Bridging the Gap Between Relevance Matching and Semantic Matching for Short Text Similarity Modeling

Jinfeng Rao,[1] **Linqing Liu**,[2] **Yi Tay**,[3] **Wei Yang**,[2] **Peng Shi**,[2] and **Jimmy Lin**[2]
[1] Facebook Assistant
[2] David R. Cheriton School of Computer Science, University of Waterloo
[3] Nanyang Technological University
raojinfeng@fb.com

Abstract

A core problem of information retrieval (IR) is *relevance matching*, which is to rank documents by relevance to a user's query. On the other hand, many NLP problems, such as question answering and paraphrase identification, can be considered variants of *semantic matching*, which is to measure the semantic distance between two pieces of short texts. While at a high level both relevance and semantic matching require modeling textual similarity, many existing techniques for one cannot be easily adapted to the other. To bridge this gap, we propose a novel model, HCAN (Hybrid Co-Attention Network), that comprises (1) a hybrid encoder module that includes ConvNet-based and LSTM-based encoders, (2) a relevance matching module that measures soft term matches with importance weighting at multiple granularities, and (3) a semantic matching module with co-attention mechanisms that capture context-aware semantic relatedness. Evaluations on multiple IR and NLP benchmarks demonstrate state-of-the-art effectiveness compared to approaches that do not exploit pretraining on external data. Extensive ablation studies suggest that relevance and semantic matching signals are complementary across many problem settings, regardless of the choice of underlying encoders.

1 Introduction

Neural networks have achieved great success in many NLP tasks, such as question answering (Rao et al., 2016; Chen et al., 2017a), paraphrase detection (Wang et al., 2017), and textual semantic similarity modeling (He and Lin, 2016). Many of these tasks can be treated as variants of a *semantic matching (SM)* problem, where two pieces of texts are jointly modeled through distributed representations for similarity learning. Various neural network architectures, e.g., Siamese networks (He et al., 2016) and attention (Seo et al., 2017; Tay et al., 2019b), have been proposed to model semantic similarity using diverse techniques.

A core problem of information retrieval (IR) is *relevance matching (RM)*, where the goal is to rank documents by relevance to a user's query. Though at a high level semantic and relevance matching both require modeling similarities in pairs of texts, there are fundamental differences. Semantic matching emphasizes "meaning" correspondences by exploiting lexical information (e.g., words, phrases, entities) and compositional structures (e.g., dependency trees), while relevance matching focuses on keyword matching. It has been observed that existing approaches for textual similarity modeling in NLP can produce poor results for IR tasks (Guo et al., 2016), and vice versa (Htut et al., 2018).

Specifically, Guo et al. (2016) point out three distinguishing characteristics of relevance matching: exact match signals, query term importance, and diverse matching requirements. In particular, exact match signals play a critical role in relevance matching, more so than the role of term matching in, for example, paraphrase detection. Furthermore, in document ranking there is an asymmetry between queries and documents in terms of length and the richness of signals that can be extracted; thus, symmetric models such as Siamese architectures may not be entirely appropriate.

To better demonstrate these differences, we present examples from relevance and semantic matching tasks in Table 1. Column 'Label' denotes whether sentence A and B are relevant or duplicate. The first example from tweet search shares many common keywords and is identified as relevant, while the second pair from Quora shares all words except for the subject and is *not* considered a duplicate pair. An approach based on keyword matching alone is unlikely to be able to distinguish

Task	Label	Sentence A	Sentence B
Tweet Search	1	2022 FIFA soccer	2022 world cup FIFA could be held at the end of year in Qatar
Duplicate Detection	0	Does RBI send its employees for higher education, like MBA?	Does EY send its employees for higher education, like MBA?
Question Answering	1	What was the monetary value of the Nobel peace prize in 1989 ?	Each Nobel prize is worth $469,000 .

Table 1: Sample sentence pairs from TREC Microblog 2013, Quora, and TrecQA.

between these cases. In contrast, the third example is judged as a relevant QA pair because different terms convey similar semantics.

These divergences motivate different architectural choices. Since relevance matching is fundamentally a matching task, most recent neural architectures, such as DRMM (Guo et al., 2016) and Co-PACRR (Hui et al., 2018), adopt an interaction-based design. They operate directly on the similarity matrix obtained from products of query and document embeddings and build sophisticated modules on top to capture additional n-gram matching and term importance signals. On the other hand, many NLP problems, such as question answering and textual similarity measurement, require more semantic understanding and contextual reasoning rather than specific term matches. Context-aware representation learning, such as co-attention methods (Seo et al., 2017), has been proved effective in many benchmarks. Though improvements have been shown from adding exact match signals into representation learning, for example, the Dr.QA model of Chen et al. (2017a) concatenates exact match scores to word embeddings, it remains unclear to what extent relevance matching signals can further improve models primarily designed for semantic matching.

To this end, we examine two research questions: (1) Can existing approaches to relevance matching and semantic matching be easily adapted to the other? (2) Are signals from relevance and semantic matching complementary? We present a novel neural ranking approach to jointly model both the relevance matching process and the semantic matching process. Our model, **HCAN** (**H**ybrid **C**o-**A**ttention **N**etwork), comprises three major components:

1. A hybrid encoder module that explores three types of encoders: *deep*, *wide*, and *contextual*, to obtain contextual sentence representations.
2. A relevance matching module that measures soft term matches with term weightings between pairs of texts, starting from word-level to phrase-level, and finally to sentence-level.
3. A semantic matching module with co-attention mechanisms applied at each encoder layer to enable context-aware representation learning at multiple semantic levels.

