ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 3, No. 5, 1-5 (2018)
www.astesj.com
Special Issue on Multidisciplinary Sciences and Engineering

ASTES Journal
ISSN: 2415-6698

Evaluation of Methods for Sentence Similarity for Use in Intelligent Tutoring System

Emil Brajković[*], Daniel Vasić, Tomislav Volarić

Faculty of Science and Education, Department of Informatics, University of Mostar, 88 000, Bosnia and Herzegovina

A R T I C L E I N F O

Article history:
Received: 04 July, 2018
Accepted: 20 August, 2018
Online: 05 September, 2018

Keywords:
Intelligent Tutoring Systems
Natural Language Processing
Natural Language Generation
Sentence Similarity
Vector Space Models
Tree Edit Distance
Fuzzy Analytic Hierarchy Process

A B S T R A C T

Finding similarity of textual data is very important task in natural language processing. In this article we present approach to finding similarity of words, paragraphs, sentences and documents. Semantic similarity is one of the central tasks in many applications, including text summarization, Intelligent Tutoring Systems (ITS) etc. In ITS sentence similarity is used to compare the student's response with the correct answer. The result is used to gain information about student's level of knowledge. We propose three different methods that measure text to text semantic relatedness. There are multiple approaches to finding the right measure to determine the similarity of the sentences. Some measure the alignment of characters, and other measure semantic similarity between sentences. In this work we present and evaluate methods for finding not just similarity of sentences but even also similarity of whole paragraphs and documents. We have evaluated these methods using the data from the Yahoo Question and Answer of the Non-Factual Data Set.

1 Introduction

In this paper we extend our research from[1] where we presented and evaluated methods for finding similarity between textual data. These methods can be used inside modules of ITS to evaluate student performance [2]. ITS systems are computer-based instructional systems that are composed of four principal models: the expert model, the learner model, the tutor model and the interface. Expert model otherwise known as domain model is intended as a model that contains all concepts, rules and problem solving strategies. Expert model can be used in variety of contexts such as a source of expert knowledge, student's evaluation and student performance error detection. In this model components for sentence similarity are very useful. Student model is an overlay on expert model and is one of the main components of ITS system. In this model special attention is directed to student's affective and cognitive state and it's evaluation through the learning process [3]. Tutoring model receives the information from student and domain model. Tutoring model is an ITS component that makes choices about tutoring actions and strategies. One of the most important component of ITS system is communication module [4]. Main objective of this module is to fill in the gap between the computer who understands machine language instructions and user of an ITS system who understands natural language. Often in communication module question answering systems are implemented. The student receives the question about domain knowledge and answers the question in natural language. It is very important to find robust method of evaluating the students answer. Students answer to given question can be full sentence, phrase or even one word, so this task is not trivial. In general sentence similarity methods can be divided into two categories true understanding and text to text methods [5]. Existing methods for finding sentence similarity work with very high dimensional data and are not adaptable for use in ITS systems. This paper focuses directly on computing the similarity between very short texts. The focus is on finding true understanding text similarity for use in student answer evaluation. The structure of this article includes related work section followed by Proposed algorithm for sentence similarity and Evaluation of proposed methods section. In Related work section we list all similar works in this field. We also show how our proposed method differs from other

[*]Emil Brajković, Matice hrvatske b.b., +38763181014 & emil.brajkovic@fpmoz.sum.ba

similar methods. The section Proposed algorithm for sentence similarity includes couple subsections where we discuss the implementation of algorithm evaluated in this article. In the section Evaluation of proposed methods, we expand our previous evaluation with the new method explained in this article. In this section we briefly comment our evaluation methodology and show our results. In the conclusion we list our conclusions and future work related to this article.

