# Recognizing Textual Entailment Using a Subsequence Kernel Method

**Rui Wang and Günter Neumann**

LT-lab, DFKI
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{wang.rui, neumann}@dfki.de

### Abstract

We present a novel approach to recognizing Textual Entailment. Structural features are constructed from abstract tree descriptions, which are automatically extracted from syntactic dependency trees. These features are then applied in a subsequence-kernel-based classifier to learn whether an entailment relation holds between two texts. Our method makes use of machine learning techniques using a limited data set, no external knowledge bases (e.g. WordNet), and no handcrafted inference rules. We achieve an accuracy of 74.5% for text pairs in the Information Extraction and Question Answering task, 63.6% for the RTE-2 test data, and 66.9% for the RET-3 test data.

## Introduction

Recognizing Textual Entailment (RTE) is a concrete task based on a relationship between two plain texts, Text (**T**) and Hypothesis (**H**). If the meaning of **H** can be inferred from the meaning of **T**, we say that **T** entails **H**. This task can be viewed as a binary classification task or as a probabilistic function mapping the pair **T**-**H** to a value between zero (not entailed) and one (fully entailed). Generally, the application of RTE falls into one of the following two categories: building a semantic model with the ability to perform inferences or improving the current NLP applications, cf. (Zanzotto and Moschitti, 2006).

From a linguistic perspective, several possible types of textual entailment exist: syntactic entailment, like "*I'm eating at home*" entails "*I'm eating*"; semantic entailment, like "*He loves her*" entails "*He likes her*"; implicature, like "*He beat Mary*" entails "*Mary is injured*".

From an NLP perspective, the goal of RTE is to find a common solution for several real-world NLP applications (Dagan and Glickman, 2004), which includes Information Extraction (IE), Information Retrieval (IR), Question Answering (QA), and SUMmarization (SUM). This is also the focus of the RTE challenges (Bar-Haim et al., 2006) organized by the PASCAL network in the last two years.

Table 1 displays several examples from the RTE-2 data set. The first pair (id=13) belongs to syntactic entailment. The most relevant knowledge here is "[LN1] *city of* [LN2]" entails "[LN2] *is located in* [LN1]", although **T** focuses on the earthquake event. The last pair (id=534) is a similar case with different structures in **T**. On the other hand, the third pair (id=133) requires not only an understanding of concepts like "*buy*" and "*takeover*", but also to understand the usage of "*said*", which is a case of semantic entailment.

These aspects motivate us to explore specialized entailment strategies for different NLP tasks. In other words, we want to find out the potential connections between entailment relations belonging to different linguistic layers for different applications.

In this paper, we propose a novel approach towards structure-oriented cases based on observations of the data: 1) **H** is usually textually shorter than **T**, 2) not all the information in **T** is relevant for the entailment relation, 3) the dissimilarity of relations among the same topics between **T** and **H** is of great importance. In brief, our primary method starts from **H** to **T** (i.e. in the opposite direction of the entailment relation), excluding irrelevant information from **T**. Then corresponding topics and relations on both sides are extracted. We represent the differences between **T** and **H** obtained before by means of a set of closed-class symbols, e.g., part-of-speech tags or grammatical function symbols. Finally, these acquired representations (named Entailment Patterns - EPs) are classified by means of subsequence kernels.

The structure-oriented RTE method is combined with two robust backup strategies, which are responsible for those cases that are not handled by the acquired EPs. One is a similarity matcher applied on top of local dependency relations of **T** and **H**; the other is a simple Bag-of-Words (BoW) approach that calculates the overlapping ratio of **H**. Thus, together with our main method, we make use of these three approaches to deal with different entailment cases in the implementation.

## Related Work

Conventional methods for the RTE task define measures for the similarity between **T** and **H** either by assuming an independence between words (Corley and Mihalcea, 2005; Glickman, Dagan, and Koppel, 2005) in a BoW fashion or by exploiting syntactic interpretations. (Kouylekov and Magnini, 2006) propose the use of syntactic tree editing distance to detect entailment relations. Since they calculate the similarity between the two dependency trees of **T** and **H** directly, the noisy information may decrease accuracy. This observation, in fact, motivated us to start from **H** towards the most relevant information in **T**.

