Semantic Text Matching for Long-Form Documents

Jyun-Yu Jiang[†*], Mingyang Zhang[‡], Cheng Li[‡], Michael Bendersky[‡], Nadav Golbandi[‡], Marc Najork[‡]

[†]Department of Computer Science, University of California, Los Angeles, CA, USA
[‡]Google Inc., Mountain View, CA, USA
jyunyu@cs.ucla.edu,{mingyang,chgli,bemike,nadavg,najork}@google.com

ABSTRACT

Semantic text matching is one of the most important research problems in many domains, including, but not limited to, information retrieval, question answering, and recommendation. Among the different types of semantic text matching, long-document-to-long-document text matching has many applications, but has rarely been studied. Most existing approaches for semantic text matching have limited success in this setting, due to their inability to capture and distill the main ideas and topics from long-form text.

In this paper, we propose a novel Siamese multi-depth attention-based hierarchical recurrent neural network (SMASH RNN) that learns the long-form semantics, and enables long-form document based semantic text matching. In addition to word information, SMASH RNN is using the document structure to improve the representation of long-form documents. Specifically, SMASH RNN synthesizes information from different document structure levels, including paragraphs, sentences, and words. An attention-based hierarchical RNN derives a representation for each document structure level. Then, the representations learned from the different levels are aggregated to learn a more comprehensive semantic representation of the entire document. For semantic text matching, a Siamese structure couples the representations of a pair of documents, and infers a probabilistic score as their similarity.

We conduct an extensive empirical evaluation of SMASH RNN with three practical applications, including email attachment suggestion, related article recommendation, and citation recommendation. Experimental results on public data sets demonstrate that SMASH RNN significantly outperforms competitive baseline methods across various classification and ranking scenarios in the context of semantic matching of long-form documents.

CCS CONCEPTS

• **Information systems** → **Document representation**; **Document structure**; • **Computing methodologies** → *Information extraction*;

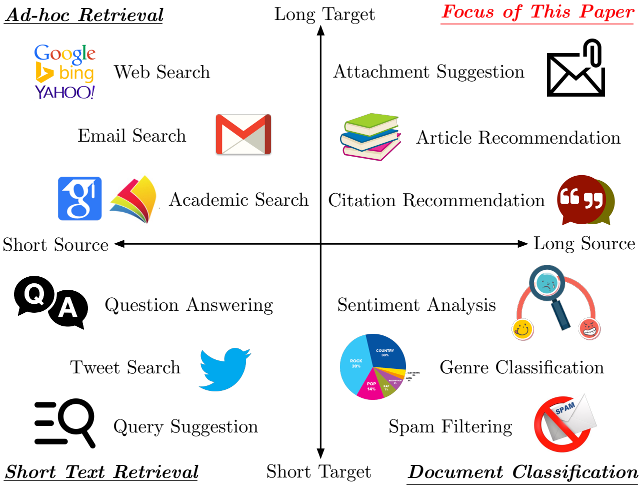

Figure 1: Applications across different lengths of source and target documents in text retrieval. In this work, we focus on the tasks with long sources and long targets.

KEYWORDS

Semantic text matching; long documents; hierarchical document structures; attention mechanism; recurrent neural networks.

ACM Reference Format:
Jyun-Yu Jiang, Mingyang Zhang, Cheng Li, Michael Bendersky, Nadav Golbandi, Marc Najork. 2019. Semantic Text Matching for Long-Form Documents. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Semantic text matching estimates semantic similarity between a source and a target text pieces (e.g., query-to-document match, question-to-paragraph match, etc.). Correctly modeling semantics in text matching has long been the "holy grail" of textual information retrieval. The difficulties of semantic matching are two-fold: first, semantics of words and phrases can be ambiguous; second, when the text is long, semantics of individual words, phrases and sentences can be buried in complex document structures. While the prior research mainly focuses on the first set of difficulties, in this paper we tackle the second challenge; namely that of dealing with complex long-form documents.

To better understand the effects of document length in semantic text matching, in Figure 1 we show applications of semantic text matching across the length spectrum of source and target texts. As shown on the upper left of this figure, ad-hoc retrieval tasks

[*]Work done while interning at Google.

WWW'19, May 2019, San Francisco, CA, USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

like web search [7] use short source queries while targeting long-form documents; on the lower left, short text retrieval tasks like Twitter search [9, 41] use short source queries and target short documents; on the lower right, the tasks of document classification like sentiment analysis aim to categorize long-form documents into a limited set of classes [25, 30]. Interestingly, the upper right part of the figure is relatively less explored in the semantic text match setting, and, as we empirically demonstrate, many of the previously proposed semantic matching methods deteriorate when source and target documents become longer. This poses an important research challenge, since semantic text matching for long-form documents can benefit a myriad of applications, such as related article recommendation, email attachment suggestion, citation recommendation, etc.

One of the most conventional approaches to semantic text matching is to compute a vector as the representation for each document, such as bag-of-words and latent Dirichlet allocation models [5, 18], and then apply typical similarity metrics to compute the matching scores. Unfortunately, the performance of traditional approaches is unsatisfactory, as they often fail to capture the semantic document structure. The advances in deep learning in recent years provide the opportunity to understand complex natural language and to learn better long-form document representations. For instance, recurrent neural networks (RNNs) treat a document as a sequence of words and derive a document representation based on the information along the sequence [32, 33]; convolutional neural networks (CNNs) obtain document representations by preserving local patterns [23, 38, 54].

However, although existing deep learning approaches significantly advanced the field of semantic text matching, they mainly focus on short documents and have apparent drawbacks when dealing with long-form documents. First, the core topic or idea can be hard to identify and extract from a complex narrative of a long-form document. Some of the previous studies [10, 52, 54] exploit attention mechanism [29] to distill important words from sentences, but valuable information can still be diluted within a large number of sentences and paragraphs in long-form documents. Second, the complex structural information of long-form documents have not yet been taken into account. Most of the existing approaches rely on word-level knowledge to compute text similarity. Structural information like relations among sentences and paragraphs is often disregarded. Third, the semantics of a document can drift over the course of a long narrative. For example, it is not uncommon to find documents in which the writer moves across a spectrum of subjects in the span of several passages. Neither RNN or CNN can naturally capture or follow semantic drifts like this. RNN-based approaches can attain confusing document representations by sequentially processing sentences with different semantics. CNN-based approaches can deteriorate when trying to pool and filter different semantics.

Natural language documents generally follow hierarchical structures [35] to help people read and understand them. Therefore, it is vital to utilize these structures in order to train machine learning models which can fully capture the semantics of long-form documents. Most generally, a document can be represented as a hierarchy of paragraph, sentence and word sequences. Different paragraphs and sentences in a document can have different semantic meaning and importance. The most similar research to this work is to use sentence-level information for document classification [10, 52]. As we will show in the experimental section, for long-form documents, sentence-level based document representations are still unsatisfactory because sentences in the same document may be associated with different importance and diverse semantics. On the contrary, a deep understanding of document structure can effectively boost semantic text matching.

