An Efficient Framework for Sentence Similarity Modeling

Zhe Quan, Zhi-Jie Wang, Yuquan Le, Bin Yao, Kenli Li, and Jian Yin

***Abstract*—Sentence similarity modeling lies at the core of many natural language processing applications, and thus has received much attention. Owing to the success of word embeddings, recently, popular neural network methods achieved sentence embedding. Most of them focused on learning semantic information and modeling it as a continuous vector, yet the syntactic information of sentences has not been fully exploited. On the other hand, prior works have shown the benefits of structured trees that include syntactic information, while few methods in this branch utilized the advantages of word embeddings and another powerful technique — attention weight mechanism. This paper suggests to absorb their advantages by merging these techniques in a unified structure, dubbed as ACV-tree. Meanwhile, this paper develops a new tree kernel, known as ACVT kernel, that is tailored for sentence similarity measure based on the proposed structure. The experimental results, based on 19 widely-used datasets, demonstrate that our model is effective and competitive, compared against state-of-the-art models. Additionally, the experimental results validate that many attention weigh mechanisms and word embedding techniques can be seamlessly integrated into our model, demonstrating the robustness and universality of our model.**

***Index Terms*—Sentence similarity, word embedding, attention weight, syntactic structure**

I. Introduction

SENTENCE similarity is a fundamental metric to measure the degree of likelihood between a pair of sentences [1], [2], [3], and plays an important role for many applications [4], [5], [6], [7]. Measuring sentence similarity is challenging due to the ambiguity and variability of linguistic expression, and thus has received much attention in recent years [8], [9], [10]. A large number of prior works focused on feature engineering, and several types of sparse features have been shown to be useful, such as knowledge-based [11] and corpus-based [12]. Some methods also used the combination of various features and multi-task learning [13].

Recently, owing to the success of *word embeddings* [14], [1], researchers have attempted to study sentence similarity modeling via *sentence embeddings*. This approach has become a successful paradigm in natural language processing (NLP) community [2], [15]; and particularly some studies have used the *attention weight mechanism* to further enhance the performance [15], [16]. In this line of works, most of previous studies focused on learning semantic information and modeling it as a continuous vector, while the syntactic information of sentences are not fully exploited. On the other hand, prior works have shown the benefits of structured trees that include syntactic information [17], [18]. Yet, few works in this branch utilized the advantages of word embeddings and the attention weight mechanism.

Inspired by the above observations, in this paper we attempt to absorb the advantages of the above mentioned techniques, and develop a more efficient method. In a nutshell, our model uses a structured manner for sentence similarity modeling. It seamlessly integrates semantic information, syntactic information, and the attention weight mechanism. To measure similarity, we develops a new tree kernel, known as the ACVT kernel, that is tailored for our proposed structure and is designed for high operability. Our model can be used as a general framework, since one can view word embedding and attention weight as the *building blocks* of the framework, allowing users to replace them using other on-shelf (or more powerful, developed in the future) word embedding techniques and attention weight schemes. Besides, unlike most of sentence embedding-based models, our model can be free from time-consuming learning/training, once word embeddings are available. On the other hand, there are also word embedding-based models [8], [2] for sentence similarity modeling. Nevertheless, our model can achieve better performance on almost all datasets used in our experiments, compared against the word-embedding based models. The novelty of this work is twofold at least: (i) it suggests a novel manner for sentence modeling, and (ii) it develops a new tree kernel.

To summarize, the main contributions of this paper are:

- We propose a new structure for sentence similarity modeling. Our model wisely combines syntactic information, semantic features, and attention weight mechanism together, absorbing the merits of various techniques. Our model is easily understood and implemented, but without loss of effectiveness.
- We developed the ACVT kernel that can allow us to efficiently perform similarity measure based on the proposed structure.
- We conduct extensive experiments based on widely-used benchmark datasets. The experimental results consistently demonstrate the superiorities and competitiveness of our proposed model.

In the next section, we introduce some preliminaries. Section III covers our proposed model. In Section IV, we report and analyze the experimental results. Section V review pre-

Z. Quan, Y. Le and K. Li are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. Email: {leyuquan,quanzhe,lkl}@hnu.edu.cn.

Z. Wang and J. Yin are with the Guangdong Key Laboratory of Big Data Analysis and Processing, School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China. Email: {wangzhij5,issjyin}@mail.sysu.edu.cn.

B. Yao is with the Department of Computer Science & Engineering, Shanghai Jiao Tong University, Shanghai, China. Email: yaobin@cs.sjtu.edu.cn.

vious works most related to ours. Finally, we conclude this paper in Section VI.

II. Background

In this section we review several main techniques, for ease of understanding our proposed model.

A. Constituency-based Parse Tree

It is known as the *constituency tree*. The interior nodes are labelled by non-terminal categories of the grammar, while the leaf nodes are labelled by terminal categories [19]. Each node in the tree is either a root node, a branch node, a non-branch node, or a leaf node. A root node doesn't have any branches on top of it. Within a sentence, there is only ever one root node. A branch node is a parent node that connects to two or more child nodes. A non-branch node is a parent node that contains only a single child. A leaf node, however, is a terminal node that does not dominate other nodes in the tree. The fundamental trait of the constituency tree is that we view sentence structure in terms of the constituency relation. Consider a sentence "*Love makes man grow up*" as an example, the constituency tree that represents the syntactic structure of this sentence is shown in Figure 1. Note that, by convention, the constituency tree usually uses some abbreviations. For example, "S for sentence", "N for noun", "NP for noun phrase", "V for verb", "VP for verb phrase", and so on.

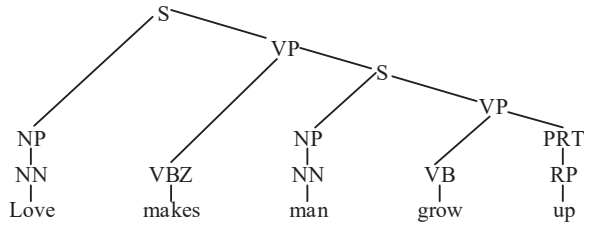


Fig. 1. Constituency tree .

B. Word Embedding

Recently, a popular framework can allow users to represent words as continuous vectors that capture lexical and semantic properties of words [14], [1]. Usually, this technique is known as *word embedding* (a.k.a., *distributed vector representation of words*). Figure 2 shows an example of a representative framework (notice: for ease of understanding, the readers can rotate the figure 90 degrees). In brief, in this framework every word is mapped to a unique vector that is represented by a column in a matrix W. The column is indexed by position of the word in the vocabulary. The concatenation or sum of the vectors is then used as features for prediction of the next word in a sentence. More formally, given a sequence of context words w_1,w_2,w_3, ...,w_T, the objective of the word vector model is to maximize the average log probability $\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t|w_{t-k},...,w_{t+k})$. The neural network based word embeddings are usually trained using stochastic gradient descent where the gradient is obtained via backpropagation. After the training converges, words with similar meaning are mapped to a similar position in the vector space. For example, "pretty" and "beautiful" are close to each other, whereas "beautiful" and "cup" are more distant. The prediction task is typically done via a multiclass classifier, such as softmax. It can be formulated as $p(w_t|w_{t-k},...,w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_t e^{y_i}}$. Each of y_i is un-normalized log probability for each output word i, and it is computed as $y = b + Uh(w_{t-k},...,w_{t+k};W)$, where U, b are the softmax parameters, and h is constructed by a concatenation or average of word vectors extracted from matrix W. As a remark, although the framework described here is from the CBOW model [20], most of other models are similar in spirit to it.

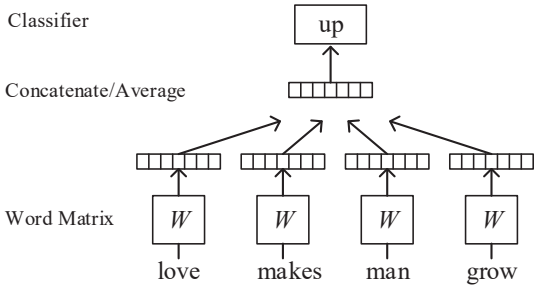


Fig. 2. Framework of word embedding.

C. Weight Mechanism

Most of neural network based sentence representation models treat each word in sentences equally [21], [22], [8]. This mechanism could be ineffective since it is inconsistent with the way that human read and understand sentences (i.e., reading some words superficially and paying more attention to others). So far, extensive studies have proven that word attributes, as represented by frequency, POS tag, length, Surprisal, etc., are all correlated with human reading time [23]. Thereby, researchers have considered to assign words with different weights (which can be treated as attention mechanism [15]), and there are many schemes to assign weights to words, such as *smooth inverse frequency* (SIF), *inverse document frequency* (IDF), *term frequency-inverse document frequency* (TF-IDF), *POS tag* (POS), [15], [24], [16]. Remark that, a more strict definition on the attention weight can be found in [25]; in this paper we slightly relax this notion, for ease of presentation.

