# SCIR-QA at SemEval-2017 Task 3: CNN Model Based on Similar and Dissimilar Information between Keywords for Question Similarity

**Le Qi, Yu Zhang, Ting Liu**
Research Center for Social Computing
and Information Retrieval
Harbin Institute of Technology
lqi,zhangyu,tliu@ir.hit.edu.cn

## Abstract

We describe a method of calculating the similarity between questions in community QA. Questions in cQA are usually very long and there are a lot of useless information about calculating the similarity between questions. Therefore, we implement a CNN model based on similar and dissimilar information on questions keywords. We extract the keywords of questions, and then model the similar and dissimilar information between the keywords, and use the CNN model to calculate the similarity.

## 1 Introduction

We participate in SemEval-2017 Task 3 Subtask B (Nakov et al., 2017) on Community Question Answering. In this task, we are given a question from community forum (named original question) and 10 related questions. We need to re-rank the related questions according to their similarity between the origin question.

Both the original question and the related question have question subject and question body. The subject is short. The body is long and contains a lot of useless information. In our system, we try to use keywords to replace questions to locate more important information on the question, so we use a keyword extraction algorithm that combines syntactic information to get more accurate keywords. Then we use a CNN model based on similar and dissimilar information between questions to calculate the similarity of questions. The model can make good use of similar information and dissimilar information between questions to get better results.

The paper is organized as follows: Section 2 introduces our system. Section 3 introduces the experiment. And in section 4, there are the conclusions.

## 2 Model

In this section we describe our system in detail. In Section 2.1 we show how we extract keywords from the subject and body, and then in Section 2.2 we describe how to construct the CNN model based on similar and dissimilar information on question keywords.

### 2.1 Keyword extraction

First, we cut the question subject and question body. Then, we extract keywords from each subsentence. We combine all the extracted keywords together as a result.

We use an unsupervised keyword extraction method based on dependency analysis. The method uses syntactic dependency relations between words as clues. For the given question, we not only use the statistical information and word vector information, but also construct the dependency graph to calculate the correlation intensity between words, and then construct the weighted graph according to the dependency degree, and use the TextRank algorithm (Mihalcea and Tarau, 2004) to iterate to calculate the word importance score. The main steps include preprocessing, the construction of the non-directional weighted graph, graph ranking, and the selection of the t words with the highest score as keywords of the question, as shown in Figure 1.

**Preprocess**: The preprocessing process includes word segmentation and removing the stop words. We use the remaining words as the candidate words of the keywords.

**Construct the undirected weighted graph**: After preprocessing, all candidate words are represented as vertices of the graph. If two words cooccur in a sentence, there is an edge to the two



*Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 305–309,
Vancouver, Canada, August 3 - 4, 2017. 

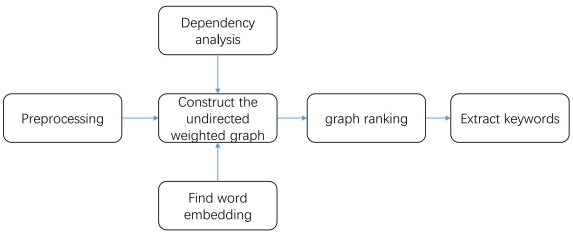



Figure 1: Keywords extraction

vertices. The weight of the edge is calculated by the statistical information on words, the word vector information and the dependent syntax analysis information.

The methods that can be used to calculate the correlation between two words are: Pointwise Mutual Information (PMI), Average Mutual Information(AMI) (Terra and Clarke, 2004), etc. However, these methods only consider the statistical information between words, and do not consider the syntactic dependencies. The syntactic dependency between words has a positive effect on measuring the importance of words.

The result of the dependency syntax analysis is analogous to the tree structure. If we remove its root node, and ignore the arc of the point, we can get an undirected dependency diagram $G' = (V', E'), V' = w_1, w_2, ..., w_n, E' = e_1, e_2, ..., e_m$, where $w_i$ denotes a word and $e_j$ denotes an undirected relationship between two words. The undirected dependency graph guarantees that there is a dependency path between any two words in the question, and the length of the dependency path reflects the intensity of the dependency relationship. Therefore, we introduce the concept of dependency degree according to the length of the dependent path (Zhang et al., 2012), as shown in Equation(1), where $dr\_path\_len(w_i, w_j)$ represents the dependency path length between words $w_i$ and $w_j$, b is the superparameter.

$$Dep(w_i, w_j) = \frac{1}{b^{dr\_path\_len(w_i, w_j)}} \quad (1)$$

The degree of correlation between two words, that is, the weight of the edge is multiplied by the gravitational value of the two words by the length of the dependent path, as shown in Equation(2).

$$weight(w_i, w_j) = Dep(w_i, w_j) * f(w_i, w_j) \quad (2)$$

Among them, the concept of gravitational values proposed by (Wang et al., 2015), inspired by gravitation. The word frequency is regarded as the object mass, and the distance between the words is taken as the distance of the object. The gravitational value $f(w_i, w_j)$ of the two words is given by the Equation(3).

$$f(w_i, w_j) = \frac{freq(w_i) * freq(w_j)}{d^2} \quad (3)$$

**Graph ranking**: We use the weighted TextRank algorithm to sort the graph. In the undirected graph $G = (V, E)$, V is the set of vertices, E is the set of edges, and $C(v_i)$ is the set of vertices connected to the vertex $v_i$. The score of the vertex $v_i$ is calculated from the Equation(4), where $weight(w_i, w_j)$ is calculated from the Equation(3), d is the damping coefficient.