In this paper, we propose Siamese multi-depth attention-based hierarchical RNN (SMASH RNN) to address the problem of long-form document semantic matching. Under the two-tower structure of Siamese network [32], each tower of the proposed model is a multi-depth attention-based hierarchical RNN (MASH RNN). MASH RNN, as the major component of our model, can derive comprehensive document representations from multiple levels of document structure. For example, word-, sentence-, and paragraph-level knowledge of a document can be derived by three attention-based hierarchical RNNs with different depths. To generate comprehensive document presentations, MASH RNN concatenates representations from all of these document levels, aiming to capture both concrete low-level observations and abstract high-level insights. Combining the document representations for both source and target documents from MASH RNN, SMASH RNN estimates semantic matching scores based on the representations of the source and the target documents, and an extra fully connected layer.

Our contributions can be summarized as follows:

- To the best of our knowledge, this paper is the first work to extensively exploit document structure for better document representations, in the context of improving the state-of-the-art performance of the long-form document semantic text matching models.
- We propose the SMASH RNN framework for long-form document semantic text matching. MASH RNN, the major component of SMASH RNN, learns document representations from multiple abstraction levels of the document structure.
- Experiments were conducted on publicly available datasets for three different applications: email attachment suggestion, related article recommendation, and citation recommendation. Experiment results demonstrate the effectiveness of SMASH RNN. We also provide an in-depth experiment analysis to prove the robustness of our proposed framework.

The rest of the paper is organized as follows. First, we present related work in Section 2. Then, the problem statement and SMASH RNN are described in Section 3. We show experiment results and provide an in-depth analysis in Section 4. Finally, some conclusions are given in Section 5.

2 RELATED WORK

In this section, we first give the background of semantic text matching and indicate the difference between our approach and previous studies. Deep document classification is then introduced as a related task. Last but not least, we discuss the attention mechanism in deep learning, which plays an important role in SMASH RNN.

2.1 Semantic Text Matching

To measure the similarity between documents, it is intuitive for traditional approaches to compare the words in the documents.

For example, Mihalcea et al. [31] compute the word-to-word similarity while Wu et al. [49] exploit the vector space model with the term frequency-inverse document frequency (TF-IDF). However, words in the documents are extremely sparse. In addition, the semantics between individual words also cannot be captured, so these approaches usually obtain unsatisfactory results. Although some works attempt to leverage the semantics in external resources such as knowledge bases [42] or alleviate the data sparseness by Latent Semantic Analysis (LSA) [53], traditional approaches are still limited by discrete words.

The recent development of deep learning in natural language processing provides a new opportunity for semantic text matching. After using deep learning models to encode the documents in distributed representations, the Siamese structure [8] for metric learning is usually applied to learn the similarity information. There is a plethora of deep learning models for encoding documents. For instance, some studies [2, 27, 32, 33, 45, 46] exploit RNNs to model the sequential information of documents while other work [19, 38, 46, 54] applies CNNs to capture the important local messages in documents for semantic text matching.

However, most of the previous studies are not designed for long-form documents. For example, RNNs can lose important messages while passing information along a very long sequence of words. The local features learned by CNNs can be inadequate to represent the complex semantics of long-form documents.

Another line of work maps the term positions of the source and the target texts to create a matching matrix [27, 45, 46, 50, 54] or computes local interactions with the target text for each word in the source text [16]. This works well for scenarios like ad-hoc retrieval, where the source queries are short. For a scenario where both the source and the target are long documents, a matching matrix would be time-consuming to compute, and is too large to fit in memory.

In addition, the document structures that could be much beneficial are generally ignored by most of the previous semantic text matching approcaches. Note that although Liu et al. [27] mention the term "hierarchical structure," they focus on the structures in sentences so that long-form documents still cannot be handled. In other words, their proposed methods are incapable to deal with long-form documents. Our approach differs from this work in that the hierarchical document structures with different depths are explicitly considered in the model, thereby directly benefiting the model to learn the semantics of different levels in the document structure.

2.2 Deep Document Classification

Instead of simultaneously modeling two documents, deep document classification that considers a single document at a time with deep learning models can be considered as a related task. Similar to semantic text matching, deep learning models can also be utilized to encode documents into distributed representations for document classification. For example, both CNNs [13, 25, 39] and RNNs [10, 28, 52] can do document classification for many applications such as sentiment analysis. Devlin et al. [12] propose bidirectional encoder representations from transformers (BERT). However, all of these works focus on short text classification, especially for sentences and documents with few sentences. Even though BERT quickly became the state-of-the-art for many NLP tasks, its multi-head attention component requires an unrealistically enormous amount of both memory and computation time for long-form documents. Moreover, most of their approaches only take word-level information into account and ignore the structure of long-form documents. HAN [52] and NSC [10] with the exactly same architecture are the only two studies using a two-layer RNN with attention to consider sentence-level information for document classification. On one hand, a document narrative can contain a large sequence of loosely related sentences. Grouping of those sentences into paragraphs or sections is not handled by these models. On the other hand, these models only consider the sentence-level knowledge, but the word-level information that would be diluted in the two-layer model structure can be also important. Therefore, existing approaches of deep document classification cannot appropriately capture the semantics of long-form documents.

2.3 Attention Mechanism in Deep Learning

The attention mechanism [4, 29] has already become one of the most important techniques in deep learning since its huge success in machine translation [44]. Given an input sequence, the attention mechanism infers the importance of each position with a learnable context vector. Besides machine translation, the attention mechanism benefits many applications, such as caption generation [51], document classification [10, 52], and question answering [40]. In this paper, the attention mechanism is exploited to extract the important knowledge in each individual branch of the hierarchical structure for a long-form document.

3 SEMANTIC TEXT MATCHING FOR LONG-FORM DOCUMENTS WITH SMASH RNN

In this section, we first formally define the objective of this paper, and then present the proposed framework, Siamese multi-depth attention-based hierarchical RNN (SMASH RNN), to address the problem of long-form document semantic text matching.

3.1 Problem Statement

In this paper, we focus on long-form documents that can be represented as a hierarchical compositions of word. Note that hierarchy depths for documents can be vary, depending on the highest level of abstraction for a document structure. To facilitate readability, we assume that there are three levels in hierarchy – paragraphs, sentences and words. However, it is important to note that our proposed method is general enough to handle variable depths of hierarchies. Figure 2 gives an illustration of hierarchical structures with different depths for a document d. Words in the d can be fitted into three hierarchical structures, i.e., W^p, W^s, and W^w, with depth 3 (paragraph-level), depth 2 (sentence-level), and depth 1 (word-level) respectively. More precisely, $W^p(k, j, i)$ is the i-th word in the j-th sentence of the k-th paragraph given a paragraph-level hierarchy; $W^s(j, i)$ is the i-th word in the j-th sentence given a sentence-level hierarchy; $W^w(i)$ is the i-th word for the word-level hierarchy with a depth of 1, which is just a long sequence. Note that the bottom level words in the three different hierarchies are exactly

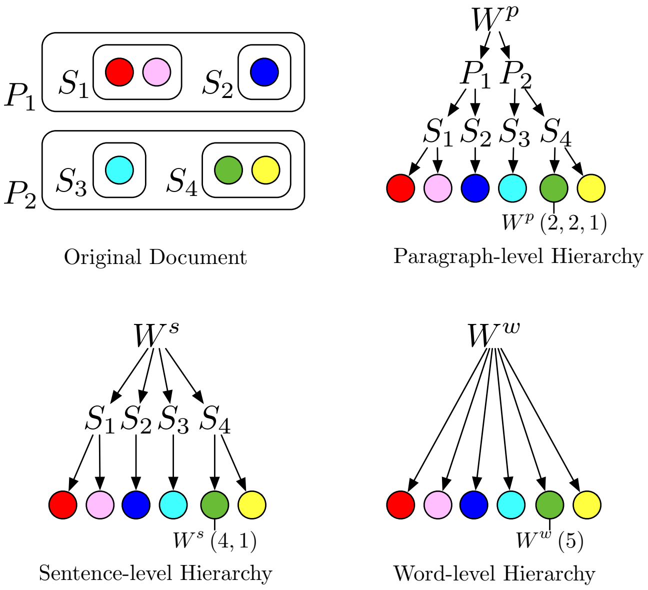

Figure 2: The illustration of hierarchical structures with different depths for an example document. P_k and S_j are the structures of paragraphs and sentences. The positions of a word in the structures depend on the depths of hierarchies.

identical while their annotations vary according to hierarchy depth and document structure.

