RTE4: Normalized Dependency Tree Alignment Using Unsupervised N-gram Word Similarity Score

Mehmet Ali Yatbaz
myatbaz@ku.edu.tr
Koç University

Abstract

We propose an unsupervised similarity metric to measure the relevance of word pairs using the Web1T data. The alignment scores between the dependency trees of the text and the hypothesis sentences are calculated based on this new similarity metric and these scores are used to predict the entailment between the text and the hypothesis sentences. The new similarity metric together with other features produces promising results on validation data sets however it yields an overall accuracy of %50.9 percent on the RTE4 test set.

1 System Overview

Our system makes entailment prediction for a text and hypothesis pair, $< t, h >$, in three stages:

1. *Preprocessing:* Sentence segmentation, sentence tokenization and part of speech tagging of words are done in this step.
2. *Dependency Tree & Normalization:* In this stage, dependency trees of sentences are constructed. After the dependency tree construction, several predefined normalization rules are applied on these dependency trees to get simpler (smaller) ones.
3. *Alignment & Prediction:* Maximum alignment scores between all dependency trees of t and h are calculated based on the new similarity metric. Using this score together with other features the system predicts the entailment of $< t.h >$.

The rest of the paper is organized in the following way. Section 1.1 describes the details of the first stage and the tools that are used in this stage. The tools of the second and the third stage are described in Section 1.2. Section 1.3, 1.4 and Section 1.5 provide the necessary information about Normalization Rules, Dependency Tree Alignment and Similarity Metric, respectively. Finally, the prediction results and brief comments are presented in Section 2 and Section 3.

1.1 Preprocessing

Sentence Segmentation & Tokenization: We use an in-house sentence splitter to split the multiple sentences into seperate ones. This tool also tokenizes the sentences.

Lemma assignments: We use an in-house tool together with the data that we used to train dependency parser to assign lemmas of each token. Therefore we prevent any future inconsistencies between the training data and the output of the tool.

POS tagger: We use SVMTool [6] with $-S\ LRL$ option to assign part of speech (POS) tags of each word.

1.2 Tools

Dependency Tree: We use a non-projective dependency parser based on spanning tree algorithms with the model that is described in [11]. The resulting labeled attachment score on the domain specific data set (i.e. WSJ) is 87.39% and the out of domain dataset (Brown) is 80.46% [12] . The performance of this model has two important consequences: (1) it affects the performance of sentence simplification rules and (2) it affects the accuracy of the alignment algorithm.

Named Entity Recognition (NER): We use dependency relations labeled with NAME or TITLE tags as a primitive method to extract NER.

Web1T Dataset: We use the Web 1T dataset [3] to extract the frequency counts of the word patterns that are described in Sec. 1.5 in detail. This data set contains domain independent counts of the word sequences up to length five in a 10^{12} word corpus derived from publicly accessible Web pages.

1.3 Normalization Rules

Normalization is a simplification process for the relatively complex sentences (ex: sentences with Wh-pronoun, passive voice or appositive structures). The idea of normalization was also used by several systems in RTE3 [5] challenge including the top system developed by [7] and by the system [10] in RTE2 [1]. The normalization rules are applied on the labeled or unlabelled dependency trees of the text and the hypothesis sentences. Similar to the system [10], simpler sentences are extracted from each original sentence by these rules and a set of new sentences is constructed for each original text and hypothesis sentence. There are six rules that we use for normalization:

1. Coordination Rule: This rule splits the sentences that include a coordination label or word. The rule is recursively applied therefore the new sentences do not have any coordination word.

2. Passive to active: This rule converts a passive sentence to an active sentence by removing the auxiliary verb and changing the real subject to the object of the sentence. Moreover, where possible this rule also changes the by-phrase to subject of the sentence.

3. Wh-pronoun: This rule recursively splits the sentences that have a wh-pronoun. (i.e. who, where, when, what etc.)

4. Appositive rule: This rule constructs new sentences by splitting the sub-tree that is connected to the main tree with the APPO tag. Moreover, the nouns or pronouns that have an appositive are replaced with the appositive to alternate the sentences with the same meaning.

5. Possessive Rule: This rule catches the possessive suffix and creates new sentences. (ex: Mark 's new car was stolen. → Mark has a new car. / Mark has a car. / Marks 's car was stolen. / Marks 's new car was stolen.)