Finally, all relevance and semantic matching signals are integrated using a fully-connected layer to yield the final classification score.

Contributions. We see our work as making the following contributions:

- We highlight and systematically explore important differences between relevance matching and semantic matching on short texts, which lie at the core of many of IR and NLP problems.
- We propose a novel model, HCAN (Hybrid Co-Attention Network), to combine best practices in neural modeling for both relevance and semantic matching.
- Evaluations on multiple IR and NLP tasks, including answer selection, paraphrase identification, semantic similarity measurement, and tweet search, demonstrate state-of-the-art effectiveness compared to approaches that do not exploit pretraining on external data. Ablation studies show that relevance and semantic matching signals are complementary in many problems, and combining them can be more data efficient.

2 HCAN: Hybrid Co-Attention Network

The overview of our model is shown in Figure 1. It is comprised of three major components: (1) a hybrid encoder module that explores three types of encoders: *deep*, *wide*, and *contextual* (Sec. 2.1);

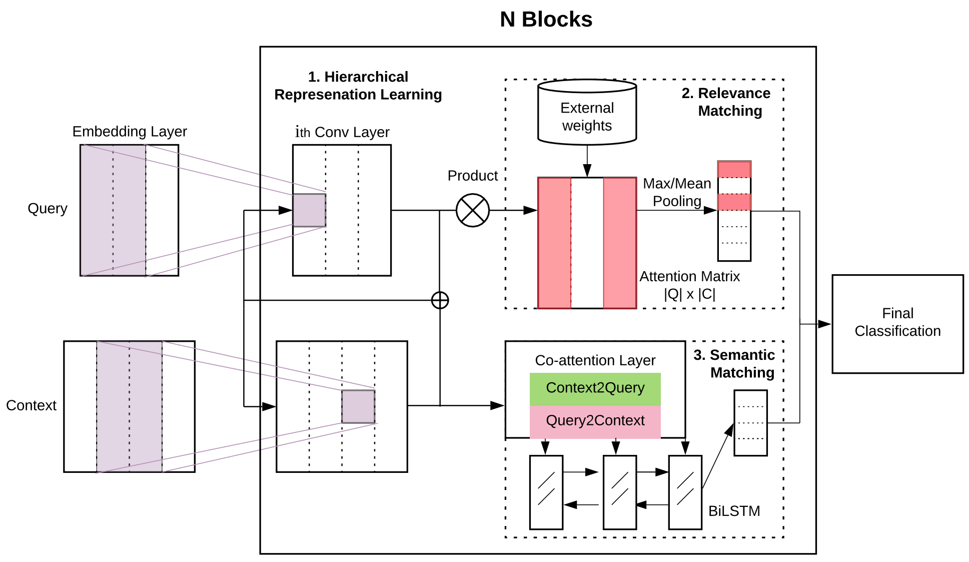


Figure 1: Overview of our Hierarchical Co-Attention Network (HCAN). The model consists of three major components: (1) a hybrid encoder module that explores three types of encoders: *deep*, *wide*, and *contextual*; (2) a relevance matching module with external weights for learning soft term matching signals; (3) a semantic matching module with co-attention mechanisms for context-aware representation learning.

(2) a relevance matching module with external weights for learning soft term matching signals (Sec. 2.2); (3) a semantic matching module with co-attention mechanisms for context-aware representation learning (Sec. 2.3). Note that the relevance and semantic matching modules are applied at each encoder layer, and all signals are finally aggregated for classification.

2.1 Hybrid Encoders

Without loss of generality, we assume that inputs to our model are sentence pairs (q, c), where (q, c) can refer to a (query, document) pair in a search setting, a (question, answer) pair in a QA setting, etc. The query q and context c are denoted by their words, $\{w_1^q, w_2^q, ..., w_n^q\}$ and $\{w_1^c, w_2^c, ..., w_m^c\}$, respectively, where n and m are the number of words in the query and the context. A word embedding layer converts both into their embedding representations $\mathbf{Q} \in \mathbb{R}^{n \times L}$ and $\mathbf{C} \in \mathbb{R}^{m \times L}$, where L is the dimension of the embeddings.

To learn effective phrase-level representations, we explore three different types of encoders: *deep*, *wide*, and *contextual*, detailed below.

Deep Encoder: This design consists of multiple convolutional layers stacked in a hierarchical manner to obtain higher-level k-gram representations. A convolutional layer applies convolutional filters to the text, which is represented by an embedding matrix $\mathbf{U}$ ($\mathbf{Q}$ or $\mathbf{C}$). Each filter is moved through the input embedding incrementally as a sliding window (with window size k) to capture the compositional representation of k neighboring terms. Assuming a convolutional layer has F filters, this CNN layer (with padding) produces an output matrix $\mathbf{U}^o \in \mathbb{R}^{\|U\| \times F}$.