Given a source document d_s and a set of candidate documents D_c, our goal is to estimate semantic similarity $\hat{y} = Sim(d_s, d_c)$ between the source document d_s and every candidate document $d_c \in D_c$ so that the target documents semantically matched to the source document have higher semantic similarity scores.

3.2 Framework Overview

Figure 3 and 4 illustrate the framework of our proposed Siamese multi-depth attention-based hierarchical RNN (SMASH RNN). Under the Siamese structure [32], each SMASH RNN has two multi-depth attention-based hierarchical RNN (MASH RNN) towers. For each document, MASH RNN derives an informative representation based on the knowledge from different levels of document structure. For each level, an attention-based hierarchical RNN (with corresponding level depth) is constructed as an encoder to generate representations for that level. For example, the paragraph-level encoder produces paragraph-level representations with a depth-3 encoder while the sentence-level encoder produces sentence-level representations with a depth-2 encoder. The final document representation is then acquired by concatenating the representations of different levels, comprehensively covering the knowledge in all document structure levels. To estimate semantic similarity for semantic text matching, SMASH RNN adopts the Siamese structure with two MASH RNN towers. Given representations generated by MASH RNN for both the source and target documents, a fully-connected layer with nonlinearity infers a probabilistic score to examine the semantic relation between two documents with a sigmoid function [34].

In Section 3.3, we formally define MASH RNN. In Section 3.4, we put two MASH RNN towers together to define SMASH RNN.

3.3 MASH RNN for Document Representation

Most of the previous studies only exploit word-level information to represent documents [25, 32, 38, 54]. To investigate deeper structures, some work [10, 52] uses sentence-level information to represent documents. However, only using sentence-level information could lead to a loss of word level information. More than that, a long-form document usually contains quite a number of sentences, and its structure is usually deeper than sentence level. The deeper document structure was not modeled in previous studies for semantic text matching.

In this paper, we propose to model documents with information from different document structure levels. To ease the discussion, we focus on three levels of document structure – paragraph, sentence, and word-level, as mentioned in Section 3.1.

The computation of encoders in MASH RNN follows a bottom-up principle with bidirectional recurrent neural networks (Bi-RNNs) with attention. Take the paragraph-level encoder as an example. Given the j-th sentence in the k-th paragraph in the paragraph-level hierarchy W^p, we first embed words in sentence to vectors through a word embedding layer as follows:

$$\boldsymbol{x}^p_{k,j,i} = emb(W^p(k, j, i)), 1 \leq i \leq L^p_{k,j},$$

where $L^p_{k,j}$ is the length of the sentence, and the word embedding layer $emb(\cdot)$ embeds the words into vectors with an embedding matrix. To encode the sentence, a Bi-RNN reads the sequence of embedding vectors during both the forward pass and the backward pass. In the forward pass, the Bi-RNN creates a sequence of forward hidden states

$$\overrightarrow{\boldsymbol{h}^p_{k,j}} = \left[\overrightarrow{\boldsymbol{h}^p_{k,j,1}}, \overrightarrow{\boldsymbol{h}^p_{k,j,2}}, \cdots, \overrightarrow{\boldsymbol{h}^p_{k,j,L^p_{k,j}}}\right],$$

where $\overrightarrow{\boldsymbol{h}^p_{k,j,i}} = \mathrm{RNN}\left(\overrightarrow{\boldsymbol{h}^p_{k,j,i-1}}, \boldsymbol{x}^p_{k,j,i}\right)$ is generated by a dynamic function such as LSTM [21] or GRU [11]. Here we use GRU instead of LSTM because it requires fewer parameters [24]. The backward pass processes the input sequence in reverse order and generates the backward hidden states as

$$\overleftarrow{\boldsymbol{h}^p_{k,j}} = \left[\overleftarrow{\boldsymbol{h}^p_{k,j,1}}, \overleftarrow{\boldsymbol{h}^p_{k,j,2}}, \cdots, \overleftarrow{\boldsymbol{h}^p_{k,j,L^p_{k,j}}}\right],$$

where $\overleftarrow{\boldsymbol{h}^p_{k,j,i}} = \mathrm{RNN}\left(\overleftarrow{\boldsymbol{h}^p_{k,j,i+1}}, \boldsymbol{x}^p_{k,j,i}\right)$. The forward and backward hidden states are concatenated as a hidden representation for words in the sentence

$$\boldsymbol{h}^p_{k,j} = \left[\boldsymbol{h}^p_{k,j,1}, \boldsymbol{h}^p_{k,j,2}, \cdots, \boldsymbol{h}^p_{k,j,L^p_{k,j}}\right],$$

where $\boldsymbol{h}^p_{k,j,i} = \left[\overrightarrow{\boldsymbol{h}^p_{k,j,i}}; \overleftarrow{\boldsymbol{h}^p_{k,j,i}}\right]$.

Since each word can have unequal importance in the sentence, the attention mechanism [4] is applied to extract and aggregate hidden representations that are more important. More precisely,

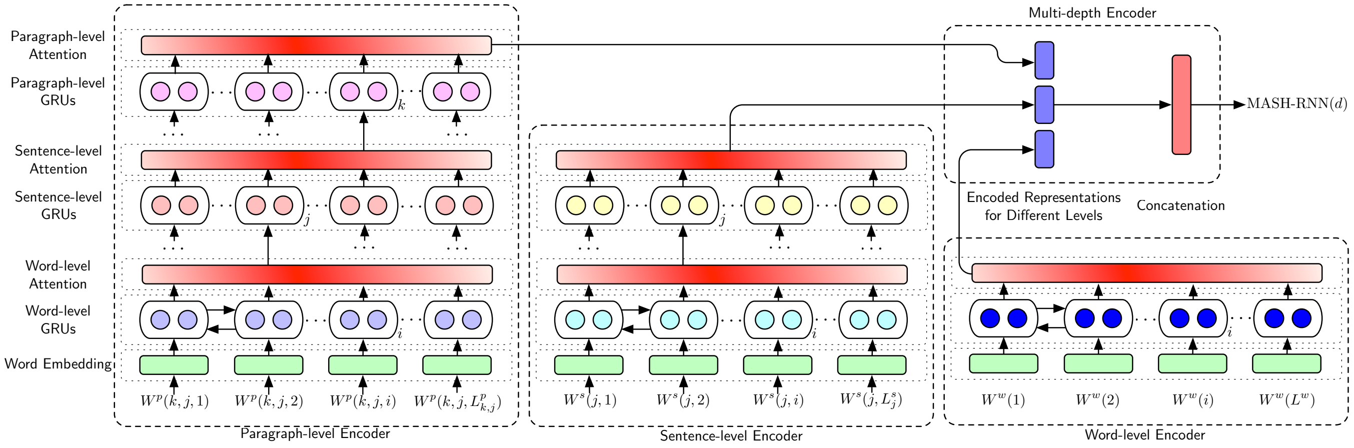

Figure 3: The schema of multi-depth attention-based hierarchical RNN (MASH RNN).