2 Related work

ITS systems that are used in many different fields[3] could greatly benefit from robust sentence similarity method. Recent advancements in natural language processing has contributed many valuable solutions in finding sentence similarity. This section reviews some of the solutions addresses the difference between them and methods proposed in this paper. Overall methods for finding sentence similarity could be knowledge based and statistical based. Some of the most popular statistical based sentence similarity methods is Latent Semantic Analysis (LSA)[6]. LSA works by applying Singular Value Decomposition to co-occurrence matrix that is generated from textual data. Resulting dense representations of words on which cosine distance can be used to determine its relatedness. LSA method can distinguish between highly similar and dissimilar words but it is insensitive to minor similarities or dissimilarities between word pairs[7]. More recent knowledge based methods for sentence similarity measurement rely on semantic databases to find distances between concepts from two compared sentences[8]. These methods relying on the existence of concepts from sentence inside the semantic database, word sense disambiguation tool and the algorithm for finding similarity of two concepts. Another approach finding sentence similarity are the methods that are based on word embeddings[9]. These methods are based on finding dense word representations that are extracted from hidden layers of neural network. The cosine distance can be used to calculate word similarity (word that appear in similar contexts are closer). This distance is very good approximation of similarity because Word2Vec might find different sorts of most-similar words. This makes Word2Vec model very scalable and usable in real world applications. Word embedding methods are more preferred over approaches using lexical databases and show better results on word similarity evaluation tasks. The state-of-the-art systems implement hybrid approach using word embeddings and lexical databases[10]. In this work we use hybrid approach for finding sentence similarity using word embeddings and non-projective tree matching algorithm.

3 Proposed algorithm for sentence similarity

In this section we provide overview of our algorithm for sentence similarity that is used for evaluation of students answers in ITS system. There are several natural language processing steps that include syntax and semantic text preprocessing. Such preprocessed text is used as a structure for our algorithm to find the list of phrases that are semantically similar. We preprocess the sentence based on it's semantic constituents. Then we build semantic tree of phrases for the two sentences that are being compared. Global steps for this approach in sentence similarity finding are:

- Text preprocessing for two sentences for which we want to find the similarity
- Finding word similarity based on word embeddings
- Finding maximum common subgraph based on association graph

The details of the implementation will be explained in subsections.

3.1 Text preprocessing

Nodes are lists of words that can be connected to other words. Globally we can divide nodes into noun nodes, verb nodes, proposition nodes and adverbial node. Nodes with multiple words contain main word which is called head word. Visual representations of the nodes in sentences are shown in fig. 01.

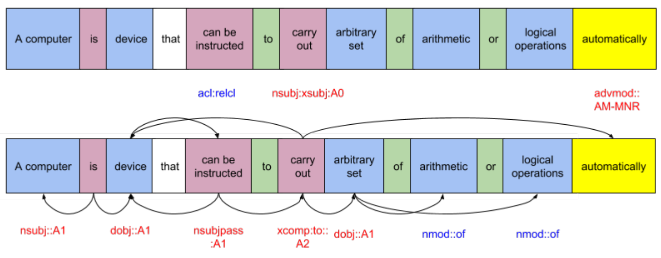

Figure 1: Parsed sentence using POS tagging algorithm for English language, node dependency relations and semantic roles

The words in sentence are segmented into nodes. These nodes are connected by syntactic and semantic relations that are parsed using CoreNLP[11] and SENNA[12] tools.

3.2 Finding similarity of words in sentences using word embeddings

To solve the word order problem the idea is to develop algorithms that will find similarity of two graphs. As shown in figures above the sentences are actually non projective trees, there are many algorithms that deal with tree similarity[13][14]. To find sentence similarity we need to compare the similarity of nodes in dependency tree of the two sentences. To find the word similarity, GloVe[15] word embeddings are used. The GloVe is actually a dictionary with word vector pairing such as Word2Vec[16]. This dictionary is developed from training a shallow neural network on large text corpora using co occurrence matrix generated from all words in corpora, the network is optimized on custom loss function and 100 dimensional weights that are at start randomly initialized after the optimization with SGD algorithm are paired with corresponding words. Such vector representations are called word embeddings and are very popular because the structure and the way the vectors are obtained give great results for finding semantically similar words. The similarity of words can be calculated with cosine distance of two word vectors. Every word is paired with 100 dimension real valued vector that is extracted from shallow layer of neural network. Nodes that are most similar are paired to so called node pairs. Nodes are paired with nodes with maximal similarity and similarity is calculated with algorithm 01.

```
Data: corresponding sentences $G_1$ and $G_2$
Result: list of edges $E$ and similarities $S$
while $v_1$ in $G_1$ do
    while $v_2$ in $G_2$ do
        if similarity($v_1$, $v_2$) > border then
            put $v_1$, $v_2$ into list of edges $E$;
            put $sim$ into list of similarities $S$;
        else
            continue the iteration;
        end
    end
end
```