For notational simplicity, we drop the superscript o from all output matrices and add a superscript h to denote the output of the h-th convolutional layer. Stacking N CNN layers therefore corresponds to obtaining the output matrix of the h-th layer $\mathbf{U}^h \in \mathbb{R}^{\|U\| \times F^h}$ via:

$$\mathbf{U}^h = \text{CNN}^h(\mathbf{U}^{h-1}), h = 1, \dots, N,$$

where $\mathbf{U}^{h-1}$ is the output matrix of the $(h-1)$-th convolutional layer. Note that $\mathbf{U}^0 = \mathbf{U}$ denotes the input matrix ($\mathbf{Q}$ or $\mathbf{C}$) obtained directly from the word embedding layer. The parameters of each CNN layer are shared by the query and the context.

Wide Encoder: Unlike the deep encoder that stacks multiple convolutional layers hierarchically, the wide encoder organizes convolutional layers in parallel, with each convolutional layer having a different window size k to obtain the corresponding k-gram representations. Given N convolutional layers, the window sizes of the CNN layers will be in $[k, k+1, ..., k+N-1]$.

Contextual Encoder: Different from both the deep and wide encoders that capture k-gram patterns with convolutions, the contextual encoder leverages Bi-directional LSTMs to extract long-range contextual features. Given N BiLSTM layers, the output at the h-th layer is computed as:

$$\mathbf{U}^h = \text{BiLSTM}^h(\mathbf{U}^{h-1}), h = 1, \ldots, N,$$

The three encoders represent different tradeoffs. The deep and wide encoders are easier for performing inference in parallel and are much faster to train than the contextual encoder. Additionally, the use of CNN layers allows us to explicitly control the window size for phrase modeling, which has been shown to be critical for relevance matching (Dai et al., 2018; Rao et al., 2019). On the other hand, the contextual encoder enables us to obtain long-distance contextual representations for each token. Comparing the deep and wide encoders, the deep encoder saves more parameters by reusing representations from the previous layer. The effectiveness of each encoder is an empirical question we will experimentally answer.

2.2 Relevance Matching

This section describes our efforts to capture keyword matching signals for relevance matching. We calculate the relevance score between the query and the context at each encoder layer by multiplying the query representation matrix $\mathbf{U}_q$ and the context representation matrix $\mathbf{U}_c$:

$$\mathbf{S} = \mathbf{U}_q{\mathbf{U}_c}^T, \mathbf{S} \in \mathbb{R}^{n \times m},$$

where $\mathbf{S}_{i,j}$ can be considered the similarity score by matching the query phrase vector $\mathbf{U}_q[i]$ with the context phrase vector $\mathbf{U}_c[j]$. Since the query and the context share the same encoder layers, similar phrases will be placed closer in a high-dimensional embedding space and their product will produce larger scores. Next, we obtain a normalized similarity matrix $\tilde{\mathbf{S}}$ by applying a *softmax* function over the context columns of $\mathbf{S}$ to normalize the similarity scores into the $[0, 1]$ range.

For each query phrase i, the above *softmax* function normalizes its matching scores over all phrases in the context and helps discriminate matches with higher scores. An exact match will dominate others and contribute a similarity score close to 1.0. We then apply *max* and *mean* pooling to the similarity matrix to obtain discriminative feature vectors:

$$\begin{aligned} \mathit{Max}(\mathbf{S}) &= [\max(\tilde{\mathbf{S}}_{1,:}), ..., \max(\tilde{\mathbf{S}}_{n,:})], \\ \mathit{Mean}(\mathbf{S}) &= [mean(\tilde{\mathbf{S}}_{1,:}), ..., mean(\tilde{\mathbf{S}}_{n,:})], \\ &\mathit{Max}(\mathbf{S}), \mathit{Mean}(\mathbf{S}) \in \mathbb{R}^n \end{aligned}$$

Each score generated from pooling can be viewed as matching evidence for a specific query phrase in the context, where the value denotes the significance of the relevance signal. Compared to *Max* pooling, *Mean* pooling is beneficial for cases where a query phrase is matched to multiple relevant terms in the context.

It's worth noting that term importance modeling can be important for some search tasks (Guo et al., 2016); therefore, we inject external weights as priors to measure the relative importance of different query terms and phrases. We multiply the score after pooling with the weights of that specific query term/phrase. These are provided as feature inputs to the final classification layer, denoted by $\mathbf{O}_{RM}$:

$$\begin{aligned} \mathbf{o}_{RM} = \{wgt(q) \odot \mathit{Max}(\mathbf{S}), wgt(q) \odot \mathit{Mean}(\mathbf{S})\}, \\ \mathbf{O}_{RM} \in 2 \cdot \mathbb{R}^n, \end{aligned} \quad (1)$$

where $\odot$ is an element-wise product between the weights of the query terms/phrases with the pooling scores, and $wgt(q)^i$ denotes the weight of the i-th term/phrase in the query; its value changes in the intermediate encoder layers since deeper/wider encoder layers capture longer phrases. We choose inverse document frequency (IDF) as our weighting function. A higher IDF weight implies a rarer occurrence in the collection and thus greater discriminative power. The weighting method also allows us to reduce the impact of large matching scores for common words like stopwords.

2.3 Semantic Matching

In addition to relevance matching, we aim to capture semantic matching signals via co-attention mechanisms on intermediate query and context representations. Our semantic matching method behaves similarly to the transformer (Vaswani et al., 2017), which also uses attention (specifically, self-attention) over hierarchical blocks to capture semantics at different granularities.