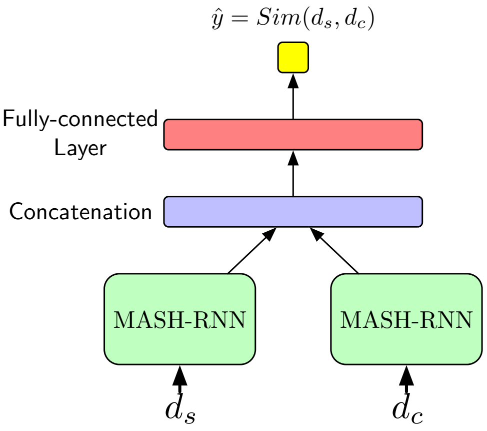

Figure 4: The architecture of Siamese multi-depth attention-based hierarchical RNN (SMASH RNN).

the representation $\boldsymbol{h^p_{k,j,i}}$ will be transformed by a fully-connected hidden layer to measure the importance $\alpha^p_{k,j,i}$ as follows:

$$\alpha^p_{k,j,i} = \frac{\exp\left(\boldsymbol{u^p_{k,j,i}} \cdot \boldsymbol{u^p_w}\right)}{\sum_{i'} \exp\left(\boldsymbol{u^p_{k,j,i'}} \cdot \boldsymbol{u^p_w}\right)},$$

in which $\boldsymbol{u^p_{k,j,i}} = \tanh\left(\mathcal{F}^p_w\left(\boldsymbol{h^p_{k,j,i}}\right)\right)$; $\mathcal{F}^p_w(\cdot)$ is a fully-connected layer; tanh is the activation for the convenience of similarity computation; $\boldsymbol{u^p_w}$ is a vector to measure the word importance. The normalized importance $\alpha^p_{k,j,i}$ can be further obtained through a softmax function. Finally, the representation of the j-th sentence in the k-th paragraph can be represented as the weighted sum of the hidden representations as follows:

$$\boldsymbol{s^p_{k,j}} = \sum_i \alpha^p_{k,j,i} \boldsymbol{h^p_{k,j,i}}.$$

With the representations of sentences, the Bi-RNN is applied again to learn the paragraph representations because a paragraph can be considered as a sequence of sentences. Given the representations of sentences in the k-th paragraph $\boldsymbol{s^p_{k,j}}$, a Bi-RNN generates the forward and backward hidden states of sentences as follows:

$$\overrightarrow{\boldsymbol{h^p_k}} = \left[\overrightarrow{\boldsymbol{h^p_{k,1}}}, \overrightarrow{\boldsymbol{h^p_{k,2}}}, \cdots, \overrightarrow{\boldsymbol{h^p_{k,L^p_k}}}\right], \overleftarrow{\boldsymbol{h^p_k}} = \left[\overleftarrow{\boldsymbol{h^p_{k,1}}}, \overleftarrow{\boldsymbol{h^p_{k,2}}}, \cdots, \overleftarrow{\boldsymbol{h^p_{k,L^p_k}}}\right],$$

where L^p_k is the number of sentences in the k-th paragraph; $\overrightarrow{\boldsymbol{h^p_{k,j}}} = \text{RNN}\left(\overrightarrow{\boldsymbol{h^p_{k,j-1}}}, \boldsymbol{s^p_{k,j}}\right)$; $\overleftarrow{\boldsymbol{h^p_{k,j}}} = \text{RNN}\left(\overleftarrow{\boldsymbol{h^p_{k,j+1}}}, \boldsymbol{s^p_{k,j}}\right)$. The hidden representations for the sentences in the paragraph can then be represented as

$$\boldsymbol{h^p_k} = \left[\boldsymbol{h^p_{k,1}}, \boldsymbol{h^p_{k,2}}, \cdots, \boldsymbol{h^p_{k,L^p_k}}\right],$$

where $\boldsymbol{h^p_{k,j}} = \left[\overrightarrow{\boldsymbol{h^p_{k,j}}}; \overleftarrow{\boldsymbol{h^p_{k,j}}}\right]$. With the attention mechanism, the importance of each sentence can be measured as follows:

$$\alpha^p_{k,j} = \frac{\exp\left(\boldsymbol{u^p_{k,j}} \cdot \boldsymbol{u^p_s}\right)}{\sum_{j'} \exp\left(\boldsymbol{u^p_{k,j'}} \cdot \boldsymbol{u^p_s}\right)},$$

where $\boldsymbol{u^p_{k,j}} = \tanh\left(\mathcal{F}^p_s\left(\boldsymbol{h^p_{k,j}}\right)\right)$; $\mathcal{F}^p_s(\cdot)$ and $\boldsymbol{u^p_s}$ are the fully-connected layer and the vector for the importance measurement. Finally, the representation of the k-th paragraph can be represented as

$$\boldsymbol{p^p_k} = \sum_j \alpha^p_{k,j} \boldsymbol{h^p_{k,j}}.$$

As the top structure in the paragraph-level hierarchy, the document can be treated as a sequence of paragraphs, so we can also utilize the Bi-RNN to generate paragraph-level document representations. Given the representations of paragraphs in the document $\boldsymbol{p^p_k}$, the forward and backward hidden states of paragraphs in the Bi-RNN are generated as follows:

$$\overrightarrow{\boldsymbol{h^p}} = \left[\overrightarrow{\boldsymbol{h^p_1}}, \overrightarrow{\boldsymbol{h^p_2}}, \cdots, \overrightarrow{\boldsymbol{h^p_{L^p}}}\right], \overleftarrow{\boldsymbol{h^p}} = \left[\overleftarrow{\boldsymbol{h^p_1}}, \overleftarrow{\boldsymbol{h^p_2}}, \cdots, \overleftarrow{\boldsymbol{h^p_{L^p}}}\right],$$

where L^p is the number of paragraphs in the document; $\overrightarrow{\boldsymbol{h}_k^p} = \mathrm{RNN}\left(\overrightarrow{\boldsymbol{h}_{k-1}^p}, \boldsymbol{p}_k^p\right)$; $\overleftarrow{\boldsymbol{h}_k^p} = \mathrm{RNN}\left(\overleftarrow{\boldsymbol{h}_{k+1}^p}, \boldsymbol{p}_k^p\right)$. The hidden representations for the paragraphs can be represented as

$$\boldsymbol{h}^p = \left[\boldsymbol{h}_1^p, \boldsymbol{h}_2^p, \cdots, \boldsymbol{h}_{L^p}^p\right],$$

where $\boldsymbol{h}_k^p = \left[\overrightarrow{\boldsymbol{h}_k^p}; \overleftarrow{\boldsymbol{h}_k^p}\right]$. The importance of each paragraph can then be measured with the attention mechanism as follows:

$$\alpha_k^p = \frac{\exp\left(\boldsymbol{u}_k^p \cdot \boldsymbol{u}_p^p\right)}{\sum_{p'} \exp\left(\boldsymbol{u}_{k'}^p \cdot \boldsymbol{u}_p^p\right)},$$

in which $\boldsymbol{u}_k^p = \tanh\left(\mathcal{F}_p^p\left(\boldsymbol{h}_k^p\right)\right)$; $\mathcal{F}_p^p(\cdot)$ and $\boldsymbol{u}_p^p$ are the fully-connected layer and the vector for the importance measurement. The paragraph-level document representation can finally be represented as the output of the paragraph-level encoder as follows:

$$\boldsymbol{d}^p = \sum_k \alpha_k^p \boldsymbol{h}_k^p.$$

Besides the paragraph-level encoder and paragraph-level representation $\boldsymbol{d}^p$, with shallower depths of hierarchy W^s and W^w, we create sentence-level and word-level encoders and generate sentence-level and word-level representations $\boldsymbol{d}^s$ and $\boldsymbol{d}^w$.

