AWE: Asymmetric Word Embedding for Textual Entailment

Tengfei Ma[1] **Chiamin Wu**[2] **Cao Xiao**[1] **Jimeng Sun**[2]
[1] IBM Research [2] Georgia Institute of Technology
Tengfei.Ma1@ibm.com, cwu392@gatech.edu, cxiao@us.ibm.com, jsun@cc.gatech.edu

Abstract

Textual entailment is a fundamental task in natural language processing. It refers to the directional relation between text fragments such that the "premise" can infer "hypothesis". In recent years deep learning methods have achieved great success in this task. Many of them have considered the inter-sentence word-word interactions between the premise-hypothesis pairs, however, few of them considered the "asymmetry" of these interactions. Different from paraphrase identification or sentence similarity evaluation, textual entailment is essentially determining a directional (asymmetric) relation between the premise and the hypothesis. In this paper, we propose a simple but effective way to enhance existing textual entailment algorithms by using asymmetric word embeddings. Experimental results on SciTail and SNLI datasets show that the learned asymmetric word embeddings could significantly improve the word-word interaction based textual entailment models. Its noteworthy that the proposed AWE-DeIsTe model can get 2.1% accuracy improvement over prior state-of-the-art on SciTail.

Introduction

Textual entailment is an important task in natural language processing (NLP). It refers to the directional relation between text fragments such that the "premise" can infer "hypothesis". To determine such relations requires semantic understanding of text, thus textual entailment is used to measure natural language understanding and can also be applied to other tasks, such as question-answering (Khot, Sabharwal, and Clark 2018; Chen et al. 2017), information retrieval (Clinchant, Goutte, and Gaussier 2006) and document summarization (Pasunuru, Guo, and Bansal 2017). The objective of textual entailment is to classify a premise-hypothesis (called P-H for simplicity) pair into two classes (entail or non-entail) or three-classes (entail or contradict or neutral). The following is an example from the SNLI dataset (Bowman et al. 2015), labeled as "entailment", which means the premise implies the hypothesis.

Premise: *Four guys in wheelchairs on a basketball court two are trying to grab a basketball in midair.*.

Hypothesis: *Four guys are playing basketball.*

The approach of textual entailment is generally the same as those of other sentence matching tasks, such as semantic similarity evaluation. However, not all sentence matching tasks are created equal. Different from semantic similarity evaluation, the semantic relation between the premise-hypothesis (P-H) pair is asymmetric in a textual entailment task, i.e. Dis(P,H) is different from Dis(H,P). Consider the word-mover-distance (Kusner et al. 2015) as an example, if we measure the semantic distance of the sentences based on symmetric word embeddings, Dis(P,H) will be equal to Dis(H,P) if we use the same word embedding space for both P and H. However this is not true for textual entailment. For example, for the example P-H pair above, if we exchange the roles of premise and hypothesis, the classification result would be different since the latter sentence cannot entail the former sentence. A symmetric distance between P and H cannot faithfully model the logical relation between texts. Therefore it is better to build an entailment model to retain the asymmetric relations between premise and hypothesis.

More recently, word-word interaction attracts much attention especially for the deep learning based textual entailment algorithms (Rocktäschel et al. 2016; Parikh et al. 2016; Yin, Roth, and Schütze 2018). These algorithms encourage reasoning over entailments of word- or phrase- pairs. However, the interactions are mostly measured by the symmetric similarity between words, while the directional entailment (i.e., inference) relation is ignored.

In this paper, we propose to utilize the asymmetric word embeddings to improve word-word interactions in textual entailment models. In particular, we first find the candidate entailment word pairs in the training data, and get asymmetric word embeddings for premise sentences and hypothesis sentences separately. The learned embeddings are then used to get entailment based word-level interaction. They are fed into existing word-word-interaction based textual entailment models (e.g. Decomposable attention model (Parikh et al. 2016) and DeIsTe (Yin, Roth, and Schütze 2018)). By adding the entailment interaction to the similarity-based word-word interactions generated by these models, our approach improved these models significantly as demonstrated on SNLI and SciTail data. Especially, we achieved the state-of-the-art performance on SciTail.

To summarize, our work has the following contributions.

- We propose a new type of asymmetric word embedding which is motivated by the asymmetric relation of entailment inherent in the texts. Unlike existing asymmetric

embedding methods, the proposed approach does not require any external knowledge.

- The asymmetric word embedding generated by our approach are general and transferable. They can be used to improve various word-word interaction based models (as we will show later in DeIsTe(Yin, Roth, and Schütze 2018) and Decomp-Att(Parikh et al. 2016)). Moreover, the extracted entailment word pairs and the derived asymmetric word embeddings can be used for other tasks. Cross-data test accuracies also demonstrated the comparable performance as training and testing on the same data.
- By modifying the word-word interactions using asymmetric word embeddings, the models that use our proposed embeddings (AWE-DeIsTe and AWE-Decomp-Att) gained significant performance improvement over the original models (DeIsTe and Decomp-Att) and state-of-the-art test accuracy on SciTail.

Background

Word Embedding Word embedding is a natural language processing technique that learns continuous vector representations of words, which can be used to represent the semantic similarity of words or support other downstream NLP tasks. These word representations are generally derived from the co-occurrence statistics of a large corpus. For example, the word2vec (Mikolov et al. 2013) is based on the co-occurrence of words in a context window.