D. Tree Kernel

Tree kernel is used to compute the similarity between structured trees. The main idea of tree kernels is to compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space [26]. A tree kernel function over T_1 and T_2 is defined as

$$TK(T_1,T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1,n_2) \qquad (1)$$

where N_{T_1} and N_{T_2} denote the set of nodes in T_1 and T_2, respectively. Note that, the $\Delta(\cdot)$ function determines the richness of the kernel space and thus can yield different tree kernels. A representative tree kernel, known as *partial tree*

kernel (PTK) [26], is highly related to our proposed tree kernel. Specifically, the $\Delta(\cdot)$ function of PTK is computed as

$$\Delta(n_1, n_2) = \begin{cases} \mu(\lambda^2 + \sum_{p=1}^{l_m} \Delta_p(c_{n_1}, c_{n_2})), n_1 = n_2 \\ 0, \text{otherwise} \end{cases} \quad (2)$$

where μ and λ are two decay factors: μ for the height of the tree, and λ for the length of the child sequences; c_{n_1} (resp., c_{n_2}) refers to the list of children nodes of n_1 (resp., n_2), in which the nodes are stored in order as same as the original order in the tree; $l_m = min\{\text{length}(c_{n_1}), \text{length}(c_{n_2})\}$; and $\Delta_p(\cdot)$ refers to the number of common subsequence whose length is p.

III. Model

In this section, we first cover the proposed structure, and then expatiate the new tree kernel tailored for computing similarity based on our structure.

A. ACV-tree

At a high level, the ACV-tree (Attention Constituency Vector-tree) is similar to the so-called constituency tree, since (1) it is also tree-like structure with only a root denoting the sentence; (2) the internal nodes are some abbreviations for various constituencies such as NP and VP; and (3) the leaf nodes contain also the words. A major difference is that, the leaf nodes of ACV-tree contain also two other elements besides the words: one is a vector storing the semantic information of the corresponding word, the other is a real number denoting the weight of the corresponding word. In what follows, we address how to construct the ACV-tree in detail.

To construct the ACV-tree, one can follow several steps below. First, we determine "part of speech" for each word in the sentence. For example, the word *man* is a noun and the word *makes* is a verb. Second, we associate each word with (1) the word vector, which can be trained from unlabelled texts in large corpus, and (2) the attention weight, which can distinguish the contribution of different words to the semantic meaning of sentences. Third, we find the *modification relations* (or dependency relations) of words in the sentence. For example, in the sentence "love makes man grow up", the word *up* modifies the word *grow*, and the words *grow up* modifies the word *man*. Finally, as similar as that in [19], we link items according to the modification relations (or dependency relations) found by the previous step, until all the modifiers are attached to the modified constituents. Note that, in the process of "linking", different rules shall be used (e.g., when a modifier (or word) modifies a noun, the NP rule is to be used). As such, we obtain our ACV-tree as shown in Figure 3.

B. ACVT Kernel

As mentioned before, tree kernel is used to compute similarity between structured trees. Yet, few existing tree kernels consider both the semantic information and the attention weight. To alleviate this issue, we develop a new tree kernel

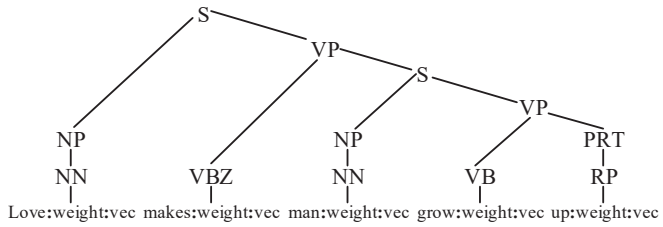


Fig. 3. An example of ACV-tree

known as ACVT kernel (Attention Constituency Vector Tree kernel). Our tree kernel is tailored for computing similarity based on the proposed ACV-tree.

As same as almost all existing tree kernels, our ACVT kernel uses also the general framework. That is, Equation 1 is also used. The major difference between our tree kernel and existing tree kernels is the $\Delta(\cdot)$ function. Let vec_1 and wt_1 (resp., vec_2 and wt_2) be the word vector and attention weight of node n_1 (resp., n_2). Our $\Delta(n_1, n_2)$ function is inspired by that of PTK (recall Section II). Specifically, it is defined as

$$\Delta(n_1, n_2) = \begin{cases} 0, n_1 \text{ and/or } n_2 \text{ are non-leaf nodes } \wedge n_1 \neq n_2 \\ Att_{weight} \times SIM(vec_1, vec_2), n_1 \text{ and } n_2 \text{ are leaf nodes} \\ \mu(\lambda^2 + \sum_{p=1}^{l_m} \Delta_p(c_{n_1}, c_{n_2})), \text{otherwise} \end{cases} \quad (3)$$

where c_{n_1}, c_{n_2}, l_m, μ and λ have the same meaning as mentioned in Section II; vec_1 and vec_2 are the word vector of n_1 and n_2, respectively; $SIM(\cdot, \cdot)$ is a function to measure the cosine similarity between vectors; $Att_{weight} = wt_1 \times wt_2$. Remark that, the symbol "$\neq$" means that the strings of these two nodes do not match; e.g., a node with a string "NP" and a node with a string "VB".

It remains to explain how to compute $\Delta_p(\cdot)$ as far as our tree kernel. To understand, consider $c_{n_1} = s_1 a$ and $c_{n_2} = s_2 b$ (a and b are the last children, s_1 and s_2 are subsequences of c_{n_1} and c_{n_2}, respectively), then one can solve $\Delta_p(c_{n_1}, c_{n_2})$ by constructing a "recursive" function as follows.

$$\begin{aligned} \Delta_p(s_1 a, s_2 b) = &\Delta(a, b) \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} (\lambda^{|s_1|-i+|s_2|-r} \\ &\times \Delta_{p-1}(s_1[1:i], s_2[1:r])) \end{aligned} \quad (4)$$

where $s_1[1:i]$ (resp., $s_2[1:r]$) is the subsequence of s_1 (resp., s_2) ranging from 1 to i (resp., from 1 to r); $|s_1|$ (resp., $|s_2|$) is the length of s_1 (resp., s_2). Note that, here $\Delta(a, b)$ is computed using Equation 3, while $\Delta_{p-1}(\cdot)$ is recursively computed using Equation 4, and the recursive process stops when it reaches at the leaf node.

C. Algorithm

The pseudo-codes of our algorithm for computing similarity score between two trees T_1 and T_2 are shown in Algorithm 1. Our algorithm follows the paradigm in [26]. In a nutshell, it works as follows. First, it constructs a matrix $\boldsymbol{K}$, and then compute the similarity of each node pair based on the rules in Equation 3. Note that, for the case of "otherwise" mentioned in Equation 3, our algorithm treats these two nodes as two new

Algorithm 1 COMPSIM(T_1,T_2)

```
1: for each node n_i in T_1 do
2:     if isLeaf(n_i) = true then
3:         for each node n_j in T_2 do
4:             if isLeaf(n_j) = true then
5:                 K[n_i][n_j] ← Att_weight × SIM(vec_1, vec_2);
6:             else
7:                 K[n_i][n_j] ← 0;
8:     else
9:         for each node n_j in T_2 do
10:            if isLeaf(n_j) = true then
11:                K[n_i][n_j] ← 0;
12:            else
13:                if n_i has the same structure with n_j then
14:                    K[n_i][n_j] ← COMPSIM(n_i, n_j);
15:                else
16:                    K[n_i][n_j] ← 0;
17: S_{T_1T_2} ← compute the sum of K and normalize it;
18: return S_{T_1T_2}
```

trees and compute their similarity (see Line 14). The final step is the same as that in [26], namely, we compute the sum of all values in $\boldsymbol{K}$ and normalize it, obtaining the similarity score.

D. Universality of Our Model

Up to now, we have presented our model for sentence similarity modeling. Our model contains a new structure ACV-tree, and a new tree kernel (i.e., ACVT kernel). Note that, the proposed structure has two major building blocks: *word embedding* and *attention weight mechanism*. Although there are many word embedding techniques (e.g., *PSL vectors* [27], *PWS vectors*[27], *D2V vectors* [28], *SGNS vectors* [29]) and attention weight mechanisms (e.g., *TF-IDF* [30], [15], *IDF* [24], *POS tags* [15], *SIF* [16]), our model can seamlessly integrate the existing word embedding technique (resp., attention weight mechanism), and it is easy to implement. In brief, at the second step of constructing ACV-tree (recall III-A), one can associate each word with the corresponding word vector (resp., attention weight). In our experiments, most of tests are based on the PSL vectors (one of common word embeddings) and TF-IDF (one of common attention weights). Nevertheless, to validate the universality of our model, our later experiments also investigate the performance of many other word embeddings and attention weight mechanisms.