In order to capture both concrete low-level observations and abstract high-level insights, MASH RNN uses representations from different document levels. Specifically, MASH RNN returns final document representation as a concatenation of representations generated by multi-depth encoders:

$$\boldsymbol{d} = \left[\boldsymbol{d}^p; \boldsymbol{d}^s; \boldsymbol{d}^w\right].$$

3.4 SMASH RNN for Semantic Text Matching

The Siamese structure associated with two identical sub-networks were shown to be effective in measuring the affinity between representations of two documents modeled in the same hidden space [32, 38, 47, 54]. To address the problem of semantic text matching for long-form documents, we propose the Siamese multi-depth attention-based hierarchical RNN (SMASH RNN) using a Siamese structure that fuses the outputs of two MASH RNNs.

Figure 4 illustrates the structure of SMASH RNN for estimating the semantic similarity between a source document d_s and a candidate document d_c. To tackle two sub-networks, there are several approaches, such as using an attention matrix [54] or a similarity matrix [38]. However, long-form documents require an enormous number of parameters with these methods. Here we propose to utilize the concatenation of representations with a fully-connected layer to learn an appropriate way for computing the semantic similarity. More formally, given $\boldsymbol{d_s}$ and $\boldsymbol{d_c}$, which are the representations generated by MASH RNN for two documents, the final feature vector can be represented as $\boldsymbol{x_f} = [\boldsymbol{d_s}; \boldsymbol{d_c}]$. The semantic similarity between two documents can be computed as follows:

$$\boldsymbol{u_f} = \mathrm{ReLU}\left(\mathcal{F}_d\left(\boldsymbol{x_f}\right)\right),$$

$$\hat{y} = \sigma\left(\mathcal{F}_f\left(\boldsymbol{u_f}\right)\right),$$

where $\mathcal{F}_d(\cdot)$ and $\mathcal{F}_f(\cdot)$ are two fully-connected hidden layers; ReLU$(\cdot)$ is the activation function for the hidden layer; $\sigma(\cdot)$ is the logistic sigmoid function for generating probabilistic scores as similarities [17].

Note that although some of the previous studies [23, 33, 38, 53] suggest to exploit predefined similarity functions, such as the Manhattan distance and the cosine similarity, to directly measure the affinity between representations, we found that predefined similarity functions do not work with long-form documents. This can be because the predefined functions are too simple to estimate the complex semantic relations among long-form documents.

3.5 Learning and Optimization

The task of semantic text matching can be modeled as a binary classification problem. Given a tuple of training data (d_s, d_c, y), where y is a Boolean value showing whether two documents are semantically matched, SMASH RNN optimizes the binary cross-entropy [20] between the estimated probabilistic score $\hat{y}$ and the gold standard y. More formally, the loss function can be written as

$$-(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})).$$

Moreover, the architecture of SMASH RNN allows end-to-end learning [55] that directly trains the full model from scratch using the existing data for a given task. Neither additional data engineering nor multi-step optimization are required.

4 EXPERIMENTS

In this section, we conduct extensive experiments and in-depth analysis with large-scale real-world datasets.

4.1 Experimental Settings

In the experiments, we verify the performance of SMASH RNN in three real-world applications of semantic text matching for long-form documents, including (1) email attachment suggestion, (2) related article recommendation, and (3) citation recommendation. More precisely, these experimental tasks are not only practical but also involving different types of long-form documents for validating the robustness of SMASH RNN. The experimental settings about the datasets employed in each task are described in the following corresponding subsection.

Evaluation. The performance of semantic text matching is mainly evaluated with standard classification metrics, including accuracy, precision, recall, and F1-score [15]. For the task of email attachment suggestion, following prior work [43], we also conduct a ranking experiment, which is evaluated with three standard information retrieval metrics, including precision at 1 (P@1), mean reciprocal rank (MRR), and mean average precision (MAP) [3].

Implementation Details. The model is implemented in Tensorflow [1]. The Adam optimizer [26] is applied to optimize the parameters with an initial learning rate of 10^{-5}. The batch size is set as 32. Note that

a large batch size can lead to enormous memory consumption because long-form documents consists of

Note that because a long-form document can have an enormous amount of words, the size of a batch cannot be so large due to the limitation of memory usage. The number of hidden neurons in

GRUs and hidden layers is set as 128, and the number of dimensions for word embeddings is 256.

Baseline Methods. For all of the tasks, we compare with the following baseline methods with and without using document structures to evaluate the performance of SMASH RNN.

- *RNN-based approach* (RNN) [32, 33] exploits an RNN to model each document as a word sequence. The Siamese structure is then applied to measure the relations between documents. It is the representative of the approaches based on word-level RNNs.
- *CNN-based approach* (CNN) [25] integrates the word embeddings into an embedding matrix and applies several convolutional filters to extract representative features with a pooling layer that covers the whole document. More precisely, a Siamese structure infers the semantic similarity with local features learned by convolutional filters. It can be also treated as the representative of the approaches based on word-level CNNs.
- *CNN with a matching matrix* (DeepQA) [38] enhances the CNN-based approach for question answering. In addition to the convolutional features, DeepQA uses a matching matrix [6] to measure an asymmetric similarity between the features of two documents with a noisy channel [14]. After joining the convolutional features of two documents and their asymmetric similarity, a fully-connected layer generates the semantic similarity. It can also be treated a CNN-based method using word-level information.
- *Hierarchical Attention Network* (HAN) [52] is the only baseline method that considers the document structure information. For each sentence, HAN applies an attention-based RNN to generate a feature vector, thereby deriving the sentence-level document representations with the other RNN. Note that HAN only obtains the sentence-level features for a document. The paragraph-level information is ignored, and the word-level knowledge can be diluted after being passed in the hierarchical model. More specifically, HAN is a special case of MASH RNN using only sentence-level encoders.

For the Siamese structure, although many studies use similarity functions to combine features, such as Manhattan distance [32] and cosine similarity [33], we found all of these methods perform worse than the methods using concatenation with a hidden layer for measuring the relations between long-form documents. Hence, we simply modify the baseline models by applying the Siamese structure shown in Section 3.4 instead of similarity functions to aggregate two document representations. Note that we do not compare with previous works incapable of processing long-form documents, such as ABCNN [54], DRMM [16], and BERT [12]. ABCNN learns an attention matrix for arbitrary position mappings, which is too large to be fitted in memory for long-form documents. DRNN requires an enormous amount of both memory and computational time to compute the local interaction between every words in the source document and the target document. BERT is also memory-inefficient and time-consuming for long-form documents, and its pretraining can only handle at most 512 tokens for a document.