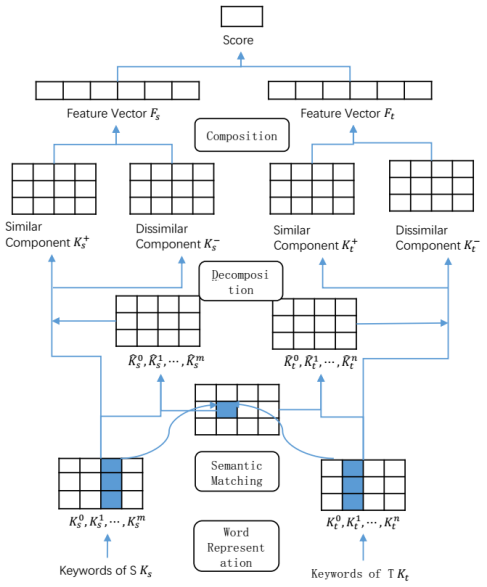
$$ws(v_i) = (1-d) + d * \sum_{v_j \in C(v_i)} \frac{weight(v_i, v_j)}{\sum_{v_k \in C(v_j)} weight(v_j, v_k)} \quad (4)$$

Then we select the t words with the highest score as the keywords.

## 2.2 CNN model based on similar and dissimilar information

We use a CNN model based on similar parts and dissimilar parts between two sentences to get sentence similarity. This model is proposed by (Wang et al., 2016), now we will introduce the model briefly. Figure 2 shows the structure of the model.

Given a sentence pair, the model represents each keyword as a vector, and calculates a semantic matching vector for each keyword based on part of keywords in the other sentence. Then each word vector is decomposed into two components based on the semantic matching vector: a similar component and a dissimilar component. After this, we use a two-channel CNN to compose the similar and dissimilar components into a feature vector. Finally, a fully connected neural network is used to predict the sentence similarity through the composed feature vector.




First, with word embedding pre-trained by Stanford using GloVe's model (Pennington et al., 2014), we transform keywords of question S and T into matrix $S = [s_1, s_2, ..., s_m]$ and $T = [t_1, t_2, ..., t_n]$, where $s_i$ and $t_j$ are 300-dimention vectors of corresponding keywords, and m and n are the length of keywords of S and T. Second, for judging the similarity between two sentences, we check whether each keyword in one sentence can be covered by the other sentence. For a sentence pair S and T, we first calculate a similarity matrix $A_{(m\times n)}$, where each element $a_{(i,j)} \in A_{(m\times n)}$ computes cosine similarity between words $s_i$ and $t_j$ as

$$a_{(i,j)} = \frac{s_i^T t_j}{||s_i|| \cdot ||t_j||} \quad \forall s_i \in S, \forall t_j \in T \qquad (5)$$

We calculate a semantic matching vector $\hat{s_i}$ for each word $s_i$ by composing part of word vectors in the other sentence T. In this way, we can match a keyword $s_i$ to some keywords in T. Similarly, we also calculate all semantic matching vectors $\hat{t_i}$ in T. We define a semantic matching functions over $A_{(m\times n)}$

$$f_{match}(s_i, T) = \frac{\sum_{j=k-w}^{k+w} a_{i,j} t_j}{\sum_{j=k-w}^{k+w} a_{i,j}} \qquad (6)$$

where

$$k = argmax_j a_{i,j}$$

w indicates the size of the window to consider centered at k (the most similar word position). So the semantic matchisng vector is a weighted average vector from $t_{k-w}$ to $t_{k+w}$.

Third, after semantic matching, we have the semantic matching vectors of $\hat{s_i}$ and $\hat{t_j}$. Take s as an example. We interpret $\hat{s_i}$ as a semantic coverage of word $s_i$ by the sentence T. However, there must be some difference between $s_i$ and $\hat{s_i}$. So based on its semantic matching vector $\hat{s_i}$, our model further decomposes word $s_i$ into two components: similar component $\hat{s_i}^+$ and dissimilar component $\hat{s_i}^-$. Then we choose a linear decomposition method. The motivation for the linear decomposition is that the more similar between $s_i$ and $\hat{s_i}$, the higher proportion of $s_i$ should be assigned to the similar component. First, we calculate the cosine similarity between $s_i$ and $\hat{s_i}$. Then, we decompose $s_i$ linearly based on $\alpha$. Eq.(7) gives the corresponding definition:

$$a_{(i,j)} = \frac{s_i^T \hat{s_i}}{||s_i|| \cdot ||\hat{s_i}||}$$

$$\hat{t_j}^+ = \alpha s_i$$

$$\hat{s_i}^- = (1-\alpha) s_i \qquad (7)$$

Finally, due to the dissimilar and similar components have strong connections, we use a two-channel CNN model (Kim, 2014) to compose them together. In the CNN model, we have three layers. The first is a convolution layer. We define a list of filters $w_o$. The shape of each filter is d h, where d is the dimension of word vectors and h is the window size. Each filter is applied to two patches (a window size h of vectors) from both similar and dissimilar channels, and generates a feature. Eq.(8) expresses this process:

$$c_{o,i} = f(w_o * S^+_{[i:i+h]} + w_o * S^-_{[i:i+h]} + b_o) \qquad (8)$$

The second layer is a pooling layer. We choose max-pooling method to deal with variable feature size. And the last layer is a full-connected layer. We use a sigmoid function to constrain the result within the range [0,1].

## 3 Experiment

We experimented with the corpus provided by SemEval-2017 task3. Training set has 267 questions, each question has 10 related questions, a total of 2670 question pairs. Development set has 50 questions, 500 question pairs. The test set has 88 questions, 880 question pairs. We do the experiment without preprocessing. We use Stanfod Parser (De Marneffe and Manning, 2008) to parse

sentences. And we use the keyword extraction algorithm described in 2.1, for each sub-sentence we extract 1/3 of the words as keywords and set b = 1.4, d = 0.8. In the CNN model, we set up the filter shape is 3*300. The number of filters is 500. We set the similarity threshold of 0.5, that is, a score greater than 0.5 is considered a positive case. And we set the learning rate as 0.001. After 20 rounds of training, we got the result in devlopment set and test set.

| Team | MAP | AvgRec | MRR |
|---|---|---|---|
| Baseline(IR) | 41.85 | 77.59 | 46.42 |
| Baseline(Random) | 29.81 | 62.65 | 33.02 |
| simbow | 47.22 | 82.60 | 50.07 |
| LearningToQuestion | 46.93 | 81.29 | 53.01 |
| SCIR-QA | 42.72 | 78.24 | 46.65 |

Table 1: Test Result

| User or Team Name | MAP | AvgRec | MRR |
|---|---|---|---|
| Sagustian | 79.6 | 94.3 | 86.0 |
| BeiHang | 76.9 | 91.2 | 83.5 |
| naman | 75.1 | 90.8 | 81.33 |
| LS2NSEMEVAL | 74.4 | 88.3 | 79.5 |
| NLMNIH | 73.7 | 88.2 | 79.33 |
| IIT-UHH | 73.6 | 89.0 | 79.33 |
| Organizers | 71.4 | 86.1 | 76.67 |
| MIT-QCRI | 71.4 | 86.1 | 76.67 |
| SCIR-QA | 70.8 | 87.5 | 77.25 |
| preslav | 55.9 | 73.2 | 62.23 |

Table 2: Develop Result

The results in test set are shown in Table 1, the first two lines are the baseline, the next two lines are the best results, the last line is our result. And results in development set are shown in Table 2. In test set, our results are better than the baseline, but there is still some distance from the best results. In development set, our result is all not so good.

We think that because we do the experiment without preprocessing, there exists too many unknown words in word embeddings, which results in poor system performance. On the other hand, because the training corpus is too small, the neural network can not be well trained and can not find meaningful features. Therefore, in the future work, we will add features of artificial extraction into neural network to improve performance. And we will add features of artificial extraction into neural network to improve performance.

## 4 Result and Future Work

We implement a CNN model based on similar and dissimilar information between questions keywords, and experiment on SemEval-2017 corpus. The experimental results show that our method is better than baseline, we can extract the key information from the long sentence to model the question better, which helps us to calculate the similarity of the question. We think that keyword extraction is important in this task, and in the future we will try other keyword extraction methods to achieve better results.

## Acknowledgments

This work was supported by the National High Technology Development 863 Program of China (No.2015AA015407), National Natural Science Foundation of China (No.61472105 and No.61472107). And I am grateful to Professor Zhang Yu for his guidance to my work. And I also appreciate the support of SCIR laboratory.

## References

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. *Unt Scholarly Works* pages 404–411.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada, SemEval '17.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Egidio Terra and C. L. A Clarke. 2004. Frequency estimates for statistical word similarity measures. In *Naacl 03 Conference of the North American Chapter of the Association for Co*. pages 244–251.

Rui Wang, Wei Liu, and Chris McDonald. 2015. Corpus-independent generic keyphrase extraction using word embedding vectors.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *COLING*.

Weinan Zhang, Zhaoyan Ming, Yu Zhang, Liqiang Nie, Ting Liu, and Tat-Seng Chua. 2012. The use of dependency relation graph to enhance the term weighting in question retrieval. In *COLING*. pages 3105–3120.