In this section, we will first briefly review one of the most popular word embedding methods: continuous skip-gram (Mikolov et al. 2013), which is closely related to our later proposed methods. Given a sequence of training words $w_1, w_2, ...w_T$, the objective is to maximize the following log-likelihood given by Eq. 1.

$$\sum_{t=1}^{T} \sum_{c \in \mathcal{C}_t} \log p(w_c|w_t) \tag{1}$$

where $\mathcal{C}_t$ denotes the indices of words surrounding word w_t. The probability of observing a context word w_c given w_t is given by a softmax function in Eq. 2.

$$p(w_c|w_t) = \frac{exp(\mathbf{v}_{w_c}^T \mathbf{u}_{w_t})}{\sum_{w=1}^{W} exp(\mathbf{v}_w^T \mathbf{u}_{w_t})} \tag{2}$$

where u_w and v_w are the "input" and "output" vectors of a word w. This objective can then be approximately optimized by hierarchical softmax or negative sampling, and we get the vector representations of words.

Word embedding has been a fundamental component for deep learning based textual entailment models. However, most previous works do not differentiate the word embeddings of the premise and the hypothesis, thus the inter-sentence word-word interactions using the word embeddings are usually symmetric. Symmetric similarity-based interactions are not sufficient to represent an entailment relationship. We will introduce our new approach to deriving asymmetric word embeddings for entailment in the next Section.

Decomposable Attention Model (Decomp-Att) The decomposable attention model (Parikh et al. 2016) is one of the most important textual entailment models that regard word-word interaction as a key component. It computes a soft alignment matrix and decomposes the entailment task into comparisons of aligned words. Given two sentences $P = (\mathbf{p}_1, ..., \mathbf{p}_{l_p})$ and $H = (\mathbf{h}_1, ..., \mathbf{h}_{l_h})$, the model first soft-aligns the elements of P and H using a variant of neural attention:

$$e_{i,j} = F'(\mathbf{p}_i, \mathbf{h}_j) = F(\mathbf{p}_i)^T F(\mathbf{h}_j) \tag{3}$$

Using these attention weights, we can get the new (softly) alignment vectors

$$\begin{aligned} \boldsymbol{\beta}_i &:= \sum_{j=1}^{l_h} \frac{\exp(e_{i,j})}{\sum_{k=1}^{l_h} \exp(e_{i,k})} \mathbf{h}_j \\ \boldsymbol{\alpha}_j &:= \sum_{i=1}^{l_p} \frac{\exp(e_{i,j})}{\sum_{k=1}^{l_p} \exp(e_{k,j})} \mathbf{p}_i \end{aligned} \tag{4}$$

where $\boldsymbol{\beta}_i$ is the subphrase in H which is aligned to $\mathbf{p}_i$; while $\boldsymbol{\alpha}_j$ is the subphrase in P that is aligned to $\mathbf{h}_j$.

Then the aligned subphrases (i.e. $\mathbf{p}_i$ and $\boldsymbol{\beta}_i$, $\mathbf{h}_j$ and $\boldsymbol{\alpha}_j$) are compared using a function G and aggregated.

$$\begin{aligned} \mathbf{v}_1 &= \sum_i \mathbf{v}_{1,i} = \sum_i G([\mathbf{p}_i, \boldsymbol{\beta}_i]) \\ \mathbf{v}_2 &= \sum_j \mathbf{v}_{2,j} = \sum_j G([\mathbf{h}_j, \boldsymbol{\alpha}_j]) \end{aligned}$$

The final prediction of entailment is based on $\mathbf{v}_1$ and $\mathbf{v}_2$: $Y = H([\mathbf{v}_1, \mathbf{v}_2])$, where H is an feed forward network.

The Decomp-Att has another optional component, intra-sentence attention, to further enhance the performance. As it is optional and not very related to the motivation of our paper, we omit this component and did not use it in our own models.

DeIsTe Most recently, (Yin, Roth, and Schütze 2018) proposed a new word-word interaction based model and achieved state-of-the-art performance on the SciTail dataset. Given word-word interactions between the premise-hypothesis pair P-H, the model uses a parameter-dynamic convolution to weight important words in P and H and a position-aware attentive convolution to encode the representation and position information of the aligned word pairs.

The word-word interaction in DeIsTe is simply the cosine similarity $\mathbf{I}[i,j] = cosine(\mathbf{p}_i, \mathbf{h}_j)$ between the words in premise and hypothesis. Given interaction I, there are three exploration strategies of these interactions.

1. importance of $\mathbf{p}_i$: $a_i = \frac{1.0}{1.0 + max(\mathbf{I}[i,:])}$
2. soft best match of $\mathbf{p}_i$: $H \cdot \text{softmax}(\mathbf{I}[i,:])$
3. hard location of best match of $\mathbf{p}_i$: $\arg\max \mathbf{I}[i,:]$

The parameter-dynamic convolution is based on the importance of p_i, which measures how important a word is in the convolution encoder for P. Specially, for each adjacent

trigram p_{i-1}, p_i, p_{i+1} in P, the parameter-dynamic convolution learns the representation

$$\mathbf{m}_i = \tanh(a_{i-1}\mathbf{W}^{-1}\mathbf{p}_{i-1}, a_i\mathbf{W}^0\mathbf{p}_i, a_{i+1}\mathbf{W}^{+1}\mathbf{p}_{i+1}) \tag{5}$$

where the parameters $\mathbf{W}^{-1}, \mathbf{W}^0, \mathbf{W}^{+1}$ are shared in all trigrams, and $\boldsymbol{\alpha}_i$ denotes the importance score of the corresponding $\mathbf{p}_i$.

