# Computing Text Similarity using Tree Edit Distance

Grigori Sidorov,
Helena Gómez-Adorno,
Ilia Markov
Center for Computing Research (CIC),
Instituto Politécnico Nacional (IPN),
Mexico City, Mexico
Email: sidorov@cic.ipn.mx,
helena.adorno@gmail.com,
markovilya@yahoo.com

David Pinto,
Nahun Loya
Faculty of Computer Science,
Autonomous University of Puebla (BUAP),
Puebla, Puebla, Mexico
Email: dpinto@cs.buap.mx,
nahun.loya@gmail.com

***Abstract*—In this paper, we propose the application of the Tree Edit Distance (TED) for calculation of similarity between syntactic n-grams for further detection of soft similarity between texts. The computation of text similarity is the basic task for many natural language processing problems, and it is an open research field. Syntactic n-grams are text features for Vector Space Model construction extracted from dependency trees. Soft similarity is application of Vector Space Model taking into account similarity of features. First, we discuss the advantages of the application of the TED to syntactic n-grams. Then, we present a procedure based on the TED and syntactic n-grams for calculating soft similarity between texts.**

## I. Introduction

Computation of similarity between texts is a basic operation for a variety of Natural Language Processing (NLP) applications and some other fields related to text processing, like Information Retrieval, for example. Text similarity plays a crucial role in many NLP tasks such as Recognizing Textual Entailment (RTE), answer selection for Question Answering (QA), Paraphrase Identification (PI), Plagiarism Detection, and many others.

The principal operation for calculation of similarity of texts is string comparison. There are extensive studies conducted on the comparison of string data using simple word overlap measures (bag of words, n-grams) [1], but they are not sufficient to solve these tasks accurately [2]. The main problem when using word overlap measures is the lack of understanding of the semantic relation between words and phrases.

Recently, we proposed a concept of text elements that are constructed in a non-linear way: syntactic n-grams, i.e., n-grams that are constructed by following paths in syntactic trees [3], [4]. There are various types of syntactic n-grams according to the types of elements they are composed of: lexical units (words, stems, lemmas), POS tags, SR tags (names of syntactic relations), characters, or the mixed ones.

Furthermore, we extended the concept of syntactic n-grams with the Integrated Syntactic Graph (ISG) [5]. The ISG integrates linguistic features from the different levels of language description in a single structure. The ISG can represent a sentence, a paragraph or a whole document. Its construction is based on the dependency parsing of each sentence of a given document and further integration of the syntactic trees.

In our previous studies we showed that syntactic n-grams allow taking into account syntactic information and can be used like traditional n-grams as features for the Vector Space Model (VSM) representation of texts [4]. Thus, we can use machine learning algorithms for the NLP tasks.

The other idea that we proposed recently is to calculate the similarity between texts using the Soft Cosine Similarity Measure [6], i.e., to take into account similarity of features in the Vector Space Model. Traditional Cosine Similarity Measure considers all features as independent ones. In the paper [6], we used traditional string similarity measure: Levenshtein distance (edit distance) for comparison of n-grams or syntactic n-grams.

In this paper, we propose to apply the Tree Edit Distance (TED) for computing the similarity between syntactic n-grams and Integrated Syntactic Graphs. Since they are non-linear tree structures, the TED should be the right measure for their comparison. Tree edit distance is a generalization of the edit distance for two strings, which measures the similarity between two strings. Tree edit distance was applied in several NLP applications such as information retrieval [7] and textual entailment [8], but only as an isolated heuristic.

We believe that TED is a more natural way to compute similarity of syntactic n-grams in comparison to what we have been doing so far in [6]. In comparison with our previous work that exploits various ad-hoc or heuristic ways of incorporating tree-edit operations, our proposal is a straightforward application of the TED as a general framework for computing text similarity based on soft comparison of syntactic n-grams or ISGs.

## II. Related Work

The TED has been broadly studied in algorithmic research. For example, Akutsu [9] reports that the TED is extensively used for comparison of tree structured data in bioinformatics, which is one of the fundamental tasks, for example, in analyzing glycans, RNA secondary structures, phylogenetic trees, etc.

In the last years, TED based methods have been widely applied in NLP related tasks. Kouylekov and Magnini [10] target the TED application to the dependency trees for recognizing textual entailment. The authors also study different methods of computing the cost functions for the edit distance algorithm.

Wang and Manning [11] apply probabilistic tree-edit models with structured latent variables for the tasks of Textual Entailment and Question Answering. The authors describe a tree-edit conditional random field model for predicting semantic relatedness of pairs of sentences. The authors also demonstrate that the generalized TED in a principled probabilistic model allows to embed alignments as structured latent variables.