Given $\mathbf{U}_q \in \mathbb{R}^{n \times F}$ and $\mathbf{U}_c \in \mathbb{R}^{m \times F}$ generated by an intermediate encoder layer, we first calculate the bilinear attention as follows:

$$\begin{aligned} \mathbf{A} &= \mathit{REP}(\mathbf{U}_q\mathbf{W}_q) + \mathit{REP}(\mathbf{U}_c\mathbf{W}_c) + \mathbf{U}_q\mathbf{W}_b\mathbf{U}_c^T \\ \mathbf{A} &= \mathit{softmax}_{\text{col}}(\mathbf{A}) \\ \mathbf{A} &\in \mathbb{R}^{n \times m} \end{aligned}$$

where $\mathbf{W}_q, \mathbf{W}_c \in \mathbb{R}^F$, $\mathbf{W}_b \in \mathbb{R}^{F \times F}$, and the *REP* operator converts the input vector to a $\mathbb{R}^{n \times m}$ matrix by repeating elements in the missing dimensions. $\text{Softmax}_{\text{col}}$ is the column-wise softmax operator. Similar to Seo et al. (2017), we perform co-attention from two directions: query-to-context and context-to-query, as follows:

$$\begin{aligned}
&\tilde{\mathbf{U}}_q = \mathbf{A}^T \mathbf{U}_q \\
&\tilde{\mathbf{U}}_c = \mathit{REP}(\max_{\text{col}}(\mathbf{A})\mathbf{U}_c) \\
&\tilde{\mathbf{U}}_q \in \mathbb{R}^{m\times F}, \tilde{\mathbf{U}}_c \in \mathbb{R}^{m\times F}
\end{aligned}$$

where $\max_{\text{col}}$ is the column-wise max-pooling operator. $\tilde{\mathbf{U}}_q$ denotes query-aware context embeddings by attending the raw query representations to the attention weights, while $\tilde{\mathbf{U}}_c$ indicates the weighted sum of the most important words in the context with respect to the query.

We then take an enhanced concatenation to explore the interaction between $\tilde{\mathbf{U}}_q$ and $\tilde{\mathbf{U}}_c$, as in Equation 2. Finally, we apply an additional BiLSTM to the concatenated contextual embeddings $\mathbf{H}$ to capture contextual dependencies in the sequence, and use the last hidden state (with dimension d) as the output features of the semantic matching module $\mathbf{O}_{\mathbf{SM}}$:

$$\begin{aligned}
&\mathbf{H} = [\mathbf{U}_c; \tilde{\mathbf{U}}_q; \mathbf{U}_c \otimes \tilde{\mathbf{U}}_q; \tilde{\mathbf{U}}_c \otimes \tilde{\mathbf{U}}_q] \\
&\mathbf{O}_{\text{SM}} = \text{BiLSTM}(\mathbf{H}) \\
&\mathbf{H} \in \mathbb{R}^{m\times 4F}, \mathbf{O}_{\text{SM}} \in \mathbb{R}^{d}
\end{aligned} \tag{2}$$

2.4 Final Classification

Given the relevance and semantic matching features $\{\mathbf{O}^l_{\text{RM}}, \mathbf{O}^l_{\text{SM}}\}$ (from Equations 1 and 2) learned at each encoder layer l, we concatenate them together and use a two-layer fully-connected layer with ReLU activation to generate the final prediction vector o. During training, we minimize the negative log likelihood loss L summed over all samples (o_i, y_i) below:

$$\begin{aligned}
o =& \mathit{softmax}(\text{MLP}(\{\mathbf{O}^l_{\text{RM}}, \mathbf{O}^l_{\text{SM}}\})), \\
l =& 1, 2, ..., N \quad \text{and} \quad o \in \mathbb{R}^{\|class\|} \\
L =& -\sum_{(o_i, y_i)} \log o_i[y_i],
\end{aligned}$$

where N is the number of encoder layers.

3 Experimental Setup

3.1 Benchmarks and Metrics

We evaluated our proposed HCAN model on three NLP tasks and two IR datasets, as follows:

Answer Selection. This task is to rank candidate answer sentences based on their similarity to the question. We use the TrecQA (Wang et al., 2007) dataset (raw version)[1] with 56k question-answer pairs. We report mean average precision (MAP) and mean reciprocal rank (MRR).

Paraphrase Identification. This task is to identify whether two sentences are paraphrases of each other. We use the TwitterURL (Lan et al., 2017) dataset with 50k sentence pairs. We report the unweighted average of F1 scores on the positive and negative classes (macro-F1).

Semantic Textual Similarity (STS). This task is to measure the degree of semantic equivalence between pairs of texts. We use the Quora (Iyer et al., 2017) dataset with 400k question pairs collected from the Quora website. We report class prediction accuracy.

Tweet Search. This task is to rank candidate tweets by relevance with respect to a short query. We use the TREC Microblog 2013–2014 datasets (Lin and Efron, 2013; Lin et al., 2014), as prepared by Rao et al. (2019), where each dataset contains around 50 queries and 40k query-tweet pairs. We report MAP and precision at rank 30 (P@30).