| Id | Task | Text | Hypothesis | Entailment |
|---|---|---|---|---|
| 13 | IE | *Sunday's earthquake was felt in the **southern Indian city of Madras** on the mainland, as well as other parts of south India. The Naval meteorological office in Port Blair said it was the second biggest aftershock after the Dec. 26 earthquake.* | *The city of **Madras** is located in **Southern India**.* | YES |
| 61 | IE | *Although they were born on different planets, Oscar-winning actor **Nicolas Cage**'s new **son** and Superman have something in common, both were named **Kal-el**.* | ***Nicolas Cage**'s **son** is called **Kal-el**.* | YES |
| 133 | SUM | *Verizon Communications Inc. **said** on Monday it would **buy** long-distance telephone company MCI Communications Inc. in a deal worth $6.75 billion, giving Verizon a foothold in the market for serving large corporations.* | *Verizon Communications Inc.'s $6.7 billion **takeover** of long-distance provider MCI Inc. transformed the telephone industry.* | NO |
| 307 | IR | *Napkins, invitations and plain old paper cost more than they did a month ago.* | ***The cost of paper** is rising.* | YES |
| 534 | IE | *The main library at 101 E. Franklin St. changes its solo and group exhibitions monthly in the Gellman Room, the Second Floor Gallery, the **Dooley Foyer** and the **Dooley Hall**.* | ***Dooley Foyer** is located in **Dooley Hall**.* | NO |

Table 1 Examples from RTE-2

Logical rules (Bos and Markert, 2005) or sequences of allowable rewrite rules (de Salvo Braz et al., 2005) are another fashion for tackling the RTE task. Of the best two teams in RTE-2, one (Tatu et al., 2006) proposed a knowledge representation model and a logical proof system, achieving about 10% better performance than the third best team. The other (Hickl et al., 2006) acquired more training data from the internet automatically, also achieving about 10% better accuracy than the third best team. Consequently, obtaining more training data and embedding background knowledge were expected to be the focus of future research, as reported in the RTE-2 summary statement. However, except for the positive cases of SUM, **T**-**H** pairs are normally not very easy to collect automatically; multi-annotator agreement is difficult to reach on most of the cases as well. The knowledge-based approach also has its caveats, since the logical rules mentioned above are designed manually, and hence require a high amount of specialized human expertise in different NLP areas.

(Zanzotto and Moschitti, 2006) have utilized a tree kernel method for cross-pair similarity and showed an improvement, motivating us to investigate kernel-based methods. The main difference in our method is that we apply subsequence kernels on patterns constructed from the dependency trees of **T** and **H**, instead of applying tree kernels on complete parsing trees. This allows us, on the one hand, to identify the essential parts that indicate an entailment relationship, and on the other hand, to reduce computational complexity.

## Preprocessing and Backup Strategies

We are using *Minipar* (Lin, 1998) as our preprocessing parser. Our first backup strategy is based on a node-edge-node representation of the resulting dependency trees that expresses the local dependency relations found by *Minipar*, while the second one is a straightforward BoW method, which we will not present in this paper, but which you can find out more about in (Corley and Mihalcea, 2005).

A dependency structure consists of a set of triples. Each triple is a complex structure of the form *<node1, relation, node2>*, where *node1* represents the head, *node2* the modifier, and *relation* the dependency relation. For instance, the parsing result of **H** from pair (id=13) in Table 1 is as follows (only some parts are shown):

{<fin:C i shoot:V>, <shoot:V subj Goosen:N>, <shoot:V obj 69:N>, …}

The inner structure of the nodes consists of the lemma and the Part-Of-Speech (POS) tag. In the rest of the paper, **T** and **H** will represent either the original texts or the dependency structures.

### Triple Set Matcher

Chief requirements for the backup system are robustness and simplicity. Accordingly, we try to construct a similarity function which operates on two triple sets and determines how many triples of **H** are contained in **T**. The core assumption here is that *the higher the number of matching triple elements, the more similar both sets and the more likely it is that* **T** *entails* **H**. The similarity checker of two triples makes use of an approximate matching function:

$$TRIPLE-MATCH(<Tn_1, Tr, Tn_2>, <Hn_1, Hr, Hn_2>):$$
$$\quad if\ (Tn_1 = Hn_1\ \&\ Tr = Hr\ \&\ Tn_2 = Hn_2):$$
$$\quad\quad return\quad FullMatch$$
$$\quad elseif\ (Tn_1 = Hn_1\ \&\ Tr = Hr):$$
$$\quad\quad return\quad LeftMatch$$
$$\quad elseif\ (Tn_2 = Hn_2\ \&\ Tr = Hr):$$
$$\quad\quad return\quad RightMatch$$
$$\quad elseif\ (Tn_1 = Hn_1\ \&\ Tn_2 = Hn_2):$$
$$\quad\quad return\quad ArgsMatch$$