6. Auxiliary, Punctuation & Model Rule: Auxiliary words, model words and punctuations are removed from the sentences.

7. Name Rule: For every original sentence we form a name set to keep the words that have a NAME or TITLE tag.

1.4 Dependecy Tree Alignment

We use the tree alignment algorithm that is developed by [9] based on the algorithm of [2]. The algorithm locates the alignment with the highest score in the space of all possible alignment of tree T and H where T and H are the labeled dependency trees of the hypothesis and test sentences. The maximum score of alignment between node $t \in T$ and $h \in H$ is defined by the $S(t,h)$ function in Eq.1.

$$S(t,h) = \max \begin{pmatrix} NODEMATCH(t,h) \\ \underset{t_i \in children\ of\ t}{\arg\max} \quad S(t_i,h) - \alpha \\ \underset{h_i \in children\ of\ h}{\arg\max} \quad S(t,h_i) - \beta \end{pmatrix} \quad (1)$$

This recursive equation considers two node matching possibilities:

1. Node h can be aligned to node t or vice versa;

2. Node t (h) can be aligned to children of h (t) with some penalty $0 \leq \beta \leq 1$ ($0 \leq \alpha \leq 1$).

The algorithm uses α and β to penalize skipping nodes from corresponding trees during the alignment. The hypothesis sentences are much shorter than the text sentences, therefore skipping a hypothesis' node will be more costly compared to skipping a test's node. Thus, the penalty of the text sentences is set to zero by default while the penalty of the hypothesis is estimated using the validation set.

The **TREEMATCH** function calculates the matching score between t and h and it is defined as:

$$\begin{aligned} TREEMATCH(t,h) &= w \cdot NODEMATCH(t,h) \\ &+ (1-w) \cdot CHILDMATCH(t,h) \end{aligned} \quad (2)$$

where w is the weight coefficient of the equation. The Eq.2 is a weighted sum of the matching score of the words of nodes, and the matching score of their children. The value of w is estimated using the validation set. The similarity between the words of t and h is calculated by the **NODEMATCH** function and it is defined as:

$$NODEMATCH(t,h) = \begin{cases} 1 & \text{if} \quad t_{word} = h_{word}, \\ 1 & \text{if} \quad t_{lemma} = h_{lemma}, \\ sim(t_{lemma}, h_{lemma}) & \text{if} \quad otherwise. \end{cases}$$

where $sim(t_{lemma}, h_{lemma})$ is the new similarity measurement between two words, that is described in Sec.1.5. There are four main classes of POS tags which are: noun, verb, adjective and other. If two words have POS tags from different classes, then their similarity score is equal to zero by default. The original version of the algorithm uses WordNet [4, 10] in **NODEMATCH** to match the words that are synonyms or hypernyms. We remove this part assuming that our new similarity metric described in Sec.1.5 assigns higher scores to the words that are synonyms or hypernyms. The similarity between the children of t and h is calculated by **CHILDMATCH** function which is formulated as:

$$CHILDMATCH(t,h) = \max_{p \in O(t,h)} \left(\sum_{(i,j) \in p} \frac{|S_{h_j}|}{|S_t|} \cdot S(t_i, h_j) \right) \quad (3)$$

where S_{h_j} is the set of all nodes rooted by t_i (similarly for S_{t_i}) and $O(t,h)$ is the set of all possible one-to-one pairings of the children of t with the children of h. This function looks for the best one-to-one matching of children nodes. Therefore maximizing this summation is nothing but finding the best sub-alignment rooted by t and h. If t or h does not have any children then this function returns 0.

1.5 Similarity Metric

The new similarity metric is defined under the assumption that the similarity between two words tends to be high if the former and the latter tokens of these words have similar distributions. Otherwise stated, words with similar frequencies for the former and the latter tokens have higher scores. In order to satisfy this assumption, we use the formulation of the DIRT algorithm [8] with some minor changes. DIRT extracts the paths and the frequency counts from the dependency trees constructed by Minipar[**?**] .On the other hand, our method extracts the frequency counts from the Web1T data without using any smoothing method. Therefore the frequency counts are more accurate than the commonly used 1GB of newspaper text (ex: Wall Street Journal). Our method assumes that every word has only one possible path and it is defined as:

$$\langle T_{-1} \; W \; T_{+1} \rangle \quad (4)$$

where T_{-1} and T_{+1} are the place holders of the former and latter tokens of the word W, respectively. The similarity score of the two words are calculated in terms of the mutual information between their paths and tokens. In other words, paths with common tokens will be more similar compared to the paths with less common tokens. We define two patterns

Verb	Score	Verb	Score
outbid	0.76	happen	0.10
cost	0.51	be	0.10
oblige	0.48	do	0.07
diversify	0.46	star	0.07
legalise	0.45	produce	0.07
purchase	0.43	seek	0.06
design	0.41	arm	0.06
include	0.40	attack	0.06
store	0.40	kill	0.03
toss	0.39	die	0.03

Table 1: Table on the left shows the top 10 similar verbs of buy and the one on the right shows the bottom 10 similar verbs of buy.