The position-aware convolution uses a similar strategy as the decomposable attention model. It also first soft-aligns $\mathbf{p}_i$ with all words h_j in the hypothesis H, and gets a soft alignment vector representation

$$\tilde{\mathbf{p}}_i = \sum_j \frac{\exp(\mathbf{I}[i,j])}{\sum_j \exp(\mathbf{I}[i,j])}\mathbf{h}_j \tag{6}$$

In addition, the word-word interaction is also used to find the hard coding of best matched position. For $\mathbf{p}_i$, we first find the index of the best matched word in H: $x_i = \arg\max_i(\mathbf{I}[i,:])$ and then encode the index with an embedding matrix M and transfers it into a continuous vector representation $\mathbf{z}_i = \mathbf{M}[x_i]$.

Concatenating the original $\mathbf{p}_i$ with the new position embedding z_i, we get a new hidden state representation $\mathbf{c}_i$. Then a position-aware convolution works at position i of P as: $\mathbf{n}_i = \tanh(\mathbf{W}[\mathbf{c}_{i-1}, \mathbf{c}_i, \mathbf{c}_{i+1}, \tilde{\mathbf{p}}_i] + \mathbf{b})$ Both the parameter-dynamic convolution and the position-aware convolution are stacked with a standard max-pooling layer to get the final representation $\mathbf{r}_{dyn}$ and $\mathbf{r}_{pos}$ respectively. Finally the concatenation $[\mathbf{r}_{dyn}, \mathbf{r}_{pos}]$ is fed to a logistic regression classifier to get the final prediction.

AWE: Asymmetric Word Embedding for Textual Entailment

Our idea is to model the asymmetric relation of entailment from *word level*. In a common deep learning approach, the words in the data are from the same embedding space. However, in textual entailment, we would like to model the relationship of $p(w$ **entails** $c)$ beyond the relationship of neighborhood co-occurrence that is used in general word embedding techniques (w is a word in premise and c is a word in hypothesis). Thus we modify the Skip-gram word2vec model by selecting different context words. Instead of utilizing the co-occurrence relationships in a neighborhood window, our method creates the entailment contexts as shown in Fig. 1.

In detail, the proposed approach can be decomposed into the following steps.

Entailment Word Pairs

The most critical step of our method is to create the entailment contexts. Different from previous approaches for asymmetric word embeddings, we do not utilize existing external knowledge but only focus on the entailment corpus.

Our assumption is that the entailment sentence consists of multiple word entailment relations. Thus we can find the entailment word pairs from the entailment sentence pairs. Another assumption is that if one word can infer another word, they are generally similar (but not vice versa). So our approach first finds the most similar words in the premise-hypothesis sentence pairs and regard them as possible entailment word pairs.

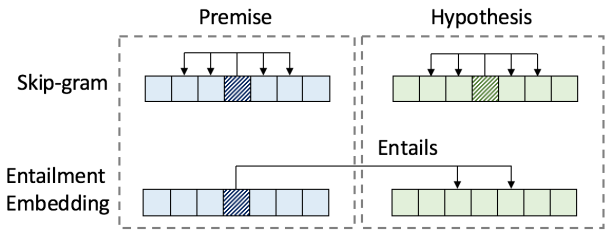

Figure 1: Skip-gram Word2vec vs. Entailment Embedding. In the Skip-gram model, we predict whether a word co-occurs with another word in its neighborhood window; while in the Entailment Embedding, given the word in premise, we predict whether it entails another word in hypothesis

The aforementioned idea is actually similar to the motivation of using word-word attention in a textual entailment model. However, if the embedding only considers the entailment sentence pairs, it will induce huge amounts of noise. Let us recall the sentence pair example in the Introduction. By calculating cosine similarities using pre-trained word embeddings, the candidate entailment word-pairs will include ("two", "four") because these two words generally have a high similarity. However, obviously the word pair ("two", "four") cannot represent an entailment relation. This is to say, using the entailment sentence pairs and word similarity only would bring in some noisy pairs.

One solution to reduce noise in the set of entailment word-pairs is to leverage the neutral (i.e. non-entailment) sentence pairs. If an word pair frequently occurs in both entailment sentence pairs and neutral sentence pairs, it is highly possible that it is noisy. Regarding the previous example, we have another sentence pair marked as "neutral":

Premise: *Four guys in wheelchairs on a basketball court two are trying to grab a basketball in midair.*

Hypothesis: *The four players are handicapped.*

Label: neutral

The previously mentioned noisy pair ("two", "four") are also included in this neutral sentence pair, so we can possibly remove the pair from the entailment set.