For simplicity, our proposed framework SMASH RNN is denoted as SMASH in experimental results. In addition, P, S, and W in the tables indicate paragraph-, sentence-, and word-level hierarchies utilized in MASH RNN.

Table 1: The statistics of examples in the training, validation, and testing datasets for email attachment suggestion.

Dataset	Training	Validation	Testing
Period of Emails	18 months	2 months	1 month
Number of Examples	49,102	6,950	3,650

Table 2: The classification performance of email attachment suggestion. All improvements of our methods denoted as (*) are significant differences over the HAN method at 99% level in both of a paired t-test and a permutation test.

Method	Accuracy	Precision	Recall	F1
RNN [32, 33]	0.5594	0.5772	0.4439	0.5018
CNN [25]	0.5612	0.5694	0.5024	0.5338
DeepQA [38]	0.5618	0.5990	0.3740	0.4605
HAN [52]	0.5900	0.6096	0.5003	0.5496
SMASH (P)	0.5878	0.5895	0.5783*	0.5839*
SMASH (P+S)	0.6363*	0.6225*	0.6927*	0.6557*
SMASH (P+S+W)	**0.6718***	**0.6440***	**0.7686***	**0.7008***

4.2 Task 1: Email Attachment Suggestion

The first application of semantic text matching for long-form documents in the experiments is email attachment suggestion. Given the content of an email and a candidate document, the goal of email attachment suggestion to classify whether the document should be an attachment for the email. If the system can precisely discriminate the attachments, the system will be able to automatically suggest those attachments, thereby saving users from spending lots of time on searching and attaching documents. This task is also evaluated as a ranking problem that aims to rank candidate documents for a given email.

Experimental Datasets. We adopt the largest publicly available Avocado Research Email Collection [36] as the experimental dataset for email attachment suggestion. The dataset is a collection of emails and attachments taken from a defunct information technology company. The emails sent within the densest 21-month period from January 2000 to September 2001 are selected for experiments. To partition the emails into training, validation, and testing sets, the first 18-month emails are the training data while the following 2-month emails are utilized for validation. The remaining 1-month emails are treated as the testing set. For attachments, we simply remove non-natural language attachments such as programming code or electronic business cards, to construct a pool of candidate attachments. For each pair of an email and its attachment, we extract the pair as a positive example. For each positive example, we randomly sample an irrelevant document from the candidate pool as a negative example to form a balanced dataset. Finally, there are 26,589 attachments in the pool of candidate attachments after filtering non-retrievable files. Table 1 shows the statistics of examples in the experimental datasets for email attachment suggestion.

Experimental Results. Table 2 shows the classification performance of the baseline methods and the proposed SMASH method with combinations of hierarchies used in MASH RNN.

Table 3: The ranking performance of email attachment suggestion. All improvements of our methods denoted as (*) are significant differences over the HAN method at 99% level in both of a paired t-test and a permutation test.

Method	P@1	MRR	MAP
RNN [32, 33]	0.1571	0.3557	0.3515
CNN [25]	0.1451	0.3499	0.3475
DeepQA [38]	0.1774	0.3634	0.3567
HAN [52]	0.1827	0.3724	0.3658
SMASH (P)	0.1534	0.3653	0.3638
SMASH (P+S)	0.2444*	0.4537*	0.4480*
SMASH (P+S+W)	**0.2692***	**0.4900***	**0.4845***

For the baseline methods, the methods using only the word-level knowledge, i.e., RNN, CNN, and DeepQA, have similar performance. HAN surpasses all of the other baseline methods because it considers the sentence-level knowledge. As the proposed method in this paper, SMASH significantly outperforms all of the baseline methods. More precisely, SMASH using all-level knowledge achieves 13.86% and 27.51% improvements against HAN in the metrics of accuracy and F1-score. While using only the paragraph-level hierarchy, SMASH (P) has a similar accuracy and a better F1-score compared to HAN. After accordingly adding sentence- and word-level knowledge, SMASH (P+S) and SMASH (P+S+W) have further improved performance. It demonstrates that all of the hierarchies in different levels are beneficial for email attachment suggestion. In addition, each of the hierarchies holds different information, so they can jointly enhance SMASH.

Aside from classification, the predicted semantic similarity as a numeric value can be also utilized for ranking candidate attachments. For each email, we randomly sample irrelevant documents to construct a list with ten candidate attachments. Table 3 demonstrates the ranking performance of different methods. The experimental results are consistent with the classification experiments. As a result, SMASH achieves 47.35% and 32.45% improvements of P@1 and MAP against HAN. It indicates that the semantic similarities generated by SMASH not only discriminate the attachments but are also predictive of their relative relevance scores.

4.3 Task 2: Related Article Recommendation

The second application in the experiments is related article recommendation. Given a pair of long-form articles, the goal of related article recommendation is to classify if the target article is relevant to the source article. This application can contribute to many real-world scenarios. For example, when a user reads a long Wikipedia page, the system can automatically push other related pages for a more comprehensive coverage of the topic; it can be also helpful for recommending news articles about related events.

Experimental Datasets. The Wikipedia [48] is adopted for the experiments on related article recommendation. 10% of the entity pages in Wikipedia are randomly sampled to build the corpus. Since there is no gold standard about the relatedness of Wikipedia articles, we use links as a source of weak supervision. First, we assume that similar articles have similar sets of outgoing links. Based on this assumption, the Jaccard similarity [22] between the outgoing links of two articles measures their *pseudo similarity*. The article pairs with pseudo similarities greater than a threshold 0.5 are considered as the positive examples. Moreover, we define the article that has the lexicographically smaller URL as the source article of the pair, for the convenience of partitioning datasets and avoiding duplicate examples. For each positive example, we randomly sample a mismatched article from the outgoing links of the source article to generate a negative example. Note that the mismatched articles are not sampled from the entire corpus because those pages will be too irrelevant to make the task challenging enough for differentiating the performance of the various methods. The examples with 80% of the source articles are the training set when each of the validation and testing sets is generated by 10% of the articles. Table 4 shows the statistics of the experimental datasets for related article recommendation.

Table 4: The statistics of examples in the training, validation, and testing datasets for related article recommendation.

Dataset	Training	Validation	Testing
% of Source Articles	80%	10%	10%
Number of Examples	65,948	8,166	8,130

Table 5: The classification performance of related article recommendation. All improvements of our methods denoted as (*) are significant differences over the HAN method at 99% level in both of a paired t-test and a permutation test.