Algorithm 1: Algorithm for pairing nodes inside two sentences

Similarity function is something that can be greatly improved, it's very dependent on the dictionary and corpora that is trained on, the newer methods for sentence similarity include finding similarities of word phrases using approaches such as skip-thoughts[17]. For every node word sense disambiguation is applied in sentence preprocessing step so the concept based similarities can also be used, but our experiments showed better results for word embeddings approach mainly because not all phrases are contained in WordNet database[18].

Glove embeddings showed very good results for finding phrase similarity. Phrase similarity is similarity of multiple words, or in this case nodes. For calculation of phrase similarity we used Word Mover's Distance[19] Finding similar node pairs is process where borders of similarity are set to exclude the nodes that have very small similarity or no similarity at all. Borders are defined based on the word type, our assumption is that if nodes are verbs or nouns the similarity should be larger, but if the node is apposition, proposition, pronoun or something else the border of similarity should be not that high. The borders are defined as follows: Verb nodes - 0.80, Noun nodes and all other node types (pronoun, aposition, proposition) - 0.60. The similar and paired nodepairs are shown in fig. 02.

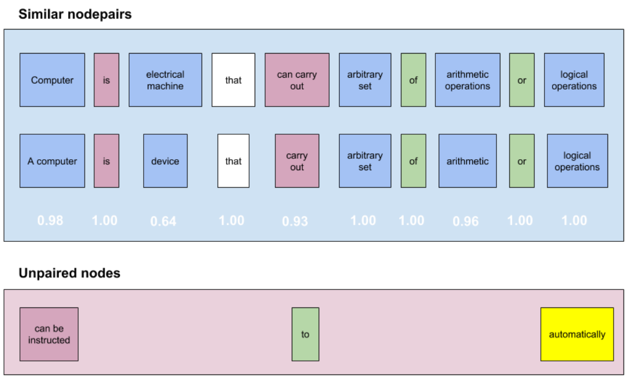

Figure 2: Paired node pairs from two sentences

3.3 Finding maximum common subgraph of whole sentences based on association graph

The next step is to find the maximum complete subgraph of two graphs, to do that we use nodes that are similar. From constructed node pairs we create so called association graph as defined in [20], all nodes that are similar are used to generate graph on which algorithm for finding maximum clique is applied.

Association graph AG The graph product $GP(V;E)$ of two graphs $G1(V1;E1)$ and $G2(V2;E2)$ is a new graph defined on the vertex set $V = V1 \otimes V2$ and the set of edges $E = E_1 \otimes E_2$. The association graph $AG(E,V)$ defined here is one of the graph products with the following adjacency conditions. Any $e = v_i v_s; v_j v_t$ considered to be adjacent:

- if $v_i \in V_{1is}$ adjacent to $v_j \in V_1$ in the original graph G_1 and $v_s \in V_2$ is adjacent to $v_t \in V_2$ in the original graph G_2 , or,
- if v_i is not adjacent to v_j and vs is not adjacent to v_t.

The association graph AG made by the previous definition should have all possibilities of vertex matches between two initial graphs $G1$ and $G2$; namely, a clique in AG corresponds to a common subgraph between $G1$ and $G2$; . Thus, the original problem of obtaining the largest match of sentences can be reduced to the computational task of searching for the largest clique in AG, $MCL(AG)$. For given two sentences the association graph is used for finding maximum clique.

A subgraph $subg_1$ of graph $G1$ is a new graph obtained from $G1$ by deleting some edges and vertices's, the same applies to $subg_2$.The nodes and vertices's that are deleted are those that do not belong to the maximum clique $MCL(AG)$.

If $(node_1, node_2) \notin AG$, where it is $node_1 \in G_1$ and $node_2 \in G_2$, then $node_1$ and $node_2$ are deleted from AG.