The main contribution of the work by Alabbas and Ramsay [8] is extension of the TED in order to deal with subtree transformation operations as well as with single nodes. The authors claim that the extended TED with subtree operations is more effective and flexible than the standard TED.

Finally, Lin et al. [7] use the TED in measuring structure similarity for natural language processing based information retrieval.

Therefore, in the recent years, a significant number of new approaches and application of the TED is implemented in the NLP related tasks. However, to the best of our knowledge, no work has been done yet on applying the TED to the analysis of similarity of textual features. This makes it important to study its impact in the task of the calculation of text similarity.

## III. Tree Edit Distance

Tree Edit Distance between two trees is defined as the minimum cost of edit operations to transform one tree into another. Different edit operations can be used, but the most common edit operations are: (1) insert a node into a tree, (2) delete a node from a tree, (3) change the label of a node [12]. Each operation has a cost function and an edit script $S$ between two trees $T_1$ and $T_2$ is a sequence of edit operations turning $T_1$ into $T_2$. The cost of $S$ is the sum of the costs of the operations in $S$. The edit cost is denoted by $\delta(T_1, T_2)$.

Let us consider an example. Figure 1 shows the syntactic trees of two sentences: (a) *John likes to eat big red apples* and (b) *John likes big red apples*. In order to calculate the TED between the trees in Figures 1.a and 1.b, it is necessary to perform delete operations over the nodes *to* and *eat*.

Dynamic programming is often used to calculate the edit distance and there are many algorithms for calculating edit distance without any constraints on insert and delete operations [12]. The most popular dynamic programming algorithm to compute edit distance between two ordered trees is Zhang-Shasha's algorithm [13].

### A. Edit Operations and Edit Mappings

Let us consider two trees $T_1$ and $T_2$. Each edit operation is represented by $(l_1 \rightarrow l_2)$, where $(l_1, l_2)\ \epsilon(\Sigma_\lambda \times \Sigma_\lambda\ (\lambda, \lambda))$. The operation is an insert if $l_1 = \lambda$, is a deletion is $l_2 = \lambda$ and a relabeling if $l_1 \neq \lambda and l_2 \neq \lambda$. Given a metric, cost function $\gamma$ is defined on pairs of labels, the cost of an edit operation is defined by setting $\gamma(l_1 \rightarrow l_2 = \gamma(l_1, l_2))$. The cost of a sequence $S = s_1, ..., s_k$ of operations is given by $\gamma(S) = \sum_{i=1}^{k} \gamma(s_i)$. The edit distance, $\delta(T_1, T_2)$, between $T_1$ and $T_2$ is formally defined as: $\delta(T_1, T_2) = \min\{\delta(S) \mid S$ is a sequence of operations transforming $T_1$ into $T_2\}$.

Since $\gamma$ is a distance metric, $\delta$ becomes a distance metric too. An edit distance mapping (or just a mapping) between $T_1$ and $T_2$ is a representation of the edit operations, which is used in many of the TED algorithms. Formally, define the triple $(M, T_1, T_2)$ to be an ordered edit distance mapping from $T_1$ to $T_2$, if $M \subseteq V(T_1) \times V(T_2)$ and for any pair $(v_1, w_1)$, $(v_2, w_2)\epsilon M$:

1) $v_1 = v_2$ iff $w_1 = w_2$ (one-to-one condition),
2) $v_1$ is an ancestor of $v_2$ iff $w_1$ is an ancestor of $w_2$ (ancestor condition),
3) $v_1$ is to the left of $v_2$ iff $w_1$ is to the left of $w_2$ (sibling condition).

### B. A simple algorithm for the TED calculation

In this section, we describe a simple recursion-based algorithm developed by [14], which constitutes the basis for the dynamic programming algorithm of Zhang-Shasha [13].

Let $F$ be a forest (a set of trees) and $v$ is a node in $F$, where $F - v$ denotes the forest obtained by deleting $v$ from $F$. Furthermore, define $F - T(v)$ as the forest obtained by deleting $v$ and all descendants of $v$. The following lemma provides a way to compute edit distances for the general case of forests.

