# ACV-tree: A New Method for Sentence Similarity Modeling

**Yuquan Le[†], Zhi-Jie Wang[‡], Zhe Quan[†,*], Jiawei He[†] and Bin Yao[#]**
[†] College of Computer Science and Electronic Engineering, Hunan University, Changsha, China
[‡] Guangdong Key Lab. of Big Data Anal. and Proc., Sun Yat-Sen University, Guangzhou, China
[#] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
{leyuquan,quanzhe,hejiawei}@hnu.edu.cn, wangzhij5@mail.sysu.edu, yaobin@cs.sjtu.edu.cn

## 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 have achieved sentence embedding, obtaining attractive performance. Nevertheless, most of them focused on learning semantic information and modeling it as a continuous vector, while 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 makes the first attempt 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.

## 1 Introduction

Sentence similarity is a fundamental metric to measure the degree of likelihood between a pair of sentences [Mikolov *et al.*, 2013; Kenter *et al.*, 2016; Conneau *et al.*, 2017], and plays an important role for many applications [Pilehvar and Navigli, 2015; Iyyer *et al.*, 2015; Jiang and Wu, 2017; Jiang *et al.*, 2015]. Measuring sentence similarity is challenging due to the ambiguity and variability of linguistic expression, and thus has received much attention in recent years [Wieting *et al.*, 2016; Pennington *et al.*, 2014; Yu and Dredze, 2015]. 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 [Fellbaum, 1998] and corpus-based [Guo and Diab, 2012]. Some methods also used the combination of various features and multi-task learning [Xu *et al.*, 2014].

Recently, owing to the success of *word embeddings* [Bengio *et al.*, 2003; Mikolov *et al.*, 2013], researchers have attempted to study sentence similarity modeling via *sentence embeddings*. This approach has become a successful paradigm in natural language processing (NLP) community [Kenter *et al.*, 2016; Wang *et al.*, 2017]; and particularly some studies have used the *attention weight mechanism* to further enhance the performance [Wang *et al.*, 2017; Arora *et al.*, 2017]. In this line of works, most 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 [Croce *et al.*, 2011; Severyn *et al.*, 2013]. 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. In this regard, it is the same as the methods in [Wieting *et al.*, 2015; 2016], which are word embedding-based models. Nevertheless, our model can achieve better performance on almost all datasets used in our experiments, compared against the word-embedding based models.

To summarize, the main contributions of this paper are:

- We propose a new structure for sentence similarity modeling. Our model wisely combines syntactic informa-

[*] means the corresponding author.

tion, 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 (Section 3.1).

- We developed the ACVT kernel that can allow us to efficiently perform similarity measure based on the proposed structure (Section 3.2).
- We conduct extensive experiments based on widely-used benchmark datasets (Section 4). The experimental results consistently demonstrate the superiorities and competitiveness of our proposed model.

## 2 Background

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

*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 [Carnie, 2012]. Each node in the tree is either a root node, a 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 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.

*Word embedding:* Recently, a popular framework can allow users to represent words as continuous vectors that capture lexical and semantic properties of words [Bengio *et al.*, 2003; Mikolov *et al.*, 2013]. Usually, this technique is known as *word embedding* (a.k.a., *distributed vector representation of words*). Figure 2 shows an example of this 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 training 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 $W$.

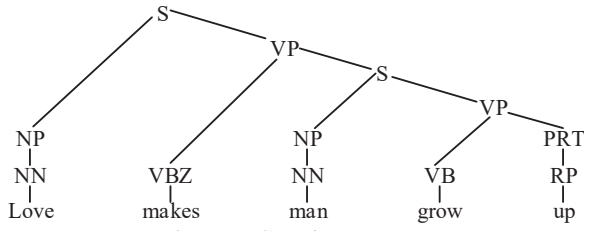


Figure 1: Constituency tree .

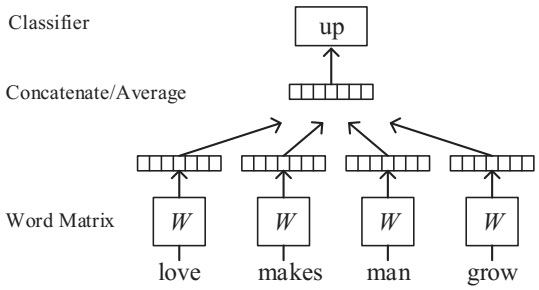


Figure 2: Framework of word embedding.