IV. Experiments

A. Datasets and Experimental Settings

Datasets. Following prior works, we conduct experiments on 19 textual similarity datasets (http://ixa.si.ehu.es/stswiki/index.php/Main_Page) that contain all the datasets from Semantic Textual Similarity (STS) tasks (2012-2015), except the SMT dataset in 2013 due to no permission. Each dataset contains many pairs of sentences (e.g. MSRvid dataset contains 750 pairs of sentences). These datasets cover a wide range of domains such as news, web forum, images, glosses, twitter. Table I summarizes these datasets (grouped by year). Please note that datasets with the same name in different years include different data.

Compared methods. In our experiments, we mainly compare two sets of baselines (note that, most of models in the second category are classic and earlier than those in the first category):

(1) The models that use word embedding and/or sentence embedding techniques, including Glove [9], PSL [27], ST [22], SCBOW [2], PROJ [8], PP-tfidf [15], DAN [5], LSTM [31], RNN and iRNN [8]. The results of the above methods are collected from [8] except SCBOW from [2] and PP-tfidf from [15].

(2) The models that are developed based on other techniques, including WUP [32], RES [33], LIN [34], JCN [35], and LCH [36], ESA [37], ADW [4]. For ESA and ADW, they have many variants, we choose the best of them for comparison. The results of these classic methods are collected from [4].

Other settings. In our paper we use the Pearson's correlation between the predicted scores and the ground-truth scores as the evaluation criterion, which is the same as that in [4], [15], [2]. The similarity score of sentence pair is from 0 to 5, where a scale of 5 means semantically equivalence, whereas 0 means complete unrelated. In our experiments, the hyper-parameters μ=[0.1,**0.2**,...,0.9,1.0], and λ =[**0.1**,0.2,...,0.9,1.0], where the numbers in **bold** denote the default settings, unless otherwise stated. In our experiments, we implement our ACV-tree by using the *Stanford Parser* [38] to generate the constituency tree of the sentence, and then attach the word vectors (i.e., lexical vectors) and the attention weights to the words in leaf nodes, for the sake of simplicity. Following prior works [16], [15], we use the *term frequency-inverse document frequency* (TF-IDF) scheme to generate the attention weights. In the computation, we view each sentence as a document and use all sentences in test data to calculate IDF. The lexical vectors we used are provided by PARAGRAM-SL999 (PSL for short) vectors, which is learned by PPDB and is the 300 dimensional Paragram embeddings tuned on SimLex999 dataset [27].

Roadmap of our experiments. In what follows, we first

STS'12	STS'13	STS'14	STS'15
MSRpar: (750,750)	headlines: (-,750)	deft-forum: (-,450)	answers-forums: (-,375)
MSRvid : (750,750)	OnWN : (-,561)	deft-news : (-,300)	answers-students : (-,750)
SMTeuroparl : (734,459)	FNWN : (-,189)	headlines : (-,750)	belief: (-,375)
OnWN : (-,750)	SMT : (-,750)	images: (-,750)	headlines: (-,750)
SMTnews : (-,399)		OnWN: (-,750)	images: (-,750)
		tweet-news: (-,750)	

TABLE I
DATASETS FOR THE SEMEVAL SEMANTIC TEXTUAL SIMILARITY TASKS (YEAR 2012 — YEAR 2015). NOTE THAT, IN THE TABLE THE NUMBERS IN BRACKET REFER TO THE NUMBER OF TRAINING DATA AND THE NUMBER OF TEST DATA, RESPECTIVELY. IN ADDITION, THE NOTATION "-" DENOTES THAT NO ANY ACCOMPANYING TRAINING DATA IS PROVIDED.

Year	Dataset	Compared methods										Our method
		PROJ	PP-tfidf	DAN	RNN	LSTM	ST	GloVe	PSL	iRNN	SCBOW	ACVT
2012	MSRpar	0.44	0.47	0.40	0.19	0.09	0.17	0.48	0.42	0.43	0.44	**0.58**
	MSRvid	0.74	0.79	0.70	0.67	0.71	0.42	0.64	0.60	0.73	0.45	**0.83**
	SMTeuroparl	0.49	**0.52**	0.44	0.41	0.44	0.35	0.46	0.42	0.47	0.45	0.43
	OnWN	0.70	**0.73**	0.66	0.63	0.56	0.30	0.55	0.63	0.70	0.64	0.70
	SMTnews	0.63	**0.66**	0.60	0.51	0.51	0.31	0.50	0.57	0.58	0.39	0.54
2013	headlines	0.73	0.74	0.71	0.60	0.49	0.35	0.64	0.69	0.73	0.65	**0.77**
	OnWN	0.68	0.75	0.64	0.55	0.50	0.10	0.49	0.48	0.69	0.50	**0.85**
	FNWN	0.47	0.50	0.43	0.31	0.38	0.30	0.34	0.38	0.45	0.23	**0.50**
2014	deft-forum	0.51	**0.54**	0.49	0.42	0.46	0.13	0.27	0.37	0.49	0.41	0.48
	deft-news	0.72	0.74	0.72	0.54	0.39	0.24	0.68	0.67	0.72	0.59	**0.74**
	headlines	0.71	0.71	0.69	0.58	0.51	0.38	0.60	0.65	0.70	0.64	**0.72**
	images	0.78	0.81	0.77	0.68	0.63	0.51	0.61	0.62	0.78	0.65	**0.81**
	OnWN	0.80	0.81	0.76	0.68	0.62	0.23	0.58	0.61	0.79	0.61	**0.87**
	tweet-news	0.76	**0.77**	0.74	0.58	0.48	0.40	0.51	0.65	0.77	0.73	0.75
2015	answers-forums	0.65	0.68	0.63	0.33	0.51	0.36	0.31	0.39	0.67	0.22	**0.69**
	answers-students	0.78	0.79	0.78	0.65	0.56	0.33	0.63	0.69	0.78	0.37	**0.79**
	belief	0.75	**0.78**	0.72	0.52	0.53	0.25	0.41	0.53	0.76	0.48	0.70
	headlines	0.75	0.77	0.74	0.65	0.57	0.44	0.62	0.69	0.75	0.22	**0.79**
	images	0.80	**0.84**	0.78	0.71	0.64	0.18	0.68	0.70	0.81	0.26	0.82

TABLE II
THE COMPARISON RESULTS; THE BOLD NUMBER HIGHLIGHTS ONE OF STRONGEST RESULTS IN EACH DATASET.

compare with the methods that used word/sentence embedding techniques, since these methods are closest to our proposed model (Section IV-B). In addition, we check whether our model can beat classic methods (Section IV-C). One could be curious that there are studies showing stronger performances in sentence embedding like Infersent [3] and Quick-Thought [39], it could be interesting to compare these methods (Section IV-D). Then, we study the impact of important parameters (Section IV-E). Also, we investigate the effectiveness of syntactic information from another viewpoint, since the aforementioned studies could not well reflect the effectiveness of syntactic information (Section IV-F). After that, we study the universality of our model by using various attention weight mechanisms and word embedding techniques (Section IV-G). It is interesting to compare some more widely used word embedding methods. To this end, we also compare our method with Glove [9] and FastText [40] (Section IV-H). Finally, we also conduct an extra experiment by replacing constituency tree with another well-known syntactic structure — dependency tree (Section IV-I). Note that, for ease of presentation, the experimental settings for investigating the universality of our model are placed in Section IV-G, since these settings are independent of other experiments.

B. Comparing with word/sentence embedding-based methods

Table II shows the comparison results. It can be seen from this table that our proposed method (shorted as ACVT) gets favourable performance. Specifically, ACVT achieves the best performance on 12 out of 19 datasets (notice: in NLP community "12 out of 19" is an attractive result [8]). Besides, we observe that for some datasets, although our method is not the best one, it is close to the best result (e.g., 12' OnWN, 14' tweet-news, 15' images). These evidences demonstrate that our proposed model is effective and competitive. Essentially, it implies that a combination of syntax, semantics and word attention mechanism could be a good choice for sentence similarity modeling. Nevertheless, we find that, on several datasets including 12' SMTeuroparl, 12' SMTnews, 14' deft-forum, and 15' belief, our method is inferior than the strongest competitor and the performance gap is larger than 0.05. It is interesting to understand why our method cannot performs well on these datasets.