The following is a detailed description of the process to create the set of entailment word pairs:

- Download pre-trained word embeddings (e.g. pretrained Glove on Wikipedia) and represent each word w as a word vector $\mathbf{u}_w$.
- For each entailment sentence pair $P-H$ in the dataset (i.e. labeled as "entailment"), split the sentences into words. For each word c in P, calculate the cosine similarity between c and all words in H: $cos(c,w) = \frac{\mathbf{u}_c^T \cdot \mathbf{u}_w}{||\mathbf{u}_c|| \cdot ||\mathbf{u}_w||}$, put all pairs $U_{ent} = \{c, w | cos(w,v) > T_+, w \in P, c \in H\}$

in the entailment word-pair set, where T_+ is a similarity threshold.

- For each neutral sentence pair $S_1 - S_2$, also find all word pairs with the similarity larger than a threshold T_-: $U_{neu} = \{c, w | cos(c, w) > T_-, w \in S_1, c \in S_2\}$. From the word-pair set U_{ent} remove all word pairs which also occur in the set U_{neu}, hence getting the final clean entailment word-pair set $U = U_{ent}/U_{neu}$.

In SNLI, there is a third category of sentence pairs labeled as "contradiction", we do not consider this category when extracting the entailment word pairs. Only "entailment" and "neutral" pairs are used.

Entailment Word Embedding

Following the same idea of the Skip-gram model, we train entailment word embeddings on the entailment word-pair set U. Denote a word in premise as c, a word in hypothesis as w, $\mathbf{v}_c$ as the embedding for a word c in premise and $\mathbf{u}_w$ for a word w in hypothesis. The objective is to predict the presence of each entailment word-pair as given by Eq. 8.

$$\arg\max_{\theta} \sum_{c,w} \log p(c|w, \theta) \tag{7}$$

$$= \arg\max_{\theta} \sum_{w} \sum_{c \in N(w)} \log \frac{\exp(\mathbf{v}_c^T \cdot \mathbf{u}_w)}{\sum_{c'} \exp(\mathbf{v}_{c'}^T \cdot \mathbf{u}_w)}$$

where $N(w)$ is the corresponding word set of w in hypothesis.

This formula is impractical due to the large size of vocabulary (i.e. number of c' in the normalizer). To train the word embedding based on the entailment word pairs, we employ the negative sampling procedure (Mikolov et al. 2013). For each word pair (c, w) in U, we randomly sample K negative samples for c. The original objective $\log p(c|w, \theta)$ is thus approximated by:

$$\log \sigma(\mathbf{v}_c^T \cdot \mathbf{u}_w) + \sum_{i=0}^{K} \mathbb{E}_{c_i' \sim P_U(c')} \log \sigma(-\mathbf{v}_{c'}^T \cdot \mathbf{u}_w) \tag{8}$$

where K is the number of negative samples for each word w; $P_U(c')$ is the distribution of words in the set U; σ is the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$. More specifically, if we write the set of negative samples for w as $N_{w,c}$, the objective is rewritten as follows:

$$\log \sigma(\mathbf{v}_c^T \cdot \mathbf{u}_w) + \sum_{c' \in N_{w,c}} \log \sigma(-\mathbf{v}_{c'}^T \cdot \mathbf{u}_w) \tag{9}$$

Notice that different from the original Skip-gram based word embedding which represents a word with two vectors (input and output vectors), in our objective function each word w in the premise only has one vector representation; each word c in the hypothesis also has only one vector representation; and the two vector representations are different for even the same word in the premise and hypothesis. That is to say, the derived word representations are asymmetric (i.e. even for the same word w, $u_w \neq v_w$). The asymmetric word embeddings can be directly used for the inference of word-level entailment relationship, e.g. $p(w \rightarrow c) = p(c|w)$ where $\rightarrow$ denotes entailment. Hence, the word-word entailment is also asymmetric: $p(w \rightarrow c) \neq p(c \rightarrow w)$.

Unknown Words in Entailment Word Embedding

As we can only select a subset of words in the corpus to form the entailment word-pair set, for the remaining words, we will not get the new embedding representations using the aforementioned method. That will result in difficulties in utilizing the new embeddings for textual entailment. To solve the problem of representing the non-entailment words (i.e. the words occurring in the data corpus but not involved in the extracted entailment word-pair set U), we bring in two new terms called UNK_1 and UNK_2 to represent all these words.

In order to get the embeddings for UNK_1 and UNK_2, we add them to the negative sample set for each word when we do negative samplings in the algorithm of entailment word embedding. In detail, we change the Equation 9 into the following form:

$$OBJ = \log \sigma(\mathbf{v}_c^T \cdot \mathbf{u}_w) + \sum_{c' \in N_{w,c}} \log \sigma(-\mathbf{v}_{c'}^T \cdot \mathbf{u}_w) \tag{10}$$
$$+ \log \sigma(-\mathbf{v}_{UNK_2}^T \cdot \mathbf{u}_w) + \log \sigma(-\mathbf{v}_c^T \cdot \mathbf{u}_{UNK_1})$$

So if a word in the premise does not occur in the new embeddings, we can use $\mathbf{u}_{UNK_1}$ to represent it; if a word in the hypothesis does not occur in the new embeddings, we represent it with $\mathbf{v}_{UNK_2}$.