*Attention weight mechanism:* Most of neural network based sentence representation models treat each word in sentences equally [Le and Mikolov, 2014; Kiros *et al.*, 2015; Wieting *et al.*, 2016]. 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 [Barrett *et al.*, 2016]. Thereby, researchers have considered to assign words with different weights (known as *attention weight mechanism*), and there are many schemes to assign attention weights to words, such as *smooth inverse frequency* (SIF), *term frequency-inverse document frequency* (TF-IDF), *Surprisal* (SUR), *POS tag* (POS), *CCG supertag* (CCG) [Wang *et al.*, 2017; Arora *et al.*, 2017].

*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 [Moschitti, 2006]. 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) [Moschitti, 2006], is highly related to our proposed tree kernel. Specifically, the $\Delta(\cdot)$ function of PTK is com-

puted 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 $\mu$ and $\lambda$ are two decay factors: $\mu$ for the height of the tree, and $\lambda$ 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., $n_2$); $l_m = min\{\text{length}(c_{n_1}), \text{length}(c_{n_2})\}$; and $\Delta_p(\cdot)$ refers to the number of common subsequence whose length is $p$.

## 3 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.

### 3.1 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. 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* of each word in the sentence (e.g., in the sentence "a beautiful girl is over there", the word *beautiful* modifies the word *girl*). Finally, we link items according to the modification relation 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.

### 3.2 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 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

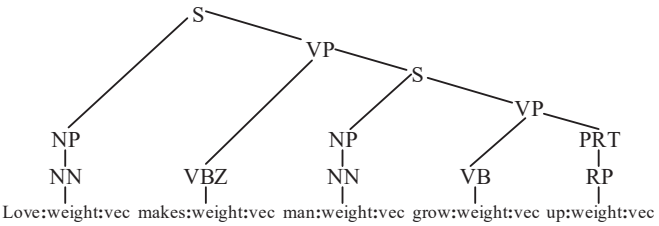



Figure 3: An example of ACV-tree

of node $n_1$ (resp., $n_2$). Our $\Delta(n_1, n_2)$ function is inspired by that of PTK (recall Section 2). 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 $c_{n_1}$, $c_{n_2}$, $l_m$, $\mu$ and $\lambda$ have the same meaning as mentioned in Section 2; $vec_1$ 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; $Att_{weight} = wt_1 \times wt_2$.

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.

**Algorithm 1** COMPSIM($T_1$,$T_2$)

```
for each node n_i in T_1 do
    if isLeaf(n_i) = true then
        for each node n_j in T_2 do
            if isLeaf(n_j) = true then
                K[n_i][n_j] ← Att_weight × SIM(vec_1, vec_2);
            else
                K[n_i][n_j] ← 0;
    else
        for each node n_j in T_2 do
            if isLeaf(n_j) = true then
                K[n_i][n_j] ← 0;
            else
                if n_i has the same structure with n_j then
                    K[n_i][n_j] ← COMPSIM(n_i, n_j);
                else
                    K[n_i][n_j] ← 0;
S_{T_1T_2} ← compute the sum of K and normalize it;
return S_{T_1T_2}
```

*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 [Moschitti, 2006]. 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 trees and compute their similarity (see Line 14). The final step is the same as that in [Moschitti, 2006], namely, we compute the sum of all values in $\boldsymbol{K}$ and normalize it, obtaining the similarity score.

## 4 Experiments

This section covers our experimental results. Our codes are available at the open-source code repository (https://github.com/yuquanle/Sentence-similarity-modeling.git).

### 4.1 Datasets and Experimental Settings

*Datasets:* Following prior works, we conduct experiments on 19 textual similarity datasets (http://ixa.si.ehu.eus/) 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 1 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 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 [Pennington *et al.*, 2014], PSL [Wieting *et al.*, 2015], ST [Kiros *et al.*, 2015], SCBOW [Kenter *et al.*, 2016], PROJ [Wieting *et al.*, 2016], PP-tfidf [Wang *et al.*, 2017], DAN [Iyyer *et al.*, 2015], LSTM [Gers and Schraudolph, 2002], RNN and iRNN [Wieting *et al.*, 2016]. The results of the above methods are collected from [Wieting *et al.*, 2016] except SCBOW from [Kenter *et al.*, 2016] and PP-tfidf from [Wang *et al.*, 2017].

(2) The models that are developed based on other techniques, including WUP [Wu and Palmer, 1994], RES [Resnik, 1995], LIN [Lin and others, 1998], JCN [Jiang and Conrath, 1997], and LCH [Leacock and Chodorow, 1998], ESA [Gabrilovich and Markovitch, 2007], ADW [Pilehvar and Navigli, 2015]. 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 [Pilehvar and Navigli, 2015].