3.2 Baselines and Implementations

For the answer selection, paraphrase identification, and STS tasks, we compared against the following baselines: InferSent (Conneau et al., 2017), ESIM (Chen et al., 2017b), DecAtt (Parikh et al., 2016), and PWIM (He and Lin, 2016). Additionally, we report state-of-the-arts results on each dataset from published literature. We also include the current state-of-the-art BERT (Devlin et al., 2019) results on each dataset.

For the tweet search task, we mostly follow the experimental setting in Rao et al. (2019). Baselines include the classic query likelihood (QL) method, RM3 query expansion (Abdul-Jaleel et al., 2004), learning to rank (L2R), as well as a number of neural ranking models: DRMM (Guo et al., 2016), DUET (Mitra et al., 2017), K-NRM (Xiong et al., 2017b), and PACRR (Hui et al., 2017). For the neural baselines, we used implementations in MatchZoo.[2] For L2R, we used LambdaMART (Burges, 2010) on the same feature sets as Rao et al. (2019): text-based, URL-

[1]The leaderboard can be found in `https://aclweb.org/aclwiki/Question_Answering_(State_of_the_art)`

[2]`https://github.com/NTMC-Community/MatchZoo`

Model	TrecQA		TwitterURL	Quora
	MAP	MRR	macro-F1	Acc
InferSent	0.521	0.559	0.797	0.866
DecAtt	0.660	0.712	0.785	0.845
ESIM$_{seq}$	0.771	0.795	**0.822**	0.850
ESIM$_{tree}$	0.698	0.734	-	0.755
ESIM$_{seq+tree}$	0.749	0.768	-	0.854
PWIM	0.739	0.795	0.809	0.834
State-of-the-Art Models				
Rao et al. (2016)	**0.780**	0.834	-	-
Gong et al. (2018)	-	-	-	**0.891**
BERT	0.838	0.887	0.852	0.892
Our Approach				
RM	0.756	0.812	0.790	0.842
SM	0.663	0.725	0.708	0.817
HCAN	0.774	**0.843**	0.817	0.853

Table 2: Results on TrecQA, TwitterURL, and Quora. The best scores except for BERT are bolded. In these experiments, all our approaches use the *deep* encoder in Sec. 2.1. RM and SM denote that only relevance and semantic matching signals are used, respectively. HCAN denotes the complete HCAN model.

Model	TREC-2013		TREC-2014	
	MAP	P@30	MAP	P@30
QL	0.2532	0.4450	0.3924	0.6182
RM3	0.2766	0.4733	**0.4480**	0.6339
L2R	0.2477	0.4617	0.3943	0.6200
Neural Baselines				
DUET	0.1380	0.2528	0.2680	0.4091
DRMM	0.2102	0.4061	0.3440	0.5424
K-NRM	0.1750	0.3178	0.3472	0.5388
PACRR	0.2627	0.4872	0.3667	0.5642
BERT	0.3357	0.5656	0.5176	0.7006
Our Approach				
RM	0.2818	0.5222	0.4304	0.6297
SM	0.1365	0.2411	0.2414	0.3279
HCAN	**0.2920**	**0.5328**	0.4365	**0.6485**

Table 3: Results on TREC Microblog 2013–2014, organized in the same manner as Table 2.

based, and hashtag-based. Finally, we include the BERT results from Yang et al. (2019b).

In our experiments, we use trainable 300d word2vec (Mikolov et al., 2013) embeddings with the SGD optimizer. For out-of-vocabulary words, we initialize their word embeddings with a uniform distribution from [0, 0.1]. Since our model also uses external weights, we calculate the IDF values from the training corpus. The number of convolutional layers N is set to 4, and the convolutional filter size k is set to 2. Hidden dimension d is set to 150. We tune the learning rate in [0.05, 0.02, 0.01], the number of convolutional filters F in [128, 256, 512], batch size in [64, 128, 256], and the dropout rate between 0.1 and 0.5. Our code and datasets are publicly available.[3]

4 Results

Our main results on the TrecQA, TwitterURL, and Quora datasets are shown in Table 2 and results on TREC Microblog 2013–2014 are shown in Table 3. The best numbers for each dataset (besides BERT) are bolded. We compare to three variants of our HCAN model: (1) only relevance matching signals (RM), (2) only semantic matching signals (SM), and (3) the complete model (HCAN). In these experiments, we use the *deep* encoder.

From Table 2, we can see that on all three datasets, relevance matching (RM) achieves significantly higher effectiveness than semantic matching (SM). It beats other competitive baselines (InferSent, DecAtt and ESIM) by a large margin on the TrecQA dataset, and is still comparable to those baselines on TwitterURL and Quora. This finding suggests that soft term matching signals alone are fairly effective for many textual similarity modeling tasks. However, SM performs much worse on TrecQA and TwitterURL, while the gap between SM and RM is reduced on Quora. By combining SM and RM signals, we observe consistent effectiveness gains in HCAN across all three datasets, establishing new state-of-the-art (non-BERT) results on TrecQA.