Method	Accuracy	Precision	Recall	F1
RNN [32, 33]	0.7430	0.7328	0.7647	0.7484
CNN [25]	0.6714	0.7204	0.5601	0.6302
DeepQA [38]	0.7366	0.7197	0.7749	0.7463
HAN [52]	0.8089	0.7521	**0.9234**	0.8290
SMASH (P)	0.8047	0.7499	0.9120	0.8230
SMASH (P+S)	**0.8219***	**0.7987***	0.8677	**0.8318***
SMASH (P+S+W)	0.8144*	0.7626*	0.9137	0.8313*

Experimental Results. Table 5 demonstrates the performance of the different methods for related article recommendation. CNN-based methods, i.e., CNN and DeepQA, perform worse than other methods using RNNs. It may be due to the fact that Wikipedia articles are more structural so that convolutional features and the pooling layer cannot capture the document organization. RNN is slightly better than DeepQA since it captures the word dependencies in documents. HAN that considers the sentence-level structure is still the best baseline method for the task. Similar to the results of the previous task, the performance of HAN and SMASH with only the paragraph-level hierarchy is similar. When SMASH considers multi-level knowledge, it outperforms all of the baseline methods. It is also worth mentioning that although SMASH (P+S+W) is better than HAN, it does not improve SMASH (P+S) by adding the word-level hierarchy. It indicates the limitations of word-level information for capturing the semantics of long articles in Wikipedia.

Table 6: The statistics of examples in the training, validation, and testing datasets for citation recommendation.

Dataset	Training	Validation	Testing
% of Source Papers	80%	10%	10%
Number of Examples	169,346	20,742	20,358

Table 7: The performance of citation recommendation. All improvements of our methods against the HAN method are significant differences at 99% level in both of a paired t-test and a permutation test.

Method	Accuracy	Precision	Recall	F1
RNN [32, 33]	0.7352	0.7358	0.7339	0.7348
CNN [25]	0.7309	**0.8048**	0.6097	0.6938
DeepQA [38]	0.7410	0.7579	0.7084	0.7323
HAN [52]	0.7813	0.7454	**0.8544**	0.7962
SMASH (P)	0.7739	0.7532*	0.8149	0.7828
SMASH (P+S)	**0.8068***	0.8019*	0.8150	**0.8084***
SMASH (P+S+W)	0.8058*	0.8038*	0.8092	0.8065*

4.4 Task 3: Citation Recommendation

Last but not least, the third application in the experiments is citation recommendation for academic papers. Given the content of an academic paper and the other paper as a candidate citation, we aim to classify if the candidate should be cited by the paper. This application can benefit researchers in exploring relevant studies for the paper. It not only accelerates the process of doing research and paper-writing but also prevents missing related works.

Experimental Datasets. Here we adopt the AAN Anthology Network Corpus [37] for the experiments. The corpus consists of the descriptions of 23,766 papers written by 18,862 authors in 373 venues and 124,857 citations in a citation network. In addition, the text of some papers are available in the corpus. For each paper with available text, the paper and each of its citations with text in the corpus is treated as a positive example. For each positive example, a irrelevant paper is randomly sampled to create a negative example for constructing a balanced dataset. To prevent the leakage of ground truth, the *References* sections are manually removed. We also remove the abstract sections to increase the difficulty of the task. The dataset is partitioned by the source papers of examples. The training set includes the examples generated by 80% of the source papers when each of the validation and testing sets is generated by 10% of the source papers, respectively. Table 6 demonstrates the statistics of the experimental datasets for citation recommendation.

Experimental Results. Table 7 shows the performance of the methods for citation recommendation. The experimental results are consistent with the results of the previous two tasks. Three baseline methods with only word-level knowledge have similar performance; CNN without using the asymmetric similarity matrix and considering word dependencies performs slightly worse than the other methods. HAN with the sentence-level information still outperforms all of the other baseline methods. After exploiting the knowledge of multi-level hierarchies, SMASH surpasses all of the baseline methods. Similar to the results for related article recommendation, the word-level hierarchy does not enhance the performance after considering the paragraph- and sentence-level knowledge. It depicts that high-level structural information is much more important for the task of citation recommendation.

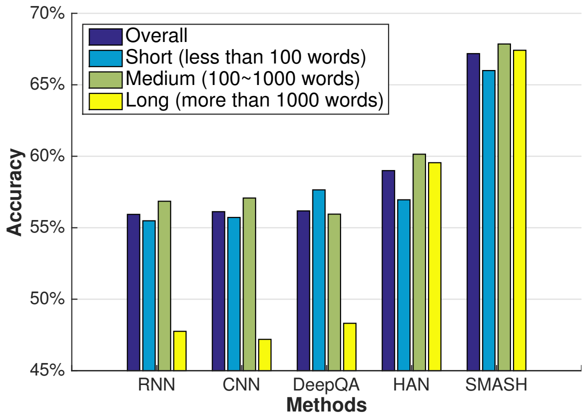

Figure 5: The classification performance of different methods across different document lengths in email attachment suggestion. SMASH uses all-level hierarchies.

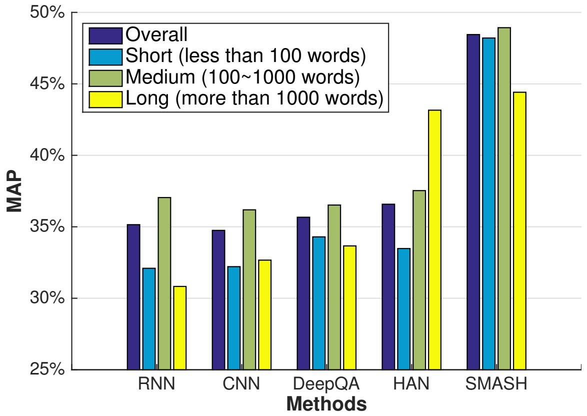

Figure 6: The ranking performance of different methods across different document lengths in email attachment suggestion. SMASH uses all-level hierarchies.

4.5 Robustness Analysis and Discussion

After evaluating the effectiveness of SMASH RNN in different real-world applications, we analyze its robustness in this section.

4.5.1 Document length analysis. First, we verify that the improvements achieved by SMASH are consistent across different lengths of documents with the task of email attachment suggestion. Here we simply categorize document lengths into three groups, including short (less than 100 words), medium (100 to 1,000 words), and long (more than 1,000 words). For each testing example, we classify the example into the three length groups based on the maximum of the length of the source and candidate documents. Figure 5 and 6 show the classification and ranking performance of different methods for

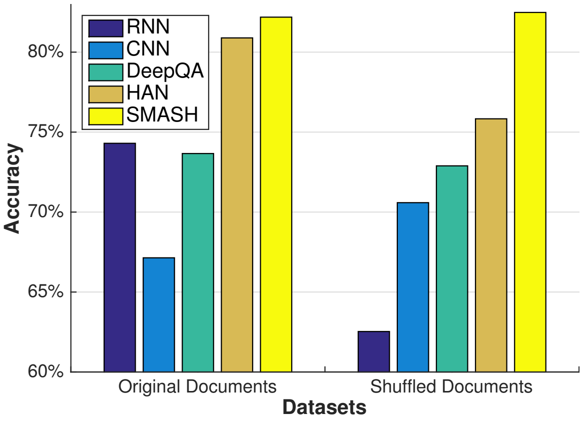

Figure 7: The classification performance of different methods with the original and shuffled documents in related article recommendation. SMASH uses the paragraph- and sentence-level hierarchies.

email attachment suggestion. Although HAN has a similar performance to other baseline methods for short and medium documents, it significantly improves both accuracy and MAP for the long-form documents by taking the sentence-level knowledge into account. It demonstrates the importance of understanding the document structure to deal with long-form documents. The improvements of SMASH against the baseline methods, including HAN, are significant and consistent across different document lengths. This demonstrates that hierarchies utilized in SMASH faithfully capture the document structure and semantics through multiple levels of abstraction. Hence, SMASH is robust across the spectrum of document lengths, and is capable of modeling both short as well as long-form documents.