Note that in all cases a successful match between two nodes means that they have the same lemma and POS tag. The triple matcher is applied to the series of triple sets of **T** and **H**, ignoring sentence boundaries. The motivation for returning the different matching cases is to perform a partial match instead of an exact one. Different cases (i.e. ignoring either the parent node or the child node, or the relation between nodes) might provide different indications for the similarity of **T** and **H**. Consequently, the similarity function can be defined more precisely based on the sum of the matched triple elements of **H** divided by the cardinality of **H** needed for normalization,

$$Similarity(T,H) = \frac{1}{Card(H)} \times (a_1 \times NumOfFullMatch + a_2 \times NumOfLeftMatch + a_3 \times NumOfRightMatch + a_4 \times NumOfArgsMatch)$$

*Card(H)* is the number of triples in **H**; $a_1$ to $a_4$ are the different weights for the different matching cases.

Normalizing the sum of matching elements by the cardinality of **H** guarantees that $0 \leq Similarity(T, H) \leq 1$. A value of *0* means that **H** has nothing in common with **T**, a value of *1* means that **H** is completely covered by **T**, and a value in between means that **H** is partially covered by **T**.

The weights (i.e. $a_1$ to $a_4$) learned from the corpus in the later stage imply that the different "amount of missing linguistic information" influences the entailment relation differently.

## A Structure-Oriented Approach

Our approach is based on the hypothesis that *some particular differences between T and H will block or change the entailment relationship*. When judging the entailment relation, we initially assume that the relationship actually holds for each **T**-**H** pair (using the default value "YES"). The following steps are performed:

- Locate keywords (i.e. nouns and verbs) in **H**, and connect them by means of the dependency tree in order to extract the sub-tree. A sub-tree without the inner yield is defined as a *Tree Skeleton* (TS), the left-most and right-most nouns as *Foot Nodes* (FNs), the verb as *Root Node* (RN). The path from a foot node to the root is called a *Spine*. Usually, all the keywords are contained in two spines viz. a left spine and a right spine.
- Locate FNs, RN in **T** and form the TS of **T**.
- Generalize the two TSs using *Closed-Class Symbols* (CCSs) from a set of a limited size. The set consists of dependency relation tags, some POS tags, etc.
- Merge the spines obtained for **T** and **H**. The parts that do not match are called a *Spine Difference* (SD), usually, a left SD and a right SD.
- Use subsequence kernels to perform the binary classification on patterns containing the two spine differences.

### Tree Skeleton Extraction

The major motivation for firstly constructing the tree skeleton of **H** ($TS_H$) is that **H** indicates how to extract relevant parts of **T** for the entailment relation. Firstly, we construct a keyword set using all the nouns and verbs in **H**. Then we mark them in the dependency tree of **H** and extract the sub-tree by ignoring the inner yield. Normally, the root node of **H** ($RN_H$) is the main verb of **H**; all the keywords are contained in the two spines of $TS_H$.

Then we try to construct a tree skeleton from **T** (i.e. $TS_T$) using the information of $TS_H$. Before constructing $TS_T$, we need to extend the keyword set to hit the corresponding nodes in **T**. Thus, we apply a partial search using stemming and some word variation techniques on the substring level. For instance, the extended keyword set for the pair (id=61) in Table 1 is

{call:V, nicolas:N, nicolas_cage:N, son:N, kal-el:N, kal:N}

We identify these keywords in **T**. Since we have two foot nodes in **H**, two corresponding foot nodes will be marked in **T** accordingly. Starting from these two nodes, we traverse the dependency tree of **T** from bottom to top in order to identify the lowest common parent node, which we mark as $RN_T$. Note that such a node can be a verb, a noun or a dependency relation. If the two foot nodes of **T** belong to two sentences, a dummy node is created that connects the two spines.

To sum up, the major pre-requisite for this algorithm is that both the tree skeleton of **H** and the tree skeleton of **T** have two spines, each containing all the keywords of **H**. In practice, according to the experimental results we obtained from the RTE-2 data set, among all the 800 **T**-**H** pairs of the RTE-2 test set, we successfully extracted tree skeletons in 296 text pairs, i.e., 37% of the test data is covered by this step (see section Evaluation).

### Spine Generalization and Merging

Next, we collapse some of the dependency relation names from *Minipar* to more generalized tag names, like collapsing <*OBJ2*> and <*DESC*> to <*OBJ*>. We group together all the nodes that have relation labels like <*CONJ*> or <*PERSON*>, since they either refer to the same entity or belong to one class of entities sharing some common characteristics. Lemmas are removed except for the keywords. Finally, we add all the tags into CCS , the set of Closed-Class Symbols.