`AWEsify` Textual Entailment Models

From the process of generating entailment word-pairs, it is easy to see that only a subset of words are contained in the entailment word-pair set thus the derived new asymmetric word embeddings will only cover a fraction of words in the corpus. So if we simply take place of all original word embeddings used in a textual entailment model with our new asymmetric word embeddings, we will get a lot of UNKs which do not contain much information. Instead, our idea is to utilize the new asymmetric word embeddings to get more accurate alignment in a word-word interaction based model, such as the Decomposable Attention Model (Decomp-Att)(Parikh et al. 2016) and DeIsTe(Yin, Roth, and Schütze 2018). In this way, we can integrate the power of original word embeddings for basic input representation of a sentence model and the new asymmetric entailment embeddings for better alignment. In the following sections, we will describe how to utilize our new embeddings to modify Decomp-Att and DeIsTe.

`AWEsify` Decomp-Att

The word-word interaction in the Decomp-Att model is based on the soft neural attention in Equation 3. We use our new asymmetric embeddings to get new alignment vectors:

$$\boldsymbol{\beta}_i' := \sum_{j=1}^{l_h} \frac{\exp(\mathbf{v}_j^T \cdot \mathbf{u}_i)}{\sum_{k=1}^{l_h} \exp(\mathbf{v}_k^T \cdot \mathbf{u}_i))} \mathbf{h}_j \tag{11}$$

$$\boldsymbol{\alpha}_j' := \sum_{i=1}^{l_p} \frac{\exp(\mathbf{v}_j^T \cdot \mathbf{u}_i))}{\sum_{k=1}^{l_p} \exp(\mathbf{v}_j^T \cdot \mathbf{u}_k))} \mathbf{p}_i$$

where $\mathbf{u}_i$ is the entailment word embedding for word w_i in premise P, $\mathbf{v}_j$ is the entailment word embedding for word w_j in hypothesis H. Next we combine $\boldsymbol{\beta}'_i$ and $\boldsymbol{\alpha}'_j$ with $\boldsymbol{\beta}_i$ and $\boldsymbol{\alpha}_j$, and we add the combined vectors to the model for better prediction.

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_i &= \eta\boldsymbol{\beta}_i + (1-\eta)\boldsymbol{\beta}'_i \\
\hat{\boldsymbol{\alpha}}_j &= \eta\boldsymbol{\alpha}_j + (1-\eta)\boldsymbol{\alpha}'_j \\
\mathbf{v}'_1 &= \sum_i \mathbf{v}'_{1,i} = \sum_i G([\mathbf{p}_i, \hat{\boldsymbol{\beta}}_i]) \\
\mathbf{v}'_2 &= \sum_j \mathbf{v}'_{2,j} = \sum_j G([\mathbf{h}_j, \hat{\boldsymbol{\alpha}}_j]) \qquad (12)
\end{aligned}$$

where η is a learnable mixture weight. The final prediction of entailment is based on $\mathbf{v}'_1$ and $\mathbf{v}'_2$: $Y = H([\mathbf{v}'_1, \mathbf{v}'_2])$, where H is an feed-forward network. For convenience, we call this model **AWE-Decomp-Att** in the following sections.

AWEsify DeIsTe

To improve DeIsTe, we still first focus on improving the word-word interaction $\mathbf{I}[i,j] = cosine(\mathbf{p}_i, \mathbf{h}_j)$. A straightforward way to improve the model is to utilize the new asymmetric word embeddings to enhance the interaction:

$$\begin{aligned}
\mathbf{I}'[i,j] &= \max\{\mathbf{I}_0[i,j], \mathbf{I}_1[i.j]\} \\
&= \max\{cosine(\mathbf{p}_i, \mathbf{h}_j), cosine(\mathbf{u}_i, \mathbf{v}_j)\}
\end{aligned}$$

where $\mathbf{I}_0$ is the original word-word interaction in DeIsTe, and the new $\mathbf{I}_1 = cosine(\mathbf{u}_i, \mathbf{v}_j)$ indicates a new "entailment" probability.

In DeIsTe, the word-word interaction not only impacts on the alignment, but also helps derive an importance score of each word in P (as in Equation 5). For the new alignment calculation, we can directly use the new interaction matrix $\mathbf{I}'$ which integrates the word similarity and word entailment. However, we cannot directly use it to take place of the original $\max(\mathbf{I}_0)$ in the calculation of $\boldsymbol{\alpha}_i$. For the importance score of a word in the dynamic convolution network on P, a word with higher entailment probability should be more important. So we change the importance score a_i for $\mathbf{p}_i$ into the following form:

$$a_i = \frac{1 - \min(\mathbf{I}_1[i,:])}{1 + \max(\mathbf{I}_0[i:])} \qquad (13)$$

Compared to the one used in DeIsTe, the additional term $1-\min(\mathbf{I}_1[i,:])$ is used to punish the words with high entailment probability to infer even the noisy words in hypothesis. Generally an importance entailment word in premise can only infer a few words in hypothesis, so if a word has high inference probability (i.e. $\mathbf{I}$) with all the words in hypothesis, it might be a common word which does not provide useful information for sentence entailment. For convenience, the new model is called **AWE-DeIsTe** in the following sections.

Experiments

Data We used the following two natural language inference datasets: SNLI[1] and SciTail[2].