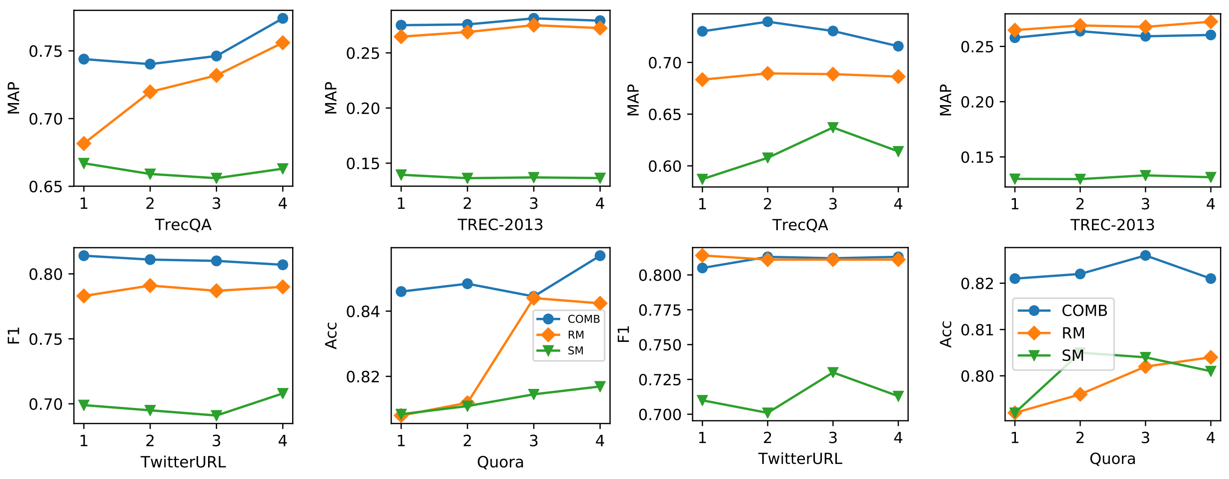
In Table 3, we observe that the query expansion method (RM3) outperforms most of the neural ranking models except for BERT, which is consistent with Yang et al. (2019a). We suggest two reasons: (1) tweets are much shorter and the informal text is “noisier” than longer documents in the web or newswire settings, which are what the previous neural models were designed for; (2) most neural baselines build directly on top of the embedding similarity matrix without any representation learning, which can be less effective.

Comparing our proposed approaches in Table 3, RM achieves fairly good scores while SM is not effective at all, affirming our hypothesis that term matching signals are essential to IR tasks. This finding further supports our motivation for bridging SM and RM. Indeed, semantic matching methods alone are ineffective when queries are comprised of only a few keywords, without much semantic information to exploit. However, the

[3] https://github.com/jinfengr/hcan.git

Encoder	Model	TrecQA		TwitURL	Quora	TREC-2013		TREC-2014	
		MAP	MRR	macro-F1	Acc	MAP	P@30	MAP	P@30
Deep	RM	0.756	0.812	0.790	0.842	0.282	0.522	0.430	0.630
	SM	0.663	0.725	0.708	0.817	0.137	0.241	0.241	0.328
	HCAN	**0.774**	0.843	**0.817**	**0.853**	**0.292**	**0.533**	**0.437**	**0.649**
Wide	RM	0.758	0.806	0.790	0.830	0.278	0.510	0.421	0.617
	SM	0.673	0.727	0.719	0.811	0.138	0.247	0.247	0.336
	HCAN	0.770	**0.847**	0.795	0.843	0.285	0.524	0.435	0.642
Contextual	RM	0.690	0.736	0.811	0.804	0.272	0.503	0.417	0.613
	SM	0.668	0.735	0.730	0.805	0.133	0.256	0.242	0.324
	HCAN	0.739	0.790	0.815	0.826	0.285	0.524	0.434	0.635

Table 4: Evaluation of different encoders in Sec. 2.1 (best numbers on each dataset are bolded).



(a) Deep encoder

(b) Contextual encoder

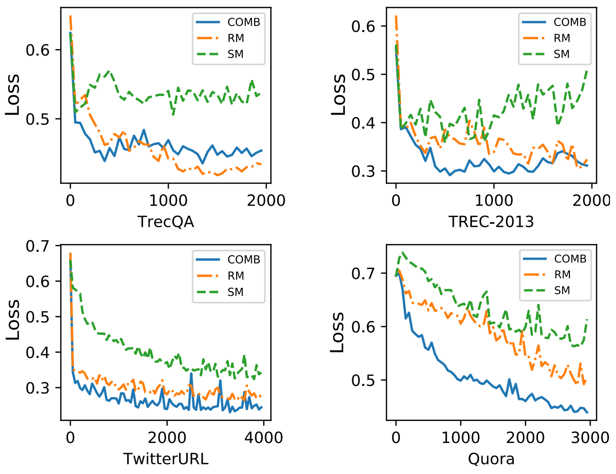
Figure 2: Model effectiveness with different numbers of encoder layers.

context-aware representations learned from SM *do* contribute to RM, leading to the superior results of our complete HCAN model.

