–642.

[2] R. Child, S. Gray, A. Radford, and I. Sutskever. 2019. Generating Long Sequences with Sparse Transformers. (2019). arXiv:1904.10509

[3] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M Voorhees. 2020. Overview of the TREC 2019 deep learning track. (2020). arXiv:2003.07820

[4] Z. Dai and J. Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *SIGIR '19*.

[5] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. 2019. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. (2019). arXiv:1901.02860

[6] J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018). arXiv:1810.04805

[7] W. B. Dolan and C. Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *IWP 2005*. 9–16.

[8] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM '16*. 55–64.

[9] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and X. Cheng. 2019. A Deep Look into Neural Ranking Models for Information Retrieval. (2019). arXiv:1903.06902

[10] K. He, X. Zhang, S. Ren, and J. Sun. 2015. Deep Residual Learning for Image Recognition. (2015). arXiv:1512.03385

[11] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans. 2019. Axial Attention in Multidimensional Transformers. (2019). arXiv:1912.12180

[12] B. Hu, Z. Lu, H. Li, and Q. Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS '14*. 2042–2050.

[13] P. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. P. Heck. 2013. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data. In *CIKM '13*. 2333–2338.

[14] J. Jiang, M. Zhang, C. Li, M. Bendersky, N. Golbandi, and M. Najork. 2019. Semantic Text Matching for Long-Form Documents. In *WWW '19*. 795–806.

[15] J. Johnson, M. Douze, and H. Jégou. 2017. Billion-scale similarity search with GPUs. (2017). arXiv:1702.08734

[16] N. Kitaev, L. Kaiser, and A. Levskaya. 2020. Reformer: The Efficient Transformer. In *ICLR '20*.

[17] H. Li and J. Xu. 2014. *Semantic Matching in Search*. Now Publishers Inc., Hanover, MA, USA.

[18] R. Lowe, N. Pow, I. Serban, and J. Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. (2015). arXiv:1506.08909

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS '13*. 3111–3119.

[20] B. Mitra, F. Diaz, and N. Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *WWW '17*. 1291–1299.

[21] I. Ounis, C. MacDonald, and I. Soboroff. 2008. Overview of the TREC 2008 Blog Track. In *TREC '08*.

[22] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. 2016. Text Matching as Image Recognition. In *AAAI '16*. 2793–2799.

[23] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. (2018). arXiv:1802.05365

[24] J. Qiu, H. Ma, O. Levy, S. W. Yih, S. Wang, and J. Tang. 2019. Blockwise Self-Attention for Long Document Understanding. (2019). arXiv:1911.02972

[25] D. R. Radev, P. Muthukrishnan, and V. Qazvinian. 2009. The ACL Anthology Network Corpus. In *NLPIR4DL '09*. 54–61.

[26] A. Radford. 2018. Improving Language Understanding by Generative Pre-Training. Preprint, OpenAI. (2018).

[27] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. 2019. Compressive Transformers for Long-Range Sequence Modelling. (2019). arXiv:1911.05507

[28] A. Roy, M. T. Saffar, D. Grangier, and A. Vaswani. 2020. Efficient Content-Based Sparse Attention with Routing Transformers. (2020). arXiv:2003.05997

[29] S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin. 2019. Adaptive Attention Span in Transformers. (2019). arXiv:1905.07799

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł . Kaiser, and I. Polosukhin. 2017. Attention is All You Need. In *NIPS '17*.

[31] X. Wu, R. Guo, A. Suresh, S. Kumar, D. Holtmann-Rice, D. Simcha, and F. Yu. 2017. Multiscale Quantization for Fast Similarity Search. In *NIPS '17*. 5745–5755.

[32] Y. Wu, W. Wu, C. Xing, M. Zhou, and Z. Li. 2017. Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-Based Chatbots. In *ACL '17*. 163–197.

[33] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR '17*. 55–64.

[34] L. Yang, Q. Ai, J. Guo, and W. B. Croft. 2016. aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model. In *CIKM '16*. 287–296.

[35] Y. Yang, W. Yih, and C. Meek. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *EMNLP '15*. 2013–2018.

[36] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. (2019). arXiv:1906.08237

[37] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *NAACL '16*. 1480–1489.

[38] W. Yin and H. Schütze. 2015. Convolutional Neural Network for Paraphrase Identification. In *NAACL '15*. 901–911.

[39] J. Yu, M. Qiu, J. Jiang, J. Huang, S. Song, W. Chu, and H. Chen. 2018. Modelling Domain Relationships for Transfer Learning on Retrieval-based Question Answering Systems in E-commerce. In *WSDM '18*. 682–690.

[40] X. Zhang, F. Wei, and M. Zhou. 2019. HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization. (2019). arXiv:1905.06566