*Lemma 1:* Let $F_1$ and $F_2$ be ordered forests and a metric cost function defined on labels. Let $v$ and $w$ be the rightmost (if any) roots of the trees in $F_1$ and $F_2$, respectively. The equations are the following:

$$\delta(\theta, \theta) = 0$$
$$\delta(F_1, \theta) = \delta(F_1 - v, \theta) + \gamma(v \rightarrow \lambda)$$
$$\delta(\theta, F_2) = \delta(\theta, F_2 - v) + \gamma(\lambda \rightarrow w)$$
$$\delta(F_1, F_2) = \min \begin{cases} \delta(F_1 - v, F_2) + \gamma(v \rightarrow \lambda) \\ \delta(F_1, F_2 - w) + \gamma(\lambda \rightarrow w) \\ \delta(F_1(v), F_2(w)) + \\ \delta(F_1 - T_1(v), F_2 - T_2(w)) + \\ \gamma(v \rightarrow w) \end{cases}$$

Lemma 1 suggests a dynamic program. The value of $\delta(F_1, F_2)$ depends on the constant number of subproblems of a smaller size. Hence, we can compute $\delta(F_1, F_2)$ by computing $\delta(S_1, S_2)$ for all pairs of subproblems $S_1$ and $S_2$ in order of increasing size. Each new subproblem can be computed in constant time. Hence, the time complexity is bounded by the number of subproblems of $F_1$ times the number of subproblems of $F_2$.

## IV. Tree Edit Distance for Computing Text Similarity based on Non-linear Elements

In our previous work [6], we used the Levenshtein distance for computing the similarity between n-grams, but in that case, we used the plain representation of the n-grams, both traditional and syntactic n-grams. In this work, we propose the use of the TED for this calculation, given that it is a more natural and direct way to measure the distance between this kind of structures. The use of the TED for calculating the similarity between the syntactic representations of a text can prevent the loss of information while transforming the syntactic information (tree) into a plain representation.

For computing the similarity between the two trees let's consider Figure 1 that shows the dependency trees corresponding to the sentences $a$ and $b$. In the figure, the labels in the nodes correspond to the words of the sentences.

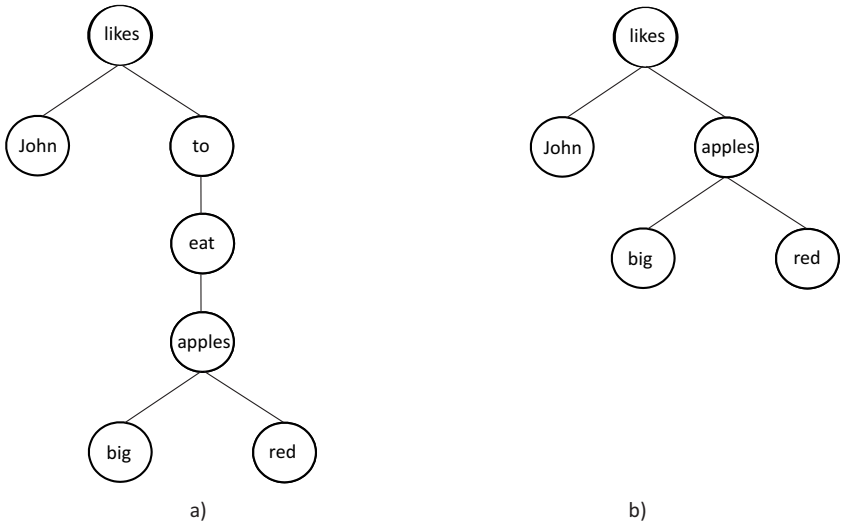


Fig. 1. Syntactic trees of the sentences (a) *John likes to eat big red apples.* and (b) *John likes big red apples.*

We can extract syntactic n-grams from each tree and there after compute the TED over them: our goal is to evaluate their similarity. So, for $T_1$ we have the syntactic 7-gram *likes[John,eat[to,apples[big,red]]]* and for $T_2$ a syntactic 5-gram *likes[John,apples[big,red]]*. Here we use metalanguage for representation of syntactic n-grams described, for example, in [3]. Using the Zhang-Shasha algorithm, the TED between $T_1$ and $T_2$ is 2. But if we calculate the string edit distance using the Levenshtein distance between $T_1$ and $T_2$, the result is 8, i.e., the TED gives much more intuitive result because these n-grams are very similar.

There are various ways to convert this distance into its inverse, similarity:

$$sim_1(T_i, T_j) = \frac{1}{1+d}, \tag{1}$$

$$sim_2(T_i, T_j) = \sqrt{\frac{1}{1+d}}, \tag{2}$$

where $d = TED(T_i, T_j)$ and $T_i, T_j$ are trees.