As for 12' SMTeuroparl and 12' SMTnews datasets, it could be due to that some particular properties such as a large number of numerical items or special characters in these datasets weaken the performance of our model. For example, in the 12' SMTeu dataset items like "5.30pm" and "(A5-0323/2000)" account for around 10% of the total test sample; in the 12' SMTnews dataset, items like "5.2%" and "24 May" account for around 8% of the total test sample. Note that, our model currently lacks for the strong ability to model numerical items and special characters (e.g., the sentence pair for phone numbers and email addresses).

Meanwhile, we find that the 14' def forum dataset contains the forum post sentences, and the 15' belief dataset contains the Belief pairs for which their source documents are English Discussion Forum data. It is easy to understand that people usually write sentences in forums without using rigorous syntactic format, and so the grammars used in these sentences could be not guaranteed; and particularly sentences are also doped with a large number of colloquial terms and network abbreviations. These factors lead to the construction of syntactic structure inaccurate, weakening the performance of our model.

C. Comparing with classic methods

As same as to [4], we here also use the SemEval-2012 Semantic Similarity task to compare these classic methods.

Dataset	Compared methods							Our method
	JCN	WUP	LCH	LIN	RES	ESA	ADW	ACVT
MSRvid	0.65	0.64	0.67	0.70	0.73	0.74	0.80	**0.83**
MSRpar	0.44	0.44	0.45	0.45	0.46	0.44	0.56	**0.58**
SMTeuroparl	0.20	0.21	0.18	0.23	0.25	0.48	**0.55**	0.43
OnWN	0.53	0.55	0.53	0.57	0.59	0.62	0.63	**0.70**
SMTnews	0.26	0.28	0.27	0.28	0.30	0.40	0.40	**0.54**

TABLE III
THE COMPARISON BETWEEN OUR METHOD AND CLASSIC METHODS.

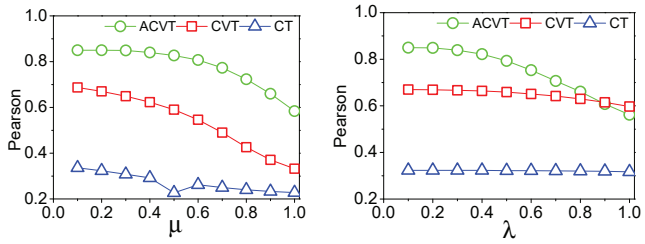
Table III lists the compared results. It can be seen from the table, our method beats all these methods for almost all these datasets. This further demonstrates the competitiveness of our model. Note that, as for the SMTeuroparl dataset, our model is significantly inferior than ADW (i.e., the performance gap is about 0.12). The reason is the same as our previous analysis. That is, this dataset contains much more *numerical items* and *special characters* for which our model lacks for the strong ability to model.

D. Comparing with Infersent and Quick-Thought

In this part, we compare our method with Infersent [3] and Quick-Thought [39], which have been shown stronger performances in sentence embeddings. For a fair comparison with them, we keep the dimension of word vector as same as our method. In addition, as for Quick-Thought, we use the official pre-training model (https://s3.amazonaws.com/senteval/infersent/infersent1.pkl), other settings are the default values described in https://github.com/facebookresearch/SentEval. As for Quick-Thought, we use also the official pre-training model (https://bit.ly/2uttm2j), and other settings are the default values described in https://github.com/lajanugen/S2V. Table IV lists the comparison results. It can be seen that, (i) compared against Infersent, our method achieves better performance on 17 out of 19 datasets; and (ii) compared against Quick-Thought, our method shows better performance on all these datasets. These results essentially further demonstrate the competitiveness and effectiveness of our proposed method.

E. Impact of μ and λ

Recall Section III-B, our model is involved with two important parameters μ and λ, where μ is a decay factor for the height of the tree, and λ is a decay factor for the length of the child sequences. We here study the impact of these two parameters on the accuracy of our model. Note that, in this set of experiments, we also test two other methods: one is known as CT which did not incorporate the word embedding technique and the attention weight mechanism; the other is known as CVT, which did not incorporate the attention weight mechanism. This way, it might be helpful for us to study the effectiveness of various techniques used in our model. To compute the similarity, CT uses PTK, while CVT uses the simple version of ACVT kernel in which the "weight" part is removed by setting "$Att_{weight} = 1$ in Equation 3". Next, we use the representative results to analyze the performance (notice: other results have the similar performance tendencies, and so we here do not exhaustively plot them, for saving space). Specifically, Figure 4 shows the compared results obtained by using the 13' OnWN dataset.



(a) (b)

Fig. 4. Varying μ and λ on the 13' OnWN dataset.

It can be seen from Figure 4 that ACVT basically outperforms CVT, and CVT basically outperforms CT. This demonstrates that the word embedding technique and the attention weight mechanism are useful when we combine them together. As for ACVT, we can see from Figure 4(a) that, it has the best performance when we set $\mu = 0.2$ or 0.1 (compared to other settings such as $\mu = 0.9$). On the other hand, from Figure 4(b) we can see that our model can obtain best performance when we set $\lambda = 0.1$. These facts justify our default settings for parameter μ and λ, recall Section IV-A.

One could be curious why the curve of ACVT goes down when μ (resp., λ) increases. The main reason could be the followings. When the parameter μ (resp., λ) turns smaller, nodes near to the leaf level (resp., nodes with long child sequences) shall be penalized much more. This way, it makes our model pay more attention to the key information of sentences, which usually located at the upper layers of the ACV-tree. As such, it has higher probability to match people's reading habit (i.e., when people compare two complex sentences, most of people tend toward comparing the main components and essential meaning of the sentences), and thus improves the accuracy.

F. Effectiveness of Syntactic Information

Although we can see clearly from Figure 4 that (1) the constituency tree (CT) alone achieves the weak performance, and (2) *word embedding* and *attention weight mechanism* contribute a lot to the overall performance; it is still necessary

Dataset / Model	2012					2013			2014						2015				
	MSRp	MSRv	SMTe	OnWN	SMTn	hea	OnWN	FNWN	dt-f	dt-n	head	imag	OnWN	tt-n	as-f	as-s	beli	head	imag
QT	0.32	0.83	0.41	0.21	0.31	0.65	0.82	0.49	0.47	0.28	0.57	0.65	0.54	0.64	0.15	0.43	0.28	0.28	0.37
InferSent	0.45	0.74	**0.48**	0.68	**0.58**	0.69	0.73	0.35	0.47	0.67	0.63	0.76	0.73	0.72	0.35	0.60	0.51	0.19	0.68
ACVT	**0.58**	**0.83**	0.43	**0.70**	0.54	**0.77**	**0.85**	**0.50**	**0.48**	**0.74**	**0.72**	**0.81**	**0.87**	**0.75**	**0.69**	**0.79**	**0.70**	**0.79**	**0.82**

TABLE IV
THE COMPARISONS WITH INFERSENT AND QUICK-THOUGHT. NOTICE THAT, WE USE ABBREVIATIONS FOR SOME NAMES, FOR EASE OF ORGANIZING THE TABLE. FOR EXAMPLE, MSRPAR IS WRITTEN AS MSRP.

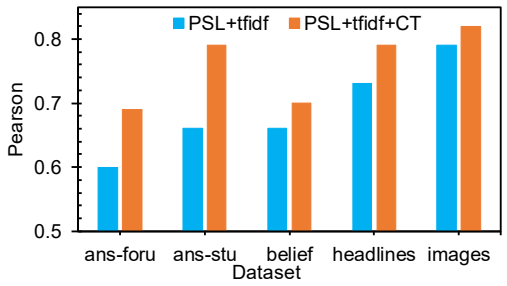


Fig. 5. Effectiveness of syntactic information. In this figure, "ans-foru" means the "answer-forums" dataset, and "ans-stu" means the answer-student dataset, respectively.

Year	Dataset	Methods			
		ACVT	**ACVT**$_{pws}$	**ACVT**$_{d2v}$	**ACVT**$_{sgns}$
2014	deft-forum	**0.48**	0.46	0.47	0.39
	deft-news	**0.74**	**0.74**	0.70	0.65
	headlines	**0.72**	0.71	0.70	0.62
	images	**0.81**	0.80	0.75	0.64
	OnWN	**0.87**	**0.87**	0.86	0.82
	tweet-news	**0.75**	0.74	0.73	0.64
2015	answers-forums	**0.69**	0.62	0.62	0.50
	answers-students	**0.79**	0.60	0.60	0.54
	belief	**0.70**	0.68	0.64	0.53
	headlines	**0.79**	0.78	0.75	0.66
	images	**0.82**	0.80	0.77	0.68

TABLE V
COMPARISON RESULTS OF METHODS THAT USE DIFFERENT WORD EMBEDDINGS.

to investigate the effectiveness of the syntactic information from another viewpoint. To this end, we compare two methods: (1) PSL+tfidf, which uses the TF-IDF (as the attention weight) and PSL vector (cf., Section IV-A) together; and (2) PSL+tfidf+CT (i.e., our proposed method ACVT), which uses three techniques together. Figure 5 shows the representative results obtained by using five datasets from the 15' STS, which are answers-forums, answers-students, belief, headlines, and images. Remark that, we here report them separately, instead of using the average value. This is mainly because the average value might mislead the readers in some cases, due to the lack of individual information.