4.5.2 Robustness to document perturbation. An interesting observation is that although SMASH achieves a 13.86% improvement of accuracy against HAN for email attachment suggestion, the improvements for the other two tasks, while statistically significant, are much smaller (1.6% and 3.1%, respectively). We assume this is because the opening words in Wikipedia pages (descriptions) and academic papers (introductions) are highly informative, and therefore the baselines can solve the problem using only positional information. To verify the assumption, we shuffle the paragraphs of Wikipedia articles to make the important texts randomly distributed in the documents for related article recommendation. Figure 7 shows the performance for related article recommendation with original and shuffled documents. RNN and HAN perform worse with shuffled documents because they cannot identify the critical parts in different positions. CNN and DeepQA have consistent performances because the convolutional features are independent to positional information. These improvements are also consistent in several trails with different random seeds for shuffling documents.

The performance of SMASH is also consistent because the model diagnoses the documents at different levels with attention. Moreover, the improvement against the best baseline method, HAN, increases significantly for the shuffled documents, and is more in line with the improvements achieved for the email attachment suggestion task. This leads us to conclude that SMASH is robust to the positioning of the core document information, and can perform equally well both for documents with well-defined introductions, as well as for complex, non-linear narratives.

5 CONCLUSIONS

In this paper, we propose to address the problem of semantic text matching for long-form documents, which has been less explored in the previous studies. In order to model the complex semantics of long-form documents, MASH RNN is introduced to generate document representations with hierarchical structures at different levels, as well as the attention mechanism in deep learning. Our model, SMASH RNN, is then formulated as a Siamese structure that aggregates the representations of the source and candidate documents derived by two MASH RNNs.

In addition to formulating the theoretical framework, we also demonstrate the practical potential of semantic text matching for long-form documents by providing three real-world applications, including email attachment suggestion, related article recommendation, and citation recommendation. Extensive experiments demonstrate that our proposed approach significantly outperforms several competitive baseline methods within different categories in both of the classification and ranking tasks. Moreover, we also show the robustness of SMASH RNN across the spectrum of documents lengths and perturbations.

Our conclusions can be summarized as follows: (1) semantic text matching for long-form documents is impactful, with numerous useful applications; (2) the usage of hierarchical document structure is essential for semantic text matching, especially for modeling long-form documents; (3) SMASH RNN can accurately capture the complicated semantics of long-form documents, even if the important messages may occur at any position, and at any level of the document structure.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16)*. USENIX Association, 265–283.

[2] Hadi Amiri, Philip Resnik, Jordan Boyd-Graber, and Hal Daumé III. 2016. Learning text pair similarity with context-sensitive autoencoders. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, Vol. 1. ACL, 1882–1892.

[3] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*. Vol. 463. ACM.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3, Jan (2003), 993–1022.

[6] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 165–180.

[7] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems* 30, 1-7 (1998), 107–117.

[8] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems (NIPS'94)*. 737–744.

[9] Michael Busch, Krishna Gade, Brian Larson, Patrick Lok, Samuel Luckenbill, and Jimmy Lin. 2012. Earlybird: Real-time search at twitter. In *Proceedings of 2012 IEEE 28th International Conference on Data Engineering (ICDE'12)*. IEEE, 1360–1369.

[10] Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. 1650–1659.

[11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018).

[13] Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING'14)*. ACL, 69–78.

[14] Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistic (ACL'03)*. ACL, 16–23.

[15] George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research* 3, Mar (2003), 1289–1305.

[16] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM'16)*. ACM, 55–64.

[17] Jun Han and Claudio Moraga. 1995. The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning. In *Proceedings of the international workshop on artificial neural networks: From natural to artificial neural computation*. Springer-Verlag, 195–201.

[18] Taher H Haveliwala, Aristides Gionis, Dan Klein, and Piotr Indyk. 2002. Evaluating strategies for similarity search on the web. In *Proceedings of the 11th International Conference on World Wide Web (WWW'02)*. ACM, 432–442.

[19] Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*. ACL, 1576–1586.

[20] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[22] Paul Jaccard. 1912. The distribution of the flora in the alpine zone. 1. *New phytologist* 11, 2 (1912), 37–50.

[23] Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen, and Wei Wang. 2018. Learning to disentangle interleaved conversational threads with a Siamese hierarchical network and similarity ranking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18)*. ACL, 1812–1822.

[24] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML '15)*. 2342–2350.

[25] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. ACL, 1746–1751.

[26] Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: Amethod for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.

[27] Bang Liu, Ting Zhang, Fred X Han, Di Niu, Kunfeng Lai, and Yu Xu. 2018. Matching Natural Language Sentences with Hierarchical Sentence Factorization. In *Proceedings of the 2018 World Wide Web Conference (WWW'18)*. ACM, 1237–1246.

[28] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, 2873–2879.

[29] Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*. ACL, 1412–1421.

[30] Prem Melville, Wojciech Gryc, and Richard D Lawrence. 2009. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*. ACM, 1275–1284.

[31] Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the Twentieth AAAI Conference on Artificial Intelligence (AAAI'06)*, Vol. 6. AAAI Press, 775–780.

[32] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. AAAI Press, 2786–2792.

[33] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. 148–157.

[34] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*. 625–632.

[35] Geoffrey Nunberg. 1990. *The linguistics of punctuation*. Number 18. Center for the Study of Language (CSLI).

[36] Douglas Oard, William Webber, David Kirsch, and Sergey Golitsynskiy. 2015. Avocado research email collection. *Philadelphia: Linguistic Data Consortium* (2015).

[37] Dragomir R. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The ACL anthology network corpus. *Language Resources and Evaluation* (2013), 1–26. https://doi.org/10.1007/s10579-012-9211-2

[38] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15)*. ACM, 373–382.

[39] Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15)*. ACM, 959–962.

[40] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NIPS'15)*. 2440–2448.

[41] Jaime Teevan, Daniel Ramage, and Merredith Ringel Morris. 2011. # TwitterSearch: a comparison of microblog search and web search. In *Proceedings of the Fourth ACM International Conference on Web search and Data Mining (WSDM'11)*. ACM, 35–44.

[42] George Tsatsaronis, Iraklis Varlamis, and Michalis Vazirgiannis. 2010. Text relatedness based on a word thesaurus. *Journal of Artificial Intelligence Research* 37 (2010), 1–39.

[43] Christophe Van Gysel, Bhaskar Mitra, Matteo Venanzi, Roy Rosemarin, Grzegorz Kukla, Piotr Grudzien, and Nicola Cancedda. 2017. Reply with: Proactive recommendation of email attachments. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM'17)*. ACM, 327–336.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS'17)*. 5998–6008.

[45] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations.. In *AAAI*, Vol. 16. AAAI Press, 2835–2841.

[46] Chenglong Wang, Feijun Jiang, and Hongxia Yang. 2017. A hybrid framework for text modeling with convolutional RNN. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'17)*. ACM, 2061–2069.

[47] Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. (2017).

[48] Wikipedia. 2001. Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/

[49] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems* 26, 3 (2008), 13.

[50] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 55–64.

[51] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML'15)*. 2048–2057.

[52] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'16)*. ACL, 1480–1489.

[53] Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning (CoNLL'11)*. ACL, 247–256.

[54] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* 4 (2016), 259–272.

[55] Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710* (2015).