- **SNLI** Stanford Natural Language Inference (SNLI (Bowman et al. 2015)) is the largest entailment dataset (570k sentence pairs) created by asking annotators to write statements that would be true (or false) for an image given its caption. The label provided in are "entailment", "neutral", "contradiction" and "-". "-" means the annotators cannot reach consensus with each other, thus generally removed in previous works. In this paper We use the same data split as in (Bowman et al. 2015).
- **SciTail** (Khot, Sabharwal, and Clark 2018) is a more recent textual entailment dataset. It comprises 27K sentence pairs marked as entailment or neutral. It is designed from the end task of answering multiple-choice school-level science questions. With the close relationship with the question-answering task, a substantial performance gain on SciTail can be turned into better QA performance.

Baselines We want to show that asymmetric entailment embeddings could benefit word-word interaction based textual entailment models. Therefore, we compare our AWE-Decomp-Att and AWE-DeIsTe with their base models: Decomp-Att(Parikh et al. 2016) and DeIsTe(Yin, Roth, and Schütze 2018). Decomp-Att is one of the most applied advanced textual entailment model, and DeIsTe is the prior state-of-the-art model on SciTail. The results of DeIsTe on SciTail are from (Yin, Roth, and Schütze 2018), but on SNLI we use the results from our own running experiments, because the values reported in (Yin, Roth, and Schütze 2018) is from transfer learning (trained on SciTail and tested on SNLI) while in this paper for fair comparison we want to compare with the model trained on SNLI. In addition, we added HCRN (Tay, Luu, and Hui 2018) as another baseline, which also considers the asymmetric word matching problem in textual entailment but used a different way.

Training and Evaluation For both datasets, we first extracted the entailment word-pairs using a predefined threshold (T_+ and T_-). The asymmetric word embeddings are then trained on the word-pair set. Next we get the 300 dimensional embedding representations for all the words in the set as well as two special embedding vectors for UNK_1 and UNK_2.

For AWE-DeIsTe, we use the same setting and hyperparameters as DeIsTe in its original paper: batch size 50, no dropout, learning rate 0.01, filter width 3, hidden size 300. All words are initialized by 300D word2vec embeddings (Mikolov et al. 2013). For AWE-Decomp-Att, we also follow the hyperparameter settings of the original Decomp-Att model: 2-layers with hidden size 200, batch size 4, dropout raito 0.2, and learning rate 0.05. All wolds are initialized by 300D GloVe embeddings (Pennington, Socher, and Manning 2014). All embeddings remain fixed during training. All the models are trained by Adam (Kingma and

[1] https://nlp.stanford.edu/projects/snli/

[2] http://data.allenai.org/scitail/

Ba 2014). Notice that in our proposed AWE-Decomp-Att model, we omitted intra-sentence attention which is an optional component in Decomp-Att. However when comparing with Decomp-Att, we still compare with the best performance achieved by Decomp-Att which includes the intra-sentence attention. All methods are evaluated by accuracies on the development set and test set.

AWE-DeIsTe[3] is implemented by theano 1.0.2 (Theano Development Team 2016) based on the codes of DeIsTe[4], and AWE-Decomp-Att is implemented by Pytorch 0.3.1 (Paszke et al. 2017) and based on the Pytorch version of Decomp-Att[5]. For training models, we used a machine equipped with Intel Xeon E5-2640, 256GB RAM, four Nvidia Titan X and CUDA 8.0.

Results

Table 1 and Table 2 present the development accuracies and test accuracies on SciTail and SNLI. Both of the proposed models significantly outperform their base models. On test accuracy, by adding asymmetric word embeddings, AWE-Decomp-Att has a performance gain of 2.0% over Decomp-Att on SciTail and 0.3% on SNLI; AWE-DeIsTe outperforms DeIsTe by 2.1% on SciTail and 1.9% on SNLI. Especially, to the best of our knowledge, the AWE-Decomp-Att achieved the state-of-the-art performance on SciTail within all approaches without using external resources, see SciTail Leaderboard[6]. This demonstrates the effectiveness of our approach to improving word-word interaction based textual entailment models.

In addition, we also compared our model with a recently proposed advanced model, HCRN(Tay, Luu, and Hui 2018), which also considers asymmetric word matchings (HCRN does not publish their results on SNLI, so we only compare with it on SciTail). Different from our approach, HCRN still only uses the original symmetric word embeddings as input, but modifies the interaction function and make it asymmetric. It only applies to its own architecture while our method can be used to adapt most existing word-word interaction based models. By adding asymmetric word embeddings, we can also achieve asymmetric word interactions in the model, and the performance of our AWE-DeIsTe is about 4% higher than HCRN.

Table 1: Dev and test accuracies (%) on SciTail

	SciTail (dev)	SciTail (test)
HCRN	79.4	80.0
Decomp-Att	75.4	72.3
AWE-Decomp-Att	**77.1**	**74.3**
DeIsTe	82.4	82.1
AWE-DeIsTe	**85.1**	**84.2**

[3]Our codes are available at https://https://github.gatech.edu/cwu392/AWE-model

[4]https://github.com/yinwenpeng/Attentive Convolution

[5]https://github.com/libowen2121/SNLI-decomposable-attention

[6]http://data.allenai.org/scitail/leaderboard/

Table 2: Dev and test accuracies (%) on SNLI

	SNLI (dev)	SNLI (test)
Decomp-Att	86.4	86.8
AWE-Decomp-Att	**87.7**	**87.1**
DeIsTe	83.3	82.2
AWE-DeIsTe	**85.1**	**84.1**