*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 [Pilehvar and Navigli, 2015; Wang *et al.*, 2017; Kenter *et al.*, 2016]. 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 hyperparameters $\mu$=[0.1,**0.2**,...,0.9,1.0], and $\lambda$ =[**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* [Manning *et al.*, 2017] 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 [Arora *et al.*, 2017; Wang *et al.*, 2017], 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. The lexical vectors we used are provided by PARAGRAM-SL999 vectors, which is learned by PPDB and is the 300 dimensional Paragram embeddings tuned on SimLex999 dataset [Wieting *et al.*, 2015].

| STS'12 | STS'13 | STS'14 | STS'15 |
|---|---|---|---|
| MSRpar | headlines | deft-forum | answers-forums |
| MSRvid | OnWN | deft-news | answers-students |
| SMTeuroparl | FNWN | headlines | belief |
| OnWN | SMT | images | headlines |
| SMTnews | | OnWN | images |
| | | tweet-news | |

Table 1: Datasets for the SemEval Semantic Textual Similarity Tasks (year 2012 — year 2015).

### 4.2 Experimental Results

In what follows, we first compare with the methods that used word/sentence embedding techniques, since these methods are closest to our proposed model. Then, we check whether our model can beat classic methods. Finally, we study the impact of important parameters.

*Word/sentence embedding-based methods:* Table 2 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 [Wieting *et al.*, 2016]). 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' SMTeuroparl 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' deft-forum dataset con-

| 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 2: The comparison results; the bold number highlights one of strongest results in each dataset.

tains 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.

*Classic methods:* As same as to [Pilehvar and Navigli, 2015], we here also use the SemEval-2012 Semantic Similarity task to compare these classic methods. Table 3 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.

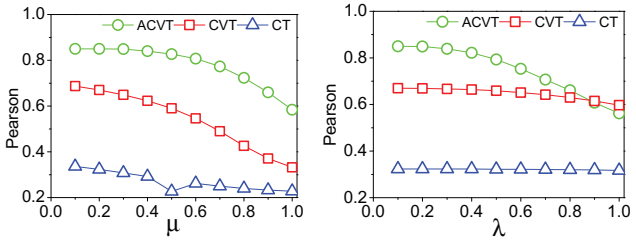
*Impact of parameters:* Recall Section 3.2, our model is involved with two important parameters $\mu$ and $\lambda$, where $\mu$ is a decay factor for the height of the tree, and $\lambda$ is a decay factor for the length of the child sequences. We here study the impact of these two parameters on the 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. Specifically, Figure 4 shows the compared results, these results are obtained by using the 13' OnWN dataset.

| 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 3: The comparison between our method and classic methods.

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 $\mu$ and $\lambda$, recall Section 4.1.

One could be curious why the curve of ACVT goes down when $\mu$ (resp., $\lambda$) increases. The main reason could be the followings. When the parameter $\mu$ (resp., $\lambda$) 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,




(a) (b)

Figure 4: Varying $\mu$ and $\lambda$ on the 13' OnWN dataset.

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.

## 5 Related Work

As mentioned before, own to the success of *word embeddings* [Bengio *et al.*, 2003; Mikolov *et al.*, 2013; Pennington *et al.*, 2014; Wieting *et al.*, 2015], much attention has been devoted to sentence similarity modeling via *sentence embeddings* in NLP community. For example, [Yu and Dredze, 2015] used the *simple additional composition* of the word vectors to achieve sentence embeddings. [Arora *et al.*, 2017] obtained the sentence embeddings by *a weighted average* of the word vectors. [Kiros *et al.*, 2015] proposed an encoder-decoder method that can reconstruct the surrounding sentences of an encoded passage. [Wieting *et al.*, 2016] studied the general-purpose sentence embeddings by using the supervision from *paraphrase databases*. [Le and Mikolov, 2014; Wang *et al.*, 2016] focused on learning "extra" sentence embeddings. [Wang *et al.*, 2017] introduced the *attention mechanism* to improve sentence embeddings. [Kenter *et al.*, 2016] presented the *Siamese CBOW model* for efficiently obtaining the high-quality sentence embeddings. In this line of works, complex nonlinear functions like *convolutional neural networks* [Kalchbrenner *et al.*, 2014; Gan *et al.*, 2017], and *recurrent neural networks* [Tai *et al.*, 2015] 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 several 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* [Wu and Lowery, 2006; Moschitti, 2006; Severyn *et al.*, 2013]. For example, [Moschitti, 2006] proposed a new tree kernel, namely the partial tree kernel; their method encodes syntactic parsing information represented by tree structures. [Collins and Duffy, 2001] discussed kernel methods for various natural language structures such as strings, trees, graphs or other discrete structures. [Croce *et al.*, 2011] proposed efficient and powerful kernels for measuring the similarity between dependency structures. [Severyn *et al.*, 2013] proposed a powerful feature-based model that relies on the kernel-based learning and simple tree structures. 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.