We can see from Figure 5 that, "PSL+tfidf+CT" always outperforms the "PSL+tfidf". This essentially demonstrates the effectiveness of syntactic information. Meanwhile, we observe that, on most of these datasets the improvements obtained by the syntactic information seem to be not so big. For example, on the "images" dataset, the syntactic information improves only the overall performance by 0.03 (from 0.79 to 0.82); on the "belief" dataset, the syntactic information also improves only the overall performance by 0.03 (from 0.67 to 0.70). Nevertheless, all these results show us that the syntactic information is consistently helpful to improve the overall performance. Remark that, results on other datasets that are not shown (due to space limit), are basically similar to this set of results. That is, the syntactic information is consistently helpful to improve the overall performance.

G. Universal Tests

In this subsection, we test the universality of our model. We use the proposed structure and kernel as the basic framework, and test various attention weight mechanisms and word embedding techniques, respectively. To save space, in our experiments we focus on four word embedding techniques and four attention weight mechanisms mentioned in Section III-D. Nevertheless, other attention weight mechanisms and word embeddings can be seamlessly integrated into our model as well. In our experiments, we use 11 datasets obtained from the STS'14 and STS'15.

In the first group of experiments, four methods are used. (1) ACVT: it is the same as our previous experiments, in which we use PSL vectors [27], which are learned by PPDB and are the Paragram embeddings tuned on SimLex999 dataset [41]. (2) ACVT$_{pws}$: it uses the Paragram-WS353 (PWS) vectors [27], which are Paragram embeddings tuned on WordSim353 dataset [42]. (3) ACVT$_{d2v}$: it uses word embedding obtained by Dict2vec approach [28], which builds new word pairs from dictionary entries so that semantic-related words are moved closer, and negative sampling filters out pairs whose words are unrelated in dictionaries. And (4) ACVT$_{sgns}$: it uses dependency-based embedding [29], which generalizes the skip-gram model with negative sampling (introduced by Mikolov et al. [1]) so as to include the dependency-based contexts.

In the second group of experiments, we also use four methods. (1) ACVT: it is the same as our previous experiments, in which TF-IDF [30], [15] is used as the attention weight. (2) ACVT$_{idf}$: it uses IDF [24] as the attention weight, which is computed as $IDF(w) = log(N/count(N_w))$, where N is the total number of articles, and N_w refers to the number of articles containing the word w; in our experiments we get the IDF value via Wikipedia, and set $IDF(w)$ as $min\{max\{0, IDF(w)\}, 10\}$. (3) ACVT$_{pos}$: it uses the *dot product* of the POS tag vector and the corresponding word embedding [15] as the attention weight; in our experiments, we use the Stanford POS tagger to assign POS tags for words in the training and testing datasets; we obtain POS tag vector with the PPDB dataset (version XL) for 10 epochs, and then train for another 10 epochs on the SICK dataset; as same as in [15], we assign a vector to each POS tag and compute the dot product with the corresponding word embedding vector; the result is a scalar parameter which reflects the importance of the word in the sentence. And (4) ACVT$_{sif}$: it uses the smooth inverse frequency (SIF) [16] as the attention weight, in which SIF is computed as $SIF(w) = \alpha/(\alpha + p(w))$, where α is parameter and $p(w)$ refers to the word frequency; in our experiments, we calculate $p(w)$ from Wikipedia, and set $\alpha = [10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]$, respectively.

Varying word embeddings. Table V shows the performance results when different word embeddings are used (notice: Figure 6 plots these results in a more intuitive manner, for ease of comparison). It can be seen that ACVT achieves the best performance on all these datasets, and ACVT$_{pws}$ achieves the best performance on 2 out of 11 datasets. In addition, we also observe that, on most of these datasets, the performance gap between ACVT and ACVT$_{pws}$ is pretty small. For example, on 4 out of 11 datasets (i.e., 14' and 15' headlines datasets,

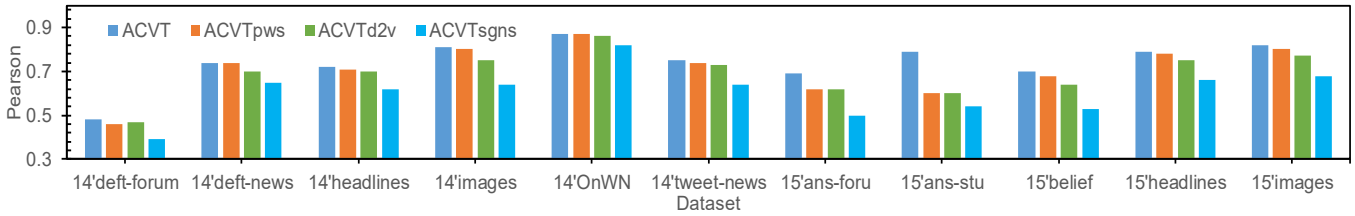


Fig. 6. Visual results of various word embeddings. In this figure, "ans-foru" means the "answer-forums" dataset, and "ans-stu" means the answer-student dataset, respectively.

15' images dataset, and 15' tweet-news dataset), these two methods have only a slight gap of 0.01; on 3 out of 11 datasets (i.e., 14' deft-forum dataset, 15' belief dataset, and 15' images dataset), these two methods have only a gap of 0.02. These results show us that, for the word embedding building block, the PSL vector performs best, and the PWS vector is also competitive in some datasets. Nevertheless, we also observe that on some individual datasets (e.g., 15' answers-students dataset), these two methods have a large performance gap: one is 0.69, and the other is 0.60. The reason could be that the 15' answer-students dataset contains many more dialogue-related words, and the PSL vector may has the stronger ability than PWL vector to represent such words.

As for the ACVT$_{d2v}$ method, its performance is close to that of ACVT$_{pwl}$ and it performs well on most of datasets (e.g., 14' deft-forum dataset, 14' and 15' headlines dataset, 14' tweet-news dataset, 14' OnWN dataset, 15' answers-forums dataset). This demonstrates that D2V vector and PWL vector may have the similar ability to represent words in these datasets. Also, on most of these datasets, the performance gap between ACVT and ACVT$_{d2v}$ is small. For example, on the 14' deft-forum dataset and 14' OnWN dataset, these two methods have only a slight gap of 0.01. In contrast, as we expected, the ACVT$_{sgns}$ method performs relatively poor on these datasets. This is mainly because dependency-based word embedding exhibits more on functional similarities instead of lexical similarities; it could be not very compatible with sentence similarity tasks.

In summary, this set of experiments show us that (1) different word embedding techniques can be applied to our framework, and most of these methods can achieve good performance, demonstrating the universality of our model. (2) in usual cases, the better the word embedding technique, the better the overall performance shall be achieved; this implies that it is possible that one can further improve the performance of our method, when more powerful word embedding techniques are available in the future. (3) some dependency-based embedding techniques (e.g., SGNS) could be not very suitable for sentence similarity tasks, since this type of techniques focus more on functional similarities rather than lexical similarities.

Varying attention weight mechanisms. Since the ACVT$_{sif}$ method involves with a parameter α, we first investigate its impacts on the overall performance of the ACVT$_{sif}$ method, and then compare the variant methods in which different attention weight mechanisms are used.