For the examples above the similarity between $T_1$ and $T_2$ using the TED is: $sim_1(T_1, T_2) = 0.33$ and $sim_2(T_1, T_2) = 0.58$. The corresponding similarity between $T_1$ and $T_2$ using Levenshtein is: $sim_1(T_1, T_2) = 0.11$ and $sim_2(T_1, T_2) = 0.33$.

Note that we can extract more than these two syntactic n-grams, but, in our opinion, it is enough for demonstration purposes. So, now we only need to apply Zhang-Shasha's algorithm to compute the TED between the two extracted n-grams. Finally, the total edit distance of the complete sentence will be the sum of the edit distances between all the syntactic n-grams obtained from both trees.

The same procedure can be applied to the ISG for comparing paragraphs or documents. When we use the ISG, we take advantage of its enriched structure that is composed not only of lexical-syntactic features, but also includes semantic relations. The semantic relations are particularly useful when two texts present the information using synonyms or hyperonyms, which do not have the direct lexical overlap.

## V. Conclusion and Future work

This paper presented a novel procedure for calculating text similarities using the tree edit distance (TED). We mention several works that successfully used the TED in various natural language processing tasks. We also include a brief overview of the concepts of the Tree Edit Distance, Edit Operations and Edit Mappings, and explain the basic recursion algorithm for calculating the TED.

We show on a concrete example that the computation of the similarity directly between trees (using the TED) gives a more accurate result, as compared to a string edit distance (Levenshtein distance).

In future work, we intend to implement the procedure proposed here for calculating soft text similarities based on non-linear elements. We will perform experiments for demonstrating the viability of using the TED in this task.

## References

[1] P. Pakray, S. Bandyopadhyay, and A. Gelbukh, "Textual entailment using lexical and syntactic similarity," 2011.

[2] V. Jijkoun and M. de Rijke, "Recognizing textual entailment: Is word similarity enough?" in *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognizing Textual Entailment*, ser. Lecture Notes in Computer Science, J. Quionero-Candela, I. Dagan, B. Magnini, and F. d'Alché–Buc, Eds. Springer, 2006, vol. 3944, pp. 449–460.

[3] G. Sidorov, "Syntactic dependency based n-grams in rule based automatic English as second language grammar correction," *International Journal of Computational Linguistics and Applications*, vol. 4, no. 2, pp. 169–188, 2013.

[4] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernández, "Syntactic n-grams as machine learning features for natural language processing," *Expert Systems with Applications*, vol. 41, no. 3, pp. 853–860, 2014.

[5] D. Pinto, H. Gómez-Adorno, D. V. Ayala, and V. K. Singh, "A graph-based multi-level linguistic representation for document understanding," *Pattern Recognition Letters*, vol. 41, pp. 93–102, 2014.

[6] G. Sidorov, A. F. Gelbukh, H. Gómez-Adorno, and D. Pinto, "Soft similarity and soft cosine measure: Similarity of features in vector space model," *Computación y Sistemas*, vol. 18, no. 3, pp. 491–504, 2014.

[7] Z. Lin, H. Wang, and S. McClean, "Measuring tree similarity for natural language processing based information retrieval," in *Proceedings of the Natural Language Processing and Information Systems, and 15th International Conference on Applications of Natural Language to Information Systems (NLDB)*. Springer-Verlag, 2010, pp. 13–23.

[8] M. Alabbas and A. Ramsay, "Dependency tree matching with extended tree edit distance with subtrees for textual entailment," in *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*, Sept 2012, pp. 11–18.

[9] T. Akutsu, "Tree edit distance problems: Algorithms and applications to bioinformatics," *IEICE Transactions*, vol. 93-D, no. 2, pp. 208–218, 2010.

[10] M. Kouylekov and B. Magnini, "Tree edit distance for recognizing textual entailment: Estimating the cost of insertion," in *Proc. of the PASCAL RTE-2 Challenge*, 2006, pp. 68–73.

[11] M. Wang and C. D. Manning, "Probabilistic tree-edit models with structured latent variables for textual entailment and question answering," in *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1164–1172.

[12] P. Bille, "A survey on tree edit distance and related problems," *Theor. Comput. Sci.*, vol. 337, no. 1-3, pp. 217–239, Jun. 2005.

[13] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM J. Comput.*, vol. 18, no. 6, pp. 1245–1262, Dec. 1989.

[14] P. N. Klein, "Computing the edit-distance between unrooted ordered trees," in *Proceedings of the 6th annual European Symposium on Algorithms (ESA)*. Springer-Verlag, 1998, pp. 91–102.