Subgraph $subg_1$ and subgraph $subg_2$ are maximal common subgraph MCS. For computing similarity we experimented with use of the Jaccard coefficient[23] which can be calculated shown in equation (1).

$$JC = (G1, G2) = M\ddot{C}S(G1 + G2) - MCS \quad (1)$$

This measure is the measure for the similarity of the sentences, so if we want to get the dissimilarity measure DM we use the JC and subtract it as follows $DM = 1 - JC$. We used this measure for evaluation and comparison with other algorithms.

```
Data: list of all edges $E$ and sentences $G_1$ and $G_2$
Result: list of connected edges $AG$
while $E_1$ in $E$ do
    while $E_2$ in $E$ do
        if $v_1$ of $E_1$ and $v_1$ of $E_2$ are adjacent in $G_1$ then
            if $v_2$ of $E_1$ and $v_2$ of $E_2$ are adjacent in $G_2$ then
                put $E_1$, $E_1$ into list;
            else
                continue the iteration;
            end
        else
            if $v_2$ of $E_1$ and $v_2$ of $E_2$ are not adjacent in $G_2$ then
                put $E_1$, $E_1$ into list;
            else
                continue the iteration;
            end
            continue the iteration;
        end
    end
end
```

Algorithm 2: Algorithm for finding association graph AG

4 Evaluation of proposed methods

Yahoo Non Factoid Question Dataset[24] was used in evaluation of good vs bad answers. The dataset contains 87,361 questions and appropriate answers, from which one of them was marked as the best answer for that question. Each question contains it's best answer along with other answers submitted by users. This dataset contains non-factoid QA, which means it covers topics beyond factoid question answering, such as "Who is the father of artificial intelligence?".

These answers are somewhat complex and finding function for similarity is a complex task. The objective was to find the method that maximizes the function of dissimilarity of two sentences. Reason for this is that answers that are labeled as best must be different than the other answers, labeled as bad answers. The evaluation methodology was conducted in three phases using three algorithms. We evaluated parse tree, knowledge tree and word vector representations. In the evaluation phase we define dissimilarity function as the function where we subtract the similarity score from maximum score. The parse tree and knowledge tree was traversed using Euler's algorithm and then Sorensen-Dice[21][22] algorithm was used to determine the similarity of trees. The results of our evaluation are shown in table I. We included the results of new proposed method described in this article. The resulting number shows the average dissimilarity measure that is generated by sum of all scores and divided by number of sentences.

Table 1: Results of two best algorithm evaluation on whole nf6l dataset

Algorithm	Number of Questions	Number of sentences	Result
Knowledge tree and Dice	5000	29723	74.078
Word2Vec sentence dissimilarity	5000	29723	68.740
New algorithm sentence dissimilarity	5000	29723	66.120
Knowledge tree and Dice	10000	61017	74.847
Word2Vec sentence dissimilarity	10000	61017	68.808
New algorithm sentence dissimilarity	10000	61017	70.234
Knowledge tree and Dice	87361	638847	76.302
Word2Vec sentence dissimilarity	87361	638847	67.778
New algorithm sentence dissimilarity	87361	638847	78.453

In our previous work[1] the method that was giving reasonable results was the knowledge tree approach. In this algorithm we used the constituency tree which was traversed using Euler's path algorithm on and nodes where compared using WordNet similarity measure. The new method in measuring sentence similarity has shown even better results for the whole corpus. Approach given in this work is insensitive to word order and semantically close terms such as synonyms, hypernyms and other. In table I it is shown that new algorithm proposed in this work gave the best results in finding dissimilarity measure between good and bad answers. In future work we plan to evaluate this method by standard means of evaluation using some of the standard tasks for sentence similarity, such as STS Benchmark[27].

References

[1] D. Vasic, E. Brajkovic, "Tree and word embedding based sentence similarity for evaluation of good answers in intelligent tutoring system", In proceedings SoftCom, 2017.

[2] T. Volarić, D. Vasić, E. Brajković Adaptive Tool for Teaching Programming Using Conceptual Maps. In: Hadžikadić M., Avdaković S. (eds) Advanced Technologies, Systems, and Applications. Lecture Notes in Networks and Systems, vol 3. Springer, 2017.