Table VI reports the experimental results when we vary the parameter α. From this table, one can observe two interesting

Year	Dataset	parameter α				
		10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
2014	deft-forum	0.49	**0.50**	0.48	0.40	0.29
	deft-news	0.76	**0.77**	0.75	0.66	0.56
	headlines	**0.73**	**0.73**	0.72	0.62	0.50
	images	0.76	**0.77**	0.74	0.58	0.34
	OnWN	0.78	0.81	**0.85**	0.79	0.54
	tweet-news	**0.77**	**0.77**	0.74	0.62	0.47
2015	answers-forums	0.70	**0.72**	0.70	0.61	0.43
	answers-students	**0.68**	**0.68**	0.64	0.51	0.34
	belief	**0.72**	**0.72**	0.71	0.62	0.40
	headlines	**0.78**	**0.78**	0.77	0.68	0.51
	images	0.84	**0.85**	0.84	0.75	0.51

TABLE VI
VARYING PARAMETER α FOR THE ACVT$_{sif}$ METHOD.

phenomena: (1) when α is very small (e.g., 10^{-5}, 10^{-6}), the ACVT$_{sif}$ cannot achieve the best performance on all these datasets; (2) the performance results when $\alpha = 10^{-5}$ are consistently better than the ones when $\alpha = 10^{-6}$ on all these datasets. These phenomena essentially illustrate that setting a too small value for α could be inappropriate. It is possible that the word frequency $p(w)$ in these datasets is usually significantly larger than 10^{-5}, and so a too small α shall lower the attention weight. Recall Section IV-G, the ACVT$_{sif}$ method uses the smooth inverse frequency (SIF) as the attention weight, in which SIF is computed as $SIF(w) = \alpha/(\alpha+p(w))$. Thus, it is obvious that the attention weight $SIF(w)$ is to be a small value when α is much smaller than $p(w)$.

Besides the above findings, it can be seen that when $\alpha = 10^{-4}$, the ACVT$_{sif}$ method achieves the best performance on one dataset (this is consistent with the finding in [16]), and it achieves the best performance on five datasets when $\alpha = 10^{-2}$. In contrast, it achieves the best performance on 10 out of 11 dataset when $\alpha = 10^{-3}$. In view of this fact, in the following experiments, we set α to 10^{-3}, unless stated otherwise. To this step, the reader could be curious why the ACVT$_{sif}$ method can achieve the best performance on 10/11 datasets when α is set 10^{-3}, yet it cannot achieve the best performance for the OnWN dataset. We conjecture that the word frequency $p(w)$ in the OnWN dataset could be slightly smaller than that in other datasets, and so the ACVT$_{sif}$ method fails to obtain the best performance when $\alpha = 10^{-3}$. Nevertheless, this does not make a significant impact on our later experiments and analyses.

Table VII reports the experimental results when different attention weight mechanisms are integrated into our framework. Correspondingly, Figure 7 plots these results in a more intuitive manner, for ease of exposition. One can see from

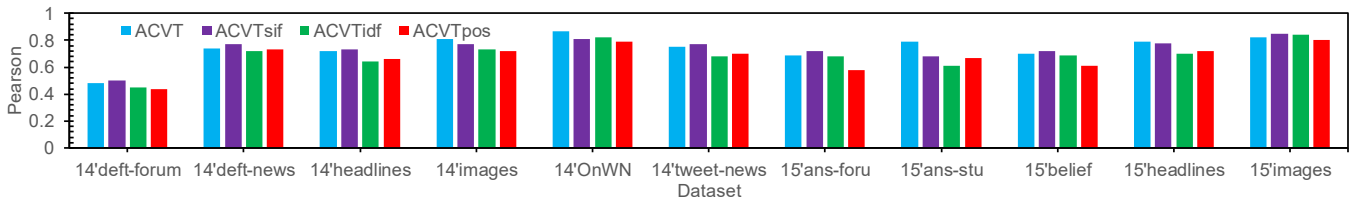


Fig. 7. Visual results of various attention weight mechanisms. In this figure, "ans-foru" means the "answer-forums" dataset, and "ans-stu" means the answer-student dataset, respectively.

Year	Dataset	Methods			
		ACVT	ACVT$_{sif}$	ACVT$_{idf}$	ACVT$_{pos}$
2014	deft-forum	0.48	**0.50**	0.45	0.44
	deft-news	0.74	**0.77**	0.72	0.73
	headlines	0.72	**0.73**	0.64	0.66
	images	**0.81**	0.77	0.73	0.72
	OnWN	**0.87**	0.81	0.82	0.79
	tweet-news	0.75	**0.77**	0.68	0.70
2015	answers-forums	0.69	**0.72**	0.68	0.58
	answers-students	**0.79**	0.68	0.61	0.67
	belief	0.70	**0.72**	0.69	0.61
	headlines	**0.79**	0.78	0.70	0.72
	images	0.82	**0.85**	0.84	0.80

TABLE VII
THE COMPARISON RESULTS OF METHODS THAT USE DIFFERENT ATTENTION WEIGHT MECHANISMS.

Table VII that on the whole these variant methods perform well; and there is no a huge performance degradation arisen by different attention weight mechanisms (cf., Figure 7). This essentially implies that different attention weight mechanisms can be seamlessly integrated into our framework, demonstrating the universality of our model from another point of view. Furthermore, we can see that the ACVT$_{sif}$ method achieves the best performance on 7 out of 11 datasets, and the ACVT method achieves the best performance on 4 out of 11 datasets. Recall Section IV-G, the ACVT method uses the TF-IDF as the attention weight. These phenomena imply that (1) SIF and TF-IDF have favourable performance in expressing the importance of words; and (2) using SIF (or TF-IDF) as the attention weight building block in our model is a strong baseline. This findings are useful for the future studies.

Compared to the above two (variant) methods (i.e., ACVT and ACVT$_{sif}$), the variant method, ACVT$_{idf}$, is slightly inferior to them on (almost) all these datasets. Note that, although ACVT$_{sif}$ is inferior to ACVT$_{idf}$ on the OnWN dataset, we used $\alpha = 10^{-3}$ in this group of experiments. In fact, ACVT$_{sif}$ can achieve a better result if one sets α to 10^{-4} (recall Table VI). On the whole, the performance of ACVT$_{idf}$ is still good (although it is inferior to those strong variant methods). These results also reveal that IDF is slightly inferior to TF-IDF in expressing the important of words. In addition, we can see that, among these four methods, the ACVT$_{pos}$ method performs the worst on the whole. The major reasons could be that, this method trains the POS tag vector and calculates the attention weight by dot product with the corresponding word embedding, yet the POS tag vector trained from the PPDB dataset might not well reflect the semantic importance of different words in the sentence (since the elements contained in the PPDB dataset are almost all phrases instead of sentences). Nevertheless, we conjecture that, if one can develop some targeted preprocessing strategies before calculating the attention weight (based on the POS tag vector and the corresponding word embedding), the ACVT$_{pos}$ method could be further improved.

In summary, this set of experiments show us that (1) replacing different attention weight mechanisms does not incur a huge performance degeneration; this further reflects the universality of our model. (2) TF-IDF and/or SIF could be most suitable for working as the attention weight building block. (3) the ACVT$_{sif}$ method is sensitive to the parameter α, which should be elaborately chosen.

H. Word Embedding Revisited

As we know, Glove [9] and FastText [40] are also two widely used word embedding methods in the literature. The readers could be curious how about if we replace the word embedding used in our method. To this end, we compare our method ACVT (in which PSL vectors are used) with the following invariants: (i) ACVT$_{glv}$ in which the Glove word embedding is adopted, and (ii) ACVT$_{fst}$ in which the FastText word embedding is adopted. As for both Glove and FastText, we use the official pre-trained word vector models available at http://nlp.stanford.edu/data/glove.840B.300d.zip and https://s3-us-west-1.amazonaws.com/fasttext-vectors/wiki-news-300d-1M.vec.zip, respectively. For a fair comparison, the word vector dimensions for both Glove and FastText are set to 300, which is as same as that in ACVT. Table VIII reports the comparison results. One can see that (i) our method ACVT outperforms ACVT$_{glv}$ on all these 19 datasets, and

Model \ Dataset	2012					2013			2014						2015				
	MSRp	MSRv	SMTe	OnWN	SMTn	hea	OnWN	FNWN	dt-f	dt-n	head	imag	OnWN	tt-n	as-f	as-s	beli	head	imag
ACVT$_{glv}$	0.56	0.79	0.34	0.63	0.49	0.69	0.80	0.39	0.39	0.69	0.64	0.70	0.81	0.64	0.50	0.52	0.51	0.70	0.71
ACVT$_{fst}$	0.56	0.79	0.32	0.63	**0.57**	0.66	0.81	0.32	0.39	0.68	0.62	0.68	0.82	0.54	0.51	0.53	0.51	0.67	0.69
ACVT	**0.58**	**0.83**	**0.43**	**0.70**	0.54	**0.77**	**0.85**	**0.50**	**0.48**	**0.74**	**0.72**	**0.81**	**0.87**	**0.75**	**0.69**	**0.79**	**0.70**	**0.79**	**0.82**

TABLE VIII
THE COMPARISONS WITH GLOVE AND FASTTEXT. NOTICE THAT, WE USE ABBREVIATIONS FOR SOME NAMES, FOR EASE OF ORGANIZING THE TABLE. FOR EXAMPLE, MSRPAR IS WRITTEN AS MSRP.