## 6 Conclusions

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. 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.

## Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2016YFB0201900), and the NSFC (No. 61602166, 61772183, and 61572179).

## References

[Arora *et al.*, 2017] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017.

[Barrett *et al.*, 2016] Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. Weakly supervised part-of-speech tagging using eye-tracking data. In *ACL*, pages 579–584, 2016.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *JMLR*, 3:1137–1155, 2003.

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

[Collins and Duffy, 2001] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *NIPS*, pages 625–632, 2001.

[Conneau *et al.*, 2017] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. pages 670–680, 2017.

[Croce *et al.*, 2011] Danilo Croce, Alessandro Moschitti, and Roberto Basili. Structured lexical similarity via convolution kernels on dependency trees. In *EMNLP*, pages 1034–1046, 2011.

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

[Gabrilovich and Markovitch, 2007] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, pages 1606–1611, 2007.

[Gan *et al.*, 2017] Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. Learning generic sentence representations using convolutional neural networks. In *EMNLP*, pages 2390–2400, 2017.

[Gers and Schraudolph, 2002] Felix A. Gers and Nicol N. Schraudolph. Learning precise timing with lstm recurrent networks. *JMLR*, 3:115–143, 2002.

[Guo and Diab, 2012] Weiwei Guo and Mona T. Diab. Modeling sentences in the latent space. In *ACL*, pages 864–872, 2012.

[Iyyer *et al.*, 2015] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé Iii. Deep unordered composition rivals syntactic methods for text classification. In *ACL*, pages 1681–1691, 2015.

[Jiang and Conrath, 1997] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *ROCLING*, page 19–33, 1997.

[Jiang and Wu, 2017] Wenjun Jiang and Jie Wu. Active opinion-formation in online social networks. In *INFOCOM*, pages 1–9, 2017.

[Jiang *et al.*, 2015] Wenjun Jiang, Jie Wu, and Guojun Wang. On selecting recommenders for trust evaluation in online social networks. *ACM Trans. Internet Techn.*, 15(4):1–21, 2015.

[Kalchbrenner *et al.*, 2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665, 2014.

[Kenter *et al.*, 2016] Tom Kenter, Alexey Borisov, and Maarten De Rijke. Siamese cbow: Optimizing word embeddings for sentence representations. In *ACL*, pages 941–951, 2016.

[Kiros *et al.*, 2015] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. In *NIPS*, pages 3294–3302, 2015.

[Le and Mikolov, 2014] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[Leacock and Chodorow, 1998] Claudia Leacock and Martin Chodorow. *Combining local context and WordNet similarity for word sense identification*. The MIT Press, Cambridge, 1998.

[Lin and others, 1998] Dekang Lin et al. An information-theoretic definition of similarity. In *ICML*, pages 296–304, 1998.

[Manning *et al.*, 2017] Chris Manning, Dan Jurafsky, and Percy Liang. Stanford parsing tool. https://nlp.stanford.edu/software/, 2017.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Moschitti, 2006] Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, pages 318–329, 2006.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[Pilehvar and Navigli, 2015] Mohammad Taher Pilehvar and Roberto Navigli. From senses to texts: An all-in-one graph-based approach for measuring semantic similarity. *Artif. Intell.*, 228:95–128, 2015.

[Resnik, 1995] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[Severyn *et al.*, 2013] Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. Building structures from classifiers for passage reranking. In *CIKM*, pages 969–978, 2013.

[Tai *et al.*, 2015] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566, 2015.

[Wang *et al.*, 2016] Yashen Wang, Heyan Huang, Chong Feng, Qiang Zhou, Jiahui Gu, and Xiong Gao. Cse: Conceptual sentence embeddings based on attention model. In *ACL*, pages 505–515, 2016.

[Wang *et al.*, 2017] Shaonan Wang, Jiajun Zhang, and Chengqing Zong. Learning sentence representation with guidance of human attention. In *IJCAI*, pages 4137–4143, 2017.

[Wieting *et al.*, 2015] John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. From paraphrase database to compositional paraphrase model and back. *TCAL*, 3:98–104, 2015.

[Wieting *et al.*, 2016] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. In *ICLR*, 2016.

[Wu and Lowery, 2006] Andi Wu and Kirk Lowery. From prosodic trees to syntactic trees. In *ACL*, pages 898–904, 2006.

[Wu and Palmer, 1994] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *ACL*, pages 133–138, 1994.

[Xu *et al.*, 2014] Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. Extracting lexically divergent paraphrases from twitter. *TCAL*, 2:435–448, 2014.

[Yu and Dredze, 2015] Mo Yu and Mark Dredze. Learning composition models for phrase embeddings. *TACL*, 3:227–242, 2015.