Since a tree skeleton actually consists of two connected spines (via the common root node), it can be transformed into a sequential structure. Figure 1 displays an example corresponding to the second pair (id=61) in Table 1. The general form of a sequential representation of a tree skeleton is:

LSP #RN# RSP

where RN is the root node label and LSP refers to the left spine and RSP to the right spine. Note that, LSP and RSP either represent the empty sequence (i.e. *NULL*) or a

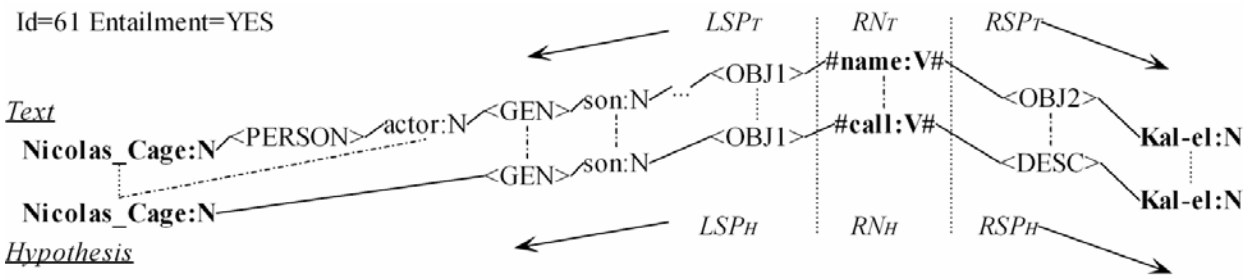


Figure 1: Examples of TSs

sequence of (generalized) CCS symbols. On the basis of this representation, the comparison between the two tree skeletons is straightforward. In fact, it can be conducted in two steps: 1) merge the two LSPs by excluding the longest common prefix, and 2) merge the two RSPs by excluding the longest common suffix. Then the Spine Difference (SD) is defined as the remaining infixes, which consist of two parts, $SD_T$ and $SD_H$. Each part can be either empty or a CCS sequence. For example, the two SDs of the example in Figure 1 (id=61) are (## is used as a special separator sign, LSD=Left Spine Difference, RDS=Right Spine Difference):

$LSD_T(...)$ ## $LSD_H(NULL)$

$RSD_T(NULL)$ ## $RSD_H(NULL)$

## Pattern Representation

We have observed that both the root node and the two neighboring dependency relations of it in a tree skeleton (i.e. <*SUBJ*> or <*OBJ*>) can play important roles in predicting the entailment relation as well. Therefore, we assign them two extra features referred to as Verb Consistency (VC) and Verb Relation Consistency (VRC). The former indicates whether two RNs have a similar meaning or not (e.g. "*buy*" and "*sell*" are not consistent), while the latter indicates whether the relations are contradicting (e.g. <*SUBJ*> and <*OBJ*> are contradicting). Considering verbs like "*to kill*", VRC will essentially convey the direction of the action, that is, from the subject to the object.

On top of LSD, RSD, VC, and VRC, we represent the differences between the tree skeletons $TS_T$ and $TS_H$ by means of an Entailment Pattern (EP), which is a quadruple <LSD, RSD, VC, VRC>, where LSD and RSD are either NULL or CCS sequences; VC is a Boolean value, where *true* means that the two root node labels are consistent and *false* otherwise; VRC has a ternary value, where *1* means that both relations are consistent, *-1* means at least one pair of corresponding relations is inconsistent, and *0* means $RN_T$ is not a verb[1]. The set of EPs actually defines the feature space for the subsequence kernels.

1 Note that $RN_H$ is guaranteed to be a verb, because otherwise the pair will be delegated to the backup systems.

## Subsequence Kernels

We have constructed two basic kernels for handling the LSD and the RSD part of an EP, and two trivial kernels for VC and VRC. They are combined linearly into a composite kernel, which performs the binary classification on them.

**Subsequence Kernel.** Since all the spine differences SDs are either empty or CCS sequences, we can utilize subsequence kernel methods to represent features implicitly, cf. (Bunescu and Mooney, 2006). Our subsequence kernel function is,

$$K_{subsequence}(<T,H>,<T',H'>) = \sum_{i=1}^{|T|}\sum_{i'=1}^{|T'|} K_{CCS}(CCS_i, CCS_{i'}) + \sum_{j=1}^{|H|}\sum_{j'=1}^{|H'|} K_{CCS}(CCS_j, CCS_{j'})$$

where *T*, *T'*, *H*, and *H'* refer to all spine differences SDs from **T** and **H**; |*T*|, |*T'*|, |*H*|, and |*H'*| represent cardinalities of SDs; The kernel function $K_{CCS}(CCS, CCS')$ checks whether the two argument CCSs are the same.