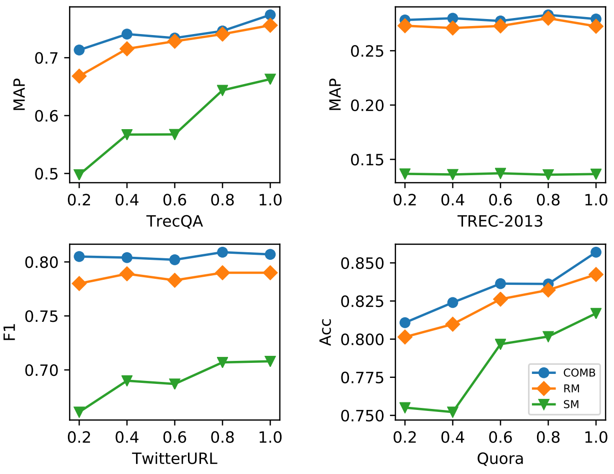
4.1 Encoder Comparisons

We report results with the three different encoders from Sec 2.1 in Table 4. Overall, the effectiveness of the *deep* and *wide* encoders are quite close, given that the two encoders capture the same types of n-gram matching signals. The *contextual* encoder performs worse than the other two on TrecQA, but is comparable on all other datasets. This finding is consistent with Rao et al. (2017a), which shows that keyword matching signals are important for TrecQA. Also, we notice that the gaps between RM and SM are smaller for all encoders on Quora. We suspect that SM is more data-hungry than RM given its larger parameter space (actually, the RM module has no learnable parameters) and Quora is about 10× larger than the other datasets. For all encoders, combing RM and SM consistently improves effectiveness, affirming that relevance and semantic matching signals are complementary regardless of the underlying encoder choice.

To better understand the different encoder mechanisms, we vary the number of encoder layers for the *deep* and *contextual* encoders in Figure 2 (since the *wide* encoder behaves similarly to the *deep* encoder, we omit the analysis here). Our complete HCAN model has $N = 4$. In Figure 2a, we can see overall increases in effectiveness for the RM and HCAN (comb) as N increases, showing that long-range phrase modeling is critical. However, increasing context window lengths don't help SM on the TrecQA and TREC-2013 datasets, likely because of dominant bigram matching signals ($N = 1$). Also, the complete HCAN model is consistently better than SM and RM alone in most settings, affirming its superior effectiveness, consistent with results in the above tables. In contrast, increasing the number of Bi-



(a) Validation losses w.r.t. number of batches.



(b) Effectiveness w.r.t. different percentage of training data.

Figure 3: Experiments exploring learning efficiency.

LSTM layers can sometimes even hurt, as shown in Figure 2b. This is not a surprise since a single BiLSTM layer ($N = 1$) can already capture long-range contextual information and increasing the number of layers can introduce more parameters and lead to overfitting.

4.2 Learning Efficiency

We also designed two experiments to examine whether the complete HCAN model is learning more efficiently than SM or RM alone. In these experiments, we used the *deep* encoder. First, we show validation losses for different numbers of batches in Figure 3a. Second, we vary the training data size by randomly selecting different percentages (from 20% to 100%) of the original training set, shown in Figure 3b.

In Figure 3a, we can see that the validation loss for the complete model drops much faster than RM and SM alone, especially on TwitterURL and Quora. In Figure 3b, we can see that, as expected, all methods in general achieve higher scores when more data are used for training. An exception is SM on the TREC-2013 Twitter dataset, which we see is not effective in Table 3. Another important finding is that both RM and HCAN are more data efficient: for TREC-2013 and TwitterURL, both can achieve effectiveness comparable to the full training set with only 20% data.

4.3 Qualitative Sample Analysis

We present sample outputs in Table 5 to gain more insight into model behavior. For space considerations, we only show the Quora dataset, but our analysis reveals similar findings on the other datasets. The column "label" denotes the label of the sentence pair: 1 means semantically equivalent and 0 means not equivalent. For each model, we output its predicted label along with its confidence score; phrases with large attention weights are highlighted in orange and red.

In the first example, SM is able to correctly identify that the two sentences convey the same meaning with high confidence, while RM fails as the two sentences have no influential phrase matches (with high IDF weights). The sentence pair in the second example has a large text overlap. It is no surprise that RM would predict a high relevance score, while SM fails to capture their relatedness. In both examples, HCAN is able to integrate SM and RM to make correct predictions. Since the third example presents a similar pattern, we omit a detailed explanation. Overall, our quantitative and qualitative analyses show that relevance matching is better at capturing overlap-based signals, while combining semantic matching signals improve representation learning.

5 Related Work

5.1 Neural Relevance Matching

Recently, deep learning has achieved great success in many NLP and IR applications (He and Lin, 2016; Sutskever et al., 2014; Yin et al., 2016; Rao et al., 2017b). Current neural models for IR can be divided into representation-based and interaction-based approaches, discussed below:

Early neural IR models mainly focus on representation-based modeling between the query and documents, such as DSSM (Huang et al.,

Label	SM Score	RM Score	HCAN Score	Sample Pair
1	1, 0.9119	0, 0.9353	1, 0.5496	- How does it feel to kill a human ? - How does it feel to be a murderer ?
1	0, 0.9689	1, 0.8762	1, 0.8481	- What are the time dilation effects on the ISS ? - According to the theory of relativity , time runs slowly under the influence of gravity . Is there any time dilation experienced on the ISS ?
0	0, 0.9927	1, 0.8473	1, 0.7280	- Does RBI send its employees for higher education such as MBA , like sponsoring the education or allowing paid / unpaid leaves ? - Does EY send its employees for higher education such as MBA , like sponsoring the education or allowing paid / unpaid leaves ?

Table 5: Sample pairs from Quora. Phrases with large attention weights are highlighted in orange and red.