[3] S.R.D. Santos, J.L.M. Amaral, J.F.M. Amaral, ”Adaptive Intelligent Systems applied to two-wheeled robot and the effect of different terrains on performance”, Advances in Science, Technology and Engineering Systems Journal, vol. 2, no. 1, pp. 1-5 (2017)

[4] R. Nkambou, J Bourdeau, R. Mizoguchi, Advances in Intelligent Tutoring Systems, Springer-Link, 2010.

[5] V. Rus, M. Lintean, A. C. Graesser, D. S. McNamara, Text-to-text similarity of sentences, Applied Natural Language Processing: Identification, Investigation and Resolution, 2011.

[6] C. Papadimitriou, H. Tamaki, P. Raghavan, P. Raghavan, S. Vempala, ”Latent Semantic Indexing: A Probabilistic Analysis”, In proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1998, pp. 159–168.

[7] S. Simmons, Z. Estes, ”Using Latent Semantic Analysis to Estimate Similarity”, In proceedings The 28th Annual Conference of the Cognitive Science Society, 2006.

[8] A. Pawar, V. Mago, ”Calculating the similarity between words and sentences using a lexical database and corpus statistics”, IEEE transactions on knowledge and data engineering, 2018.

[9] M. Tomas, I. Sutskever, K. Chen, G. Corrado, J. Dean, ”Distributed Representations of Words and Phrases and their Compositionality”, 2013.

[10] R. Speer, J. Chin, C. Havasi, ”ConceptNet 5.5: An Open Multilingual Graph of General Knowledge”, In proceedings AAAI Conference on Artificial Intelligence, 2017,

[11] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, “The Stanford CoreNLP Natural Language Processing Toolkit,” in Association for Computational Linguistics (ACL) System Demonstrations, 2014, pp. 55–60.

[12] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa. ”Natural Language Processing (Almost) from Scratch”, Journal of Machine Learning Research (JMLR), 2011.

[13] P. Bille. Asurvey on tree edit distance and related problems. In Theory of Computer Science, 2004., pp. 217–239.

[14] R. Yang, P. Kalnis, A. K. H. Tung, ”Similarity Evaluation on Tree-structured Data”, In proceedings of 2005 in SIGMOD Conference, 2005.

[15] J. Pennington, R. Socher, C. D. Manning, ”GloVe: Global Vectors for Word Representation”, Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” Jan. 2013.

[17] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun S. Fidler, ”Skip-Thought Vectors”, CoRR, 2015.

[18] C. Fellbaum, ”WordNet: An Electronic Lexical Database”, Cambridge, MA: MIT Press, 1998.

[19] M. J. Kusner, Y. Sun, N. I. Kolkin, K. Q. Weinberger, ”From Word Embeddings To Document Distances”,In Proceedings of the 32nd International Conference on International Conference on Machine Learning, 2015.

[20] M. Hattori,Y. Okuno ,S. Goto, M. Kanehisa, ”Genome informatics. International”, In proceedings of Conference on Genome Informatics, 2003.

[21] L. R. Dice, “Measures of the Amount of Ecologic Association Between Species,” Ecology, vol. 26, no. 3, 1945.

[22] T. Sorensen, “A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons, Biol. Skr., vol. 5, 1948.

[23] P. Jaccard, ”Jaccard coefficient similarity”, New Phytologist vol. 11, 1912 pp. 37–50

[24] S. Harding, “nfL6: Yahoo Non-Factoid Question Dataset.” [Online]. Available: https://ciir.cs.umass.edu/downloads/nfL6/. [Accessed: 19-May-2017].

[25] D. Y. Chang, “Applications of the extent analysis method on fuzzy AHP,” Eur. J. Oper. Res., vol. 95, no. 3, pp. 649–655, 1996.

[26] C.-L. Hwang and K. Yoon, The Technique for Order of Preference by Similarity to Ideal Solution. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 1981.

[27] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia (2017) ”SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation Proceedings of the 10th International Workshop on Semantic Evaluation”, 2017.

[28] T. Landauer, P. Foltz, and D. Laham, “An Introduction to Latent Semantic Analysis,” Discourse Process., no. 25, 1998.