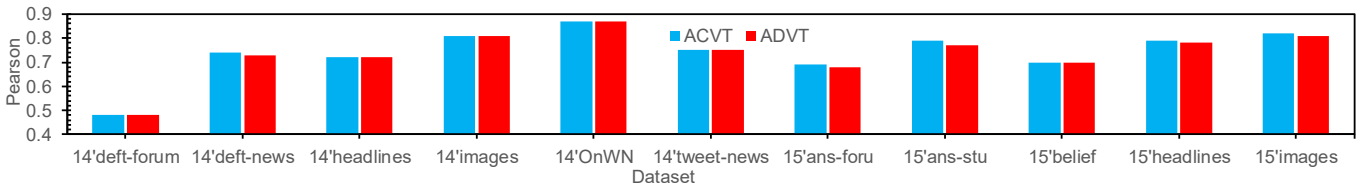


Fig. 8. The comparison results of different parse trees. In this figure, "ans-foru" means the "answer-forums" dataset, and "ans-stu" means the answer-student dataset, respectively.

outperforms ACVT$_{fst}$ on 18 out of 19 datasets; the reason could be that the PSL vector is tuned on SimLex999 dataset, which is better suitable for semantic similarity tasks; (ii) although ACVT$_{glv}$ and ACVT$_{fst}$ are basically dominated by our method, their effects are also good, to some extent. To summarize, this set of experiments further demonstrate the competitiveness and universality of our method.

I. Constituency Tree vs Dependency Tree

As we know, parse trees are usually constructed based on either the constituency relation of constituency grammars or the dependency relation of dependency grammars. The former is usually called the constituency-based parse tree (known as *constituency tree*), while the latter is usually called the dependency-based parse tree (known as *dependency tree*). In previous sections, our proposed model employs the constituency tree structure, and achieves favourable performance. It should be interesting to investigate whether the dependency tree structure has the similar properties. To this end, we adapt our model and get a variant method named ADVT, where "D" is an abbreviation of the word "dependency", and others have the similar meanings with our proposed method (i.e., ACVT). The adaptation is trivial. One can follow the steps of constructing the dependency tree [19], and associate each word with the corresponding attention weight and word vector (as similar as that in Section III-A). Since both the dependency tree and the constituency tree are tree-like structures, the kernel and the algorithm developed in Section III can be immediately used in the ADVT method. Note that, throughout our experiments, the ACVT method uses PSL vectors and TF-IDF. Thus, in this set of experiments, we also use PSL vectors and TF-IDF for the ADVT method.

Figure 8 shows the experimental results. It can be seen that ACVT is not worse than ADVT on all these datasets. This essentially demonstrates that using the constituency tree to build our model could be more reasonable. However, we have to point out that, on the whole the performance of ADVT is no much different from that of ACVT. Specifically, compared with the ACVT method, the ADVT method achieves almost the identical performance on six datasets (including 14' deft-forum, headlines, images, OnWN, tweet-news datasets and also 15' belief dataset). As for those datasets on which ADVT is inferior to ACVT, the performance gap between them is small. For example, there is only a slight gap of 0.01 on three datasets (including 14' deft-news, 15' answers-forums, 15' headlines), and a gap of 0.02 on one dataset (i.e., 15' answers-students) respectively. All these phenomena show us that the dependency tree structure is also a strong candidate for similarity modeling tasks, especially when it is combined with word embedding and attention weight mechanism.

V. Related Work

As mentioned before, own to the success of *word embeddings* [14], [1], [9], [27], much attention has been devoted to sentence similarity modeling via *sentence embeddings* in NLP community. For example, Yu and Dredze [10] used the *simple additional composition* of the word vectors to achieve sentence embeddings. Mitchell and Lapata [43] proposed a framework for representing the meaning of phrases and sentences in vector space. Arora *et al*. [16] obtained the sentence embeddings by *a weighted average* of the word vectors. Kiros *et al.* [22] proposed an encoder-decoder method that can reconstruct the surrounding sentences of an encoded passage. Wieting *et al*. [8] studied the general-purpose sentence embeddings by using the supervision from *paraphrase databases*. Mueller and Thyagarajan [44] presented a siamese adaptation of the LSTM network for labeled data comprised of pairs of variable-length sequences, in which the word-embedding vectors supplemented with synonymic information are provided to the LSTMs. The authors in [21], [45] focused on learning "extra" sentence embeddings, and presented excellent methods. Wieting *et al.* [46] and Takase *et al.* [47] applied the gate mechanism to compute embeddings of phrases and sentences. Wang *et al.* [15] introduced the *attention mechanism* to improve sentence embeddings. Ling *et al.* [48] suggested an improved method to the continuous bag-of-words model, which adds an attention model, for learning word representations. Pham *et al.* [49] applied the Skip-gram algorithm to train embeddings of phrases. Kenter *et al*. [2] presented the *Siamese CBOW model* for efficiently obtaining the high-quality sentence embeddings. Pagliardini *et al.* [50] proposed Sent2Vec, which allows to compose sentence embeddings using the word vectors along with n-gram embeddings. In this line of works, complex nonlinear functions like *convolutional neural networks* [51], [52], [53], [54] were already widely used. Compared to this line of works, our work shares several common features with theirs: (1) our work is also attributed to the development of *word embeddings*, and (2) our work also addresses the issues related to sentence similarity. Nevertheless, our work is different from theirs in one (or both) of features at least: (1) *most* of these works focused on learning semantic information and did not fully take the syntactic information of sentences into account, and (2) tree kernels are not covered in these works. In this paper we take full use of the syntactic information (besides the semantic information). Our model uses a structured manner for sentence similarity modeling;

it integrates semantic information by word embeddings and syntactic information by constituency trees, and develops the ACV-tree kernel to measure similarity, achieving favourable performance (as shown in our experiments).

Another line of works that discussed syntactic and semantic kernels are also related to our work, since our work also encodes syntactic/semantic information represented by means of *tree structures* [55], [26], [18]. For example, Moschitti [26] proposed a new tree kernel, namely the partial tree kernel; their method encodes syntactic parsing information represented by tree structures. Collins and Duffy [56] discussed kernel methods for various natural language structures such as strings, trees, graphs or other discrete structures. Croce *et al.* [17] proposed efficient and powerful kernels for measuring the similarity between dependency structures. Severyn *et al.* [18] proposed a powerful feature-based model that relies on the kernel-based learning and simple tree structures. Additionally, Tian *et al*. [57] presented a tutorial discussing the challenge of composition in distributional and formal semantics. Compared with this line of works, although our work also uses tree structures, our work is different from theirs, since (1) word embeddings are not used in these works, and (2) the attention weight mechanism is not covered in these works. In our paper, the ACV-tree and the corresponding tree kernel are designed to address these issues. As a remark, Tai *et al.* [58] and Zhou *et al.* [59] improved semantic representations from tree-structured LSTM Networks or its variants; these studies also considered the syntactic information. Nevertheless, our work is different from theirs in several points at least: (1) we utilize the tree structure directly after syntactic analysis, while their methods need to learn the syntax information into model(s) through supervised learnings, and (2) the tree kernel is necessary for our method, while their methods do not need to involve this technique.

Besides the above two lines of excellent works, there are also some other nice works such as, sentence classifying [60], [61], [62], sentence clustering [63], sentence ranking [64], new measures and/or metrics to assess sentence similarity [65], [66], [67], etc. These works are clearly different from ours, yet they are complementary to our work. This article is an extended version of the preliminary work [68].

VI. Conclusion

This paper proposed a new method for sentence similarity modeling. The central idea of the proposed model is to combine syntactic information, semantic features, and attention weight mechanism together, absorbing the merits of various techniques. We compared our model with classic and state-of-the-art models on multiple STS tasks, and the results demonstrated that our model can achieve favourable performance, and it has good universality. The major merits of our model are: (1) it can be used as a general framework, since techniques integrated in our model can be viewed as building blocks, allowing users to replace them using other on-shelf techniques or more powerful techniques developed in the future; and (2) unlike most of sentence embedding-based models, our model can be free from time-consuming learning/training, once word embeddings are available; some existing models essentially have also this merit, yet our model outperforms them in terms of performance, further reflecting the superiorities of our model.

References

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[2] T. Kenter, A. Borisov, and M. D. Rijke, "Siamese cbow: Optimizing word embeddings for sentence representations," in *ACL*, 2016, pp. 941–951.

[3] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *EMNLP*, 2017, pp. 670–680.

[4] M. T. Pilehvar and R. Navigli, "From senses to texts: An all-in-one graph-based approach for measuring semantic similarity," *Artif. Intell.*, vol. 228, pp. 95–128, 2015.