2013), C-DSSM (Shen et al., 2014), and SM-CNN (Severyn and Moschitti, 2015). These methods directly learn from query and document representations, and have been found to be ineffective when data is scarce.

Interaction-based approaches build on the similarity matrix computed from word pairs between the query and the document, often with count-based techniques to address data sparsity. For example, DRMM (Guo et al., 2016) introduced a pyramid pooling technique to convert the similarity matrix into histogram representations, on top of which a term gating network aggregates weighted matching signals from different query terms. Inspired by DRMM, Xiong et al. (2017b) proposed K-NRM, which introduced a differentiable kernel-based pooling technique to capture matching signals at different strength levels. Sharing similarities with our architecture is Rao et al. (2019), who developed a multi-perspective relevance matching method with a hierarchical convolutional encoder to capture character-level to sentence-level relevance signals from heterogeneous sources.

5.2 Neural Semantic Matching

Semantic matching is a fundamental problem for a variety of NLP tasks. For example, in paraphrase identification, SM is used to determine whether two sentences or phrases convey the same meaning. In question answering or reading comprehension (Xiong et al., 2017a; Tay et al., 2019a), SM can help identify the correct answer span given a question. Semantic understanding and reasoning for two pieces of texts lie at the core of SM. Existing state-of-the-art techniques for SM usually comprise three major components: (1) sequential sentence encoders that incorporate word context and sentence order for better sentence representations; (2) interaction and attention mechanisms (Tay et al., 2019b; Seo et al., 2017; Parikh et al., 2016; Conneau et al., 2017; Gong et al., 2018) to emphasize salient word pair interactions; (3) structure modeling (Chen et al., 2017b).

6 Conclusion

In this work, we examine the relationship between relevance matching and semantic matching, and highlight a few important differences between them. This is an important problem that lies at the core of many NLP and IR tasks. We propose the HCAN model with a relevance matching module to capture weighted n-gram matching signals and a semantic matching module for context-aware representation learning.

Thorough experiments show that relevance matching alone performs reasonably well for many NLP tasks, while semantic matching alone is not effective for IR tasks. We show that relevance matching and semantic matching are complementary, and HCAN combines the best of both worlds to achieve competitive effectiveness across a large number of tasks, in some cases, achieving the state of the art for models that do not exploit large-scale pretraining. We also find that our model can learn in a data efficient manner, further demonstrating the complementary nature of relevance matching and semantic matching signals.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Donald Metzler, Mark D. Smucker, Trevor Strohman, Howard Turtle, and Courtney Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, Maryland.

Christopher J. C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 126–134, Marina Del Rey, CA, USA.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *Proceedings of the Sixth International Conference on Learning Representations (ICLR 2018)*.

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, pages 55–64, Indianapolis, Indiana, USA.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, California.

Hua He, John Wieting, Kevin Gimpel, Jinfeng Rao, and Jimmy Lin. 2016. UMD-TTIC-UW at SemEval-2016 task 1: Attention-based multi-perspective convolutional neural networks for textual similarity measurement. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1103–1108, San Diego, California.

Phu Mon Htut, Samuel Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 120–127, New Orleans, Louisiana.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2333–2338, San Francisco, California, USA.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 279–287, Marina Del Rey, CA, USA.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. *First Quora Dataset Release: Question Pairs*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark.

Wuwei Lan and Wei Xu. 2018. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3890–3902, Santa Fe, New Mexico, USA.

Jimmy Lin and Miles Efron. 2013. Overview of the TREC-2013 Microblog Track. In *Proceedings of the Twenty-Second Text REtrieval Conference (TREC 2013)*.

Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. 2014. Overview of the TREC-2014 Microblog Track. In *Proceedings of the Twenty-Third Text REtrieval Conference (TREC 2014)*, Gaithersburg, Maryland.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1291–1299, Perth, Australia.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1913–1916, Indianapolis, Indiana, USA.

Jinfeng Rao, Hua He, and Jimmy Lin. 2017a. Experiments with convolutional neural network models for answer selection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1217–1220, Shinjuku, Tokyo, Japan.

Jinfeng Rao, Ferhan Ture, Hua He, Oliver Jojic, and Jimmy Lin. 2017b. Talking to your TV: Context-aware voice search with hierarchical recurrent neural networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 557–566, Singapore.

Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. 2019. Multi-perspective relevance matching with hierarchical ConvNets for social media search. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, pages 232–240, Honolulu, Hawaii.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the Fifth International Conference on Learning Representations (ICLR 2017)*.

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, pages 373–382, Santiago, Chile.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 373–374, Seoul, Korea.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Yi Tay, Shuohang Wang, Anh Tuan Luu, Jie Fu, Minh C. Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. 2019a. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4922–4931, Florence, Italy.

Yi Tay, Aston Zhang, Anh Tuan Luu, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu, and Siu Cheung Hui. 2019b. Lightweight and efficient neural natural language processing with quaternion networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1494–1503, Florence, Italy.

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*, pages 5998–6008.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 4144–4150, Melbourne, Australia.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017a. Dynamic coattention networks for question answering. In *Proceedings of the Fifth International Conference on Learning Representations (ICLR 2017)*.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017b. 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*, SIGIR '17, pages 55–64, Shinjuku, Tokyo, Japan.

Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019a. Critically examining the “neural hype”: Weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR’19, pages 1129–1132, Paris, France.

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

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*, 4:259–272.