$< * \, , \, W >$ and $< W \, , \, * >$ to extract the frequency counts of the former and the latter tokens, where $*$ is a wildcard. The mutual information between a token t, and its place holder T for a given path p, is defined as:

$$mi(p, T_i, t) = log(\frac{C(p,T_i,t) \times C(*,T_i,*)}{C(p,T_i,*) \times C(*,T_i,t)}) \quad (5)$$

where $C(p,T_i,t)$ is the frequency count of token t at place T_i in path p and $*$ represents a summation over all possible values of the corresponding argument. The similarity between the place holder T_i of two different paths p_1 and p_2 is computed by the formula:

$$sim(p_1, p_2, T_i). = \frac{\sum_{t \in S(p_1,T_i) \cap S(p_2,T_i)} mi(p_1,T_i,t) + mi(p_2,T_i,t)}{\sum_{t \in S(p_1,T_i)} mi(p_1,T_i,t) + \sum_{t \in S(p_2,T_i)} mi(p_2,T_i,t)} \quad (6)$$

where $S(p_i,T_i)$ is the set of all possible tokens that are observed at T_i for a given path p_i. As previously mentioned, the similarity of two words is calculated using the paths of these words, thus using $Eq.6$. The similarity can be defined as:

$$sim(p_1, p_2) = \sqrt{sim(p_1, T_{-1}) * sim(p_2, T_{+1})} \quad (7)$$

Top 10 and bottom 10 similar verbs with the verb *buy,* based on the similarity score defined in this section, are presented in Table 1.

Type	Train	Validation	Test
IE	59.3	60.0	52.0
IR	65.8	57.0	46.5
QA	68.9	68.0	54.0
SUM	64.5	60.0	51.0
Average	64.6	61.0	50.9

Table 2: Percent recall scores of the system on the training, validation and RTE4 test datasets.

1.6 Feature construction and Entailment prediction

For every $< t, h >$ pair, the system defines three features:(1) domain of the sentence (2) the maximum score of alignment and (3) the name feature. The domain feature is the type of task (i.e. QA, IR, IE or SUM). The maximum score of alignment feature is defined as:

$$MaxScore = \arg\max_{t_i \in N_t} \left(\arg\max_{h_j \in N_h} S(t_i, h_j) \right) \quad (8)$$

where N_t and N_h are the set of normalized dependency trees of t and h, and $S(t_i, h_j)$ is the alignment function defined in Sec.1.4.

The name feature is a binary feature and is set to 1 if t and h have a common name. Otherwise it is set to 0. We have defined this feature to see whether we can use the labels of the dependency trees to extract named entities or not.

K-nearest neighbor algorithm is used to predict the entailment of each $< t, h >$ pair. The parameters, described in Sec.1.4, are optimized using the training and validation datasets.

2 Results

Results of the 2-way entailment challenge are obtained by using three data sets: (1) training dataset, (2) validation dataset and (3) test dataset. The development and the test datasets of RTE3 are used as the training and the validation data set of the system, respectively. The test dataset is the RTE4 test data. The trivial baseline of the RTE4 test dataset for the 2-way classification task is 50% by selecting the majority entailment answer. Table 2 presents the percent accuracies of the system on each task. The new similarity metric together with other features produces promising results on validation data sets. However the results on the RTE4 test dataset is significantly lower than the validation dataset scores. The performance of the system is not challenged on 3-way entailment and our experiments mainly focused on the effect of the system on 2-way entailment problem.

3 Future Work

The similarity scores can be improved by defining more than one path between pairs. The paths can be extracted naturally from the Web1T data based on their frequency counts. Multiple paths between pairs will also enable the system to use n-grams where n is larger than 3. Another improvement is to apply smoothing on Web1T data to assign probabilities to unseen tokens between pairs.

References

[1] R. Bar-Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor. The Second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 1–9, 2006.

[2] R. Barzilay. *Information fusion for multidocument summarization: paraphrasing and generation.* PhD thesis, Columbia University New York, NY, USA, 2003.

[3] Thorsten Brants and Alex Franz. Web 1T 5-gram version 1. Linguistic Data Consortium, Philadelphia, 2006. LDC2006T13.

[4] Christiane Fellbaum, editor. *Wordnet: An Electronic Lexical Database.* MIT Press, 1998.

[5] D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan. The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, 2007.

[6] Jesús Giménez and Lluís Màrquez. Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the 4th LREC*, 2004.

[7] A. Hickl and J. Bensley. A Discourse Commitment-Based Framework for Recognizing Textual Entailment. *ACL*, 2007.

[8] D. Lin and P. Pantel. DIRT-Discovery of Inference Rules from Text. In *Proceedings of ACM*

SIGKDD Conference on Knowledge Discovery and Data Mining, pages 323–328, 2001.

[9] E. Marsi and E. Krahmer. Explorations in sentence fusion. In *Proceedings of the 10th European Workshop on Natural Language Generation, Aberdeen, Scotland*, pages 109–117, 2005.

[10] E. Marsi, E. Krahmer, W. Bosma, and M. Theune. Normalized alignment of dependency trees for detecting textual entailment. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 56–61, 2006.

[11] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. *ACL*, 2005.

[12] Deniz Yuret, Mehmet Ali Yatbaz, and Ahmet Engin Ural. Discriminative vs. generative approaches in semantic role labeling. In *Conference on Computational Natural Language Learning (CoNLL)*, Manchaster, UK, Aug 2008.