Impact of Thresholds The thresholds T_+ and T_- are two important parameters to extract the entailment word pairs. Different thresholds will result in different asymmetric word embeddings, thus impacting largely on the textual entailment performance. In Figure 2 and Figure 3 we show how the performance of DeIsTe changes due to different thresholds. In order not to bring in too many unrelated word-pairs from the neutral sentence pairs, we fix the non-entailment threshold T_- at a high level (0.9) and only change the entailment threshold T_+. We can see that no matter how we changes the threshold T_+, the performance (both development accuracies and test accuracies) of AWE-DeIsTe is almost always better than DeIsTe (it does not have a threshold, so the values are unchanged). For both of the datasets, the best accuracies are achieved at $T_+ = 0.7$ and $T_- = 0.9$. Then for AWE-Decomp-Att, we fix the threshold and just use the best threshold parameters.

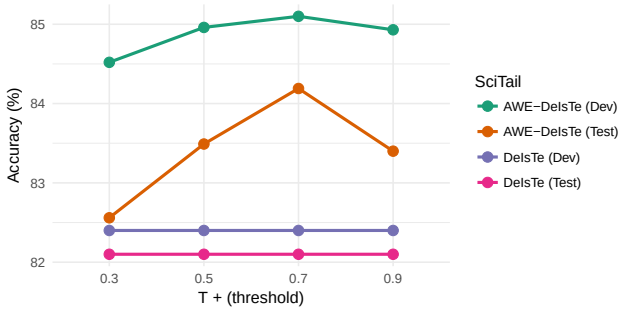

Figure 2: Performance Comparison (Accuracy v.s. Entailment threshold (T_+)) on SciTail. The non-entailment threshold T_- is fixed at 0.9.

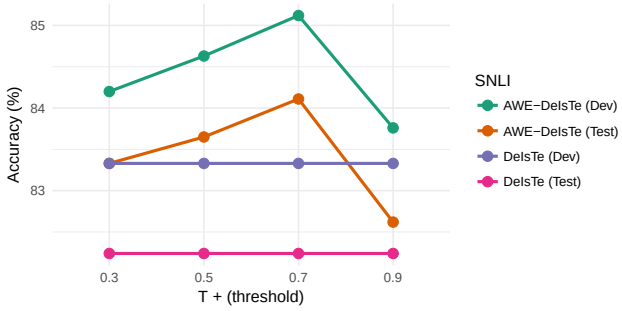

Figure 3: Performance Comparison (Accuracy v.s. Entailment threshold (T_+)) on SNLI. The non-entailment threshold T_- is fixed at 0.9.

Table 3: Cross-data evaluation of asymmetric word embeddings. "()" indicates the cross-data results, i.e. models are trained/tested on one dataset but the added asymmetric embeddings are trained from a different dataset. The results of AWE-DeIsTe without "()" are using the asymmetric word embeddings trained from the same dataset.

	SciTail (dev)	SciTail (test)	SNLI (dev)	SNLI (test)
DeIsTe (baseline)	82.4	82.1	83.3	82.2
AWE-DeIsTe + Asymmetric Embeddings on SNLI	(84.0)	(83.6)	**84.6**	83.7
AWE-DeIsTe + Asymmetric Embeddings on SciTail	**84.4**	**84.2**	(83.2)	(**84.1**)

Evaluation Embedding Transferability One advantage of our approach is that the extracted entailment word pairs and the derived asymmetric word embeddings can be further used for other tasks. To demonstrate it, we experimented with AWE-DeIsTe using asymmetric embeddings from other datasets. For example, when we train/test AWE-DeIsTe on SciTail, instead of using asymmetric word embeddings derived from SciTail, we used the alternative asymmetric embeddings from SNLI. Table 3 show the cross-data results of AWE-DeIsTe. Although the new asymmetric word embeddings are not from the same training data, they are still useful to get better results than the baseline model which did not use asymmetric word embeddings. Thanks to the large scale of both SciTail and SNLI, the cross-data test accuracies are even comparable to the original AWE-DeIsTe.

Related Works

Textual Entailment Earlier research on textual entailment (somewhere also called natural language inference) were restricted to small data size and relied on hand-crafted features and alignment systems (Androutsopoulos and Malakasiotis 2010). The development of large-scale annotated datasets (Bowman et al. 2015) motivated a series of deep neural network based algorithms (Wang and Jiang 2015; Chen et al. 2016; Liu et al. 2016) and have led to great progress in textual entailment. They can be categorized as below: 1) sentence encoding based models that find a vector representation for each sentence and classify the concatenation/difference/product of the two vector representations; 2) joint feature models which use cross-sentence features or attentions (Rocktäschel et al. 2016; Wang, Hamza, and Florian 2017; Parikh et al. 2016). The sentence matching framework generally does not differentiate textual entailment with other sentence matching tasks, such as paraphrase identification, sentence similarity evaluation, for example, in (Kim et al. 2018; McCann et al. 2017), the same framework can be used for different sentence matching tasks.