[5] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. D. Iii, "Deep unordered composition rivals syntactic methods for text classification," in *ACL*, 2015, pp. 1681–1691.

[6] W. Jiang and J. Wu, "Active opinion-formation in online social networks," in *INFOCOM*, 2017, pp. 1–9.

[7] W. Jiang, J. Wu, and G. Wang, "On selecting recommenders for trust evaluation in online social networks," *ACM Trans. Internet Techn.*, vol. 15, no. 4, pp. 1–21, 2015.

[8] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," in *ICLR*, 2016.

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, 2014, pp. 1532–1543.

[10] M. Yu and M. Dredze, "Learning composition models for phrase embeddings," *TACL*, vol. 3, pp. 227–242, 2015.

[11] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Cambridge: The MIT Press, 1998.

[12] W. Guo and M. T. Diab, "Modeling sentences in the latent space," in *ACL*, 2012, pp. 864–872.

[13] W. Xu, A. Ritter, C. Callison-Burch, W. B. Dolan, and Y. Ji, "Extracting lexically divergent paraphrases from twitter," *TCAL*, vol. 2, pp. 435–448, 2014.

[14] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *JMLR*, vol. 3, pp. 1137–1155, 2003.

[15] S. Wang, J. Zhang, and C. Zong, "Learning sentence representation with guidance of human attention," in *IJCAI*, 2017, pp. 4137–4143.

[16] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," in *ICLR*, 2017.

[17] D. Croce, A. Moschitti, and R. Basili, "Structured lexical similarity via convolution kernels on dependency trees," in *EMNLP*, 2011, pp. 1034–1046.

[18] A. Severyn, M. Nicosia, and A. Moschitti, "Building structures from classifiers for passage reranking," in *CIKM*, 2013, pp. 969–978.

[19] A. Carnie, *Syntax: A Generative Introduction, 3rd Edition*. Wiley-Blackwell, 2012.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[21] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014, pp. 1188–1196.

[22] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," in *NIPS*, 2015, pp. 3294–3302.

[23] M. Barrett, J. Bingel, F. Keller, and A. Søgaard, "Weakly supervised part-of-speech tagging using eye-tracking data," in *ACL*, 2016, pp. 579–584.

[24] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for IDF," *Journal of Documentation*, vol. 60, no. 5, pp. 503–520, 2004.

[25] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015, pp. 1–15.

[26] A. Moschitti, "Efficient convolution kernels for dependency and constituent syntactic trees," in *ECML*, 2006, pp. 318–329.

[27] J. Wieting, M. Bansal, K. Gimpel, K. Livescu, and D. Roth, "From paraphrase database to compositional paraphrase model and back," *TCAL*, vol. 3, pp. 98–104, 2015.

[28] J. Tissier, C. Gravier, and A. Habrard, "Dict2vec: Learning word embeddings using lexical dictionaries," in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, 2017, pp. 254–263.

[29] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, vol. 2, 2014, pp. 302–308.
[30] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.
[31] F. A. Gers and N. N. Schraudolph, "Learning precise timing with lstm recurrent networks," *JMLR*, vol. 3, pp. 115–143, 2002.
[32] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *ACL*, 1994, pp. 133–138.
[33] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *IJCAI*, 1995, pp. 448–453.
[34] D. Lin *et al.*, "An information-theoretic definition of similarity." in *ICML*, 1998, pp. 296–304.
[35] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," in *ROCLING*, 1997, p. 1933.
[36] C. Leacock and M. Chodorow, *Combining local context and WordNet similarity for word sense identification*. Cambridge: The MIT Press, 1998.
[37] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis." in *IJCAI*, 2007, pp. 1606–1611.
[38] C. Manning, D. Jurafsky, and P. Liang, "Stanford parsing tool," https://nlp.stanford.edu/software/, 2017.
[39] L. Logeswaran and H. Lee, "An efficient framework for learning sentence representations," in *ICLR*, 2018, pp. 1–16.
[40] E. Grave, T. Mikolov, A. Joulin, and P. Bojanowski, "Bag of tricks for efficient text classification," in *EACL*, 2017, pp. 427–431.
[41] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, vol. 41, no. 4, pp. 665–695, 2015.
[42] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *WWW*, 2001, pp. 406–414.
[43] J. Mitchell and M. Lapata, "Vector-based models of semantic composition," in *ACL*, 2008, pp. 236–244.
[44] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *AAAI*, 2016, pp. 2786–2792.
[45] Y. Wang, H. Huang, C. Feng, Q. Zhou, J. Gu, and X. Gao, "Cse: Conceptual sentence embeddings based on attention model," in *ACL*, 2016, pp. 505–515.
[46] J. Wieting and K. Gimpel, "Revisiting recurrent networks for paraphrastic sentence embeddings," in *ACL*, 2017, pp. 2078–2088.
[47] S. Takase, N. Okazaki, and K. Inui, "Composing distributed representations of relational patterns," in *ACL*, 2016.
[48] W. Ling, Y. Tsvetkov, S. Amir, R. Fermandez, C. Dyer, A. W. Black, I. Trancoso, and C. Lin, "Not all contexts are created equal: Better word representations with variable attention," in *EMNLP*, 2015, pp. 1367–1372.
[49] N. T. Pham, G. Kruszewski, A. Lazaridou, and M. Baroni, "Jointly
[58] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *ACL*, 2015, pp. 1556–1566.
optimizing word representations for lexical and sentential tasks with the C-PHRASE model," in *ACL*, 2015, pp. 971–981.
[50] M. Pagliardini, P. Gupta, and M. Jaggi, "Unsupervised learning of sentence embeddings using compositional n-gram features," in *NAACL-HLT*, 2018, pp. 528–540.
[51] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *ACL*, 2014, pp. 655–665.
[52] H. He, K. Gimpel, and J. J. Lin, "Multi-perspective sentence similarity modeling with convolutional neural networks," in *EMNLP*, 2015, pp. 1576–1586.
[53] Z. Wang, H. Mi, and A. Ittycheriah, "Sentence similarity learning by lexical decomposition and composition," in *COLING*, 2016, pp. 1340–1349.
[54] Z. Gan, Y. Pu, R. Henao, C. Li, X. He, and L. Carin, "Learning generic sentence representations using convolutional neural networks," in *EMNLP*, 2017, pp. 2390–2400.
[55] A. Wu and K. Lowery, "From prosodic trees to syntactic trees," in *ACL*, 2006, pp. 898–904.
[56] M. Collins and N. Duffy, "Convolution kernels for natural language," in *NIPS*, 2001, pp. 625–632.
[57] R. Tian, K. Mineshima, and P. Martínez-Gómez, "The challenge of composition in distributional and formal semantics," in *IJCNLP*, 2017, pp. 16–17.
[59] Y. Zhou, C. Liu, and Y. Pan, "Modelling sentence pairs with tree-structured attentive encoder," in *COLING*, 2016, pp. 2912–2922.
[60] Y. Xia, Z. Wei, and Y. Liu, "An efficient cross-lingual model for sentence classification using convolutional neural network," in *ACL*, 2016, pp. 126–131.
[61] D. Tang, B. Qin, F. Wei, L. Dong, T. Liu, and M. Zhou, "A joint segmentation and classification framework for sentence level sentiment classification," *IEEE/ACM Trans. Audio, Speech & Language Processing*, vol. 23, no. 11, pp. 1750–1761, 2015.
[62] I. Heo and W. A. Sethares, "Classification based on speech rhythm via a temporal alignment of spoken sentences," *IEEE/ACM Trans. Audio, Speech & Language Processing*, vol. 23, no. 12, pp. 2209–2216, 2015.
[63] A. Skabar and K. Abdalgader, "Clustering sentence-level text using a novel fuzzy relational clustering algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 62–75, 2013.
[64] M. A. Sultan, V. Castelli, and R. Florian, "A joint model for answer sentence ranking and answer extraction," *TACL*, vol. 4, pp. 113–125, 2016.
[65] R. Ferreira, R. D. Lins, F. Freitas, S. J. Simske, and M. Riss, "A new sentence similarity assessment measure based on a three-layer sentence representation," in *ACM Symposium on Document Engineering*, 2014, pp. 25–34.
[66] Y. Ji and J. Eisenstein, "Discriminative improvements to distributional sentence similarity," in *EMNLP*, 2013, pp. 891–896.
[67] C. Spiccia, A. Augello, G. Pilato, and G. Vassallo, "Semantic word error rate for sentence similarity," in *ICSC*, 2016, pp. 266–269.
[68] Y. Le, Z. Wang, Z. Quan, J. He, and B. Yao, "Acv-tree: A new method for sentence similarity modeling," in *IJCAI*, 2018, pp. 4137–4143.