Word-Word Interaction Since the neural attention mechanism has received success in various natural language processing tasks, it is also used in textual entailment for capturing word-by-word interactions (Rocktäschel et al. 2016; Parikh et al. 2016; Yin, Roth, and Schütze 2018). In addition, some variants of attention techniques are proposed to improve the interaction architecture, including self-attention (Parikh et al. 2016), multi-head attention (Vaswani et al. 2017), and interactive inference network (Gong, Luo, and Zhang 2017). Our approach lies on the thread of improving word-word interactions. Unlike previous works, we derive a new type of asymmetric word embedding motivated by the asymmetric relationship of entailments to represent the word-level entailment relation.

Asymmetric Word Embedding Asymmetric word relations have been explored for textual entailment. In hypernymy (Chen et al. 2017), the asymmetric word relations were built on external knowledge and were expensive to train. The HRCN (Tay, Luu, and Hui 2018) uses a hermitian co-attention network to model the asymmetric text matching, but it is limited to its own structure. In addition, there are also some works studying the asymmetric word embedding techniques and hypernymy detection(Vulić and Mrkšić 2017; Nguyen et al. 2017), but these techniques are only concerning about word-level entailment. In this work, our proposed approach does not utilize any external knowledge and thus keeps the textual entailment model lightweight. Moreover, our AWE is a general component which can be added to different word-word interaction based models as we showed in DeIsTe(Yin, Roth, and Schütze 2018) and Decomp-Att(Parikh et al. 2016).

Conclusion

In this paper we present asymmetric word embedding AWE, a simple but highly effective approach for textual entailment that leverages the asymmetric word-word interactions. We derive a new type of asymmetric word embedding from the entailment word pairs. The new asymmetric word embeddings are then used to adapt existing word-word interaction based textual entailment models, such as Decomp-Att and DeIsTe. The derived new models AWE-Decomp-Att and AWE-DeIsTe sigfinicantly outperforms their base models and AWE-DeIsTe achievs the state-of-the-art performance on SciTail, demonstrating the effectiveness of our approach. Furthermore, we prove that the learned asymmetric word embeddings is even helpful when models are trained on other datasets. Future research includes extending this work to other advanced word-word interaction based models and other asymmetric text matching tasks such as question-answering.

References

[Androutsopoulos and Malakasiotis 2010] Androutsopoulos, I., and Malakasiotis, P. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research* 38:135–187.

[Bowman et al. 2015] Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus

for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

[Chen et al. 2016] Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; and Jiang, H. 2016. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

[Chen et al. 2017] Chen, Q.; Zhu, X.; Ling, Z.-H.; and Inkpen, D. 2017. Natural language inference with external knowledge. *arXiv preprint arXiv:1711.04289*.

[Clinchant, Goutte, and Gaussier 2006] Clinchant, S.; Goutte, C.; and Gaussier, E. 2006. Lexical entailment for information retrieval. In *European Conference on Information Retrieval*, 217–228. Springer.

[Gong, Luo, and Zhang 2017] Gong, Y.; Luo, H.; and Zhang, J. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

[Khot, Sabharwal, and Clark 2018] Khot, T.; Sabharwal, A.; and Clark, P. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of AAAI*.

[Kim et al. 2018] Kim, S.; Hong, J.-H.; Kang, I.; and Kwak, N. 2018. Semantic sentence matching with densely-connected recurrent and co-attentive information. *arXiv preprint arXiv:1805.11360*.

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kusner et al. 2015] Kusner, M. J.; Sun, Y.; Kolkin, N. I.; and Weinberger, K. Q. 2015. From word embeddings to document distances. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, 957–966.

[Liu et al. 2016] Liu, P.; Qiu, X.; Chen, J.; and Huang, X. 2016. Deep fusion lstms for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1034–1043.

[McCann et al. 2017] McCann, B.; Bradbury, J.; Xiong, C.; and Socher, R. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, 6294–6305.

[Mikolov et al. 2013] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

[Nguyen et al. 2017] Nguyen, K. A.; Köper, M.; Walde, S. S. i.; and Vu, N. T. 2017. Hierarchical embeddings for hypernymy detection and directionality. *arXiv preprint arXiv:1707.07273*.

[Parikh et al. 2016] Parikh, A. P.; Täckström, O.; Das, D.; and Uszkoreit, J. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

[Pasunuru, Guo, and Bansal 2017] Pasunuru, R.; Guo, H.; and Bansal, M. 2017. Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, 27–32.

[Paszke et al. 2017] Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

[Pennington, Socher, and Manning 2014] Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

[Rocktäschel et al. 2016] Rocktäschel, T.; Grefenstette, E.; Hermann, K. M.; Kočiskỳ, T.; and Blunsom, P. 2016. Reasoning about entailment with neural attention. *International Conference on Learning Representations (ICLR)*.

[Tay, Luu, and Hui 2018] Tay, Y.; Luu, A. T.; and Hui, S. C. 2018. Hermitian co-attention networks for text matching in asymmetrical domains. In *IJCAI*, 4425–4431.

[Theano Development Team 2016] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

[Vulić and Mrkšić 2017] Vulić, I., and Mrkšić, N. 2017. Specialising word vectors for lexical entailment. *arXiv preprint arXiv:1710.06371*.

[Wang and Jiang 2015] Wang, S., and Jiang, J. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

[Wang, Hamza, and Florian 2017] Wang, Z.; Hamza, W.; and Florian, R. 2017. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

[Yin, Roth, and Schütze 2018] Yin, W.; Roth, D.; and Schütze, H. 2018. End-task oriented textual entailment via deep exploring inter-sentence interactions. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.