Since the RTE task checks the relationship between **T** and **H**, besides the SDs themselves, we need to consider collocations of some CCS subsequences between *T* and *H* as well.

**Subsequence-Collocation Kernel.** Essentially, this kernel evaluates the similarity of **T** and **H** by means of those CCS subsequences appearing on both sides. The kernel function is as follows:

$$K_{collocation}(<T,H>,<T',H'>) = \sum_{i=1}^{|T|}\sum_{i'=1}^{|T'|}\sum_{j=1}^{|H|}\sum_{j'=1}^{|H'|} K_{CCS}(CCS_i, CCS_{i'}) \cdot K_{CCS}(CCS_j, CCS_{j'})$$

On top of these two kernels, together with the two trivial ones (i.e. $K_{VC}$ and $K_{VRC}$), we use a composite kernel to combine them linearly with different weights:

$$K_{Composite} = \alpha K_{Subsequence} + \beta K_{Collocation} + \gamma K_{VC} + \delta K_{VRC}$$

In experiments, $\gamma$ and $\delta$ are learned from the training corpus; $\alpha$ and $\beta$ are set equal to each other, and currently both are *1*.

# Evaluation

We have compared three approaches, the two backup systems (BoW and Triple Set Matcher - TSM, as baselines) and the subsequence kernel method plus backup strategies

(SK+BS) in different experiments. The first group of experiments is based on the RTE-2 and RTE-3 data; and the second group of experiments exploits data collected from MUC6, BinRel (Roth and Yih, 2004), and TREC2003. For the kernel-based classification, we used the classifier SMO from the WEKA toolkit (Witten and Frank, 1999). All the numbers shown in the following tables are the percents of accuracies, except in Table 3, the first and third rows are the percents of the **T**-**H** pairs matched by the main method.

## Experiments on RTE Data

Both RTE-2 and RTE-3 data include the development set (800 **T**-**H** pairs, four tasks, IE, IR, QA, and SUM, and each task has 200 pairs) and the test set has the same size. Experiment A1 performs a 10-fold cross-validation (10-CV) on all the 1600 pairs of RTE-2 data; while Experiment A2 uses the development set for training and the test set for testing; Experiment B uses the RTE-3 development set for training and the test set for testing, cf. Table 2:

| Systems\Tasks | IE | IR | QA | SUM | ALL |
|---|---|---|---|---|---|
| Exp A1: 10-CV on RTE-2 Dev+Test Set | | | | | |
| BoW | **50**[2] | 58.8 | 58.8 | 74 | **60.4** |
| TSM | **50.8** | 57 | 62 | 70.8 | **60.2** |
| SK+BS | **61.2** | 58.8 | 63.8 | 74 | **64.5** |
| Exp A2: Train: RTE-2 Dev Set; Test: RTE-2 Test Set | | | | | |
| BoW | **50** | 56 | 60 | 66.5 | **58.1** |
| TSM | **50** | 53 | 64.5 | 65 | **58.1** |
| SK+BS | **62** | 61.5 | 64.5 | 66.5 | **63.6** |
| Exp B: Train: RTE-3 Dev Set; Test: RTE-3 Test Set | | | | | |
| SK+BS | 58.5 | 70.5 | 79.5 | 59 | **66.9** |

Table 2: Results on RTE Data

For the IE task our method SK+BS obtained the highest improvement over the baseline systems on RTE-2 data, which suggests that the kernel method seems to be more appropriate if the underlying task conveys a more "relational nature." The improvements for the other tasks are not so convincing as compared to the more "shallow" methods realized via BoW and TSM. Nevertheless, the overall result obtained in experiment A2 would have been among the top-4 of the RTE-2 challenge. Note that we do not exploit any additional knowledge source besides the dependency trees computed by *Minipar*.

Table 3 shows how SK+BS performs on top of the task-specific pairs matched by our EPs:

| Exps\Tasks | IE | IR | QA | SUM | ALL |
|---|---|---|---|---|---|
| ExpA1:matched | **63** | 18.3 | **36.3** | 16.3 | **33.5** |
| ExpA1:accuracy | 64 | 67.1 | 66.2 | 73.9 | **66.2** |
| ExpA2:matched | **64** | 23.5 | **44** | 17 | **37** |
| ExpA2:accuracy | 66.9 | 70.2 | 58.0 | 64.7 | **64.5** |

Table 3: Performances of CCS

2 The accuracy is actually 47.6%. Since random guess will achieve 50%, we take this for comparison.

For the IE and QA pairs, the method SK+BS obtained the highest coverage. However, for IR and SUM pairs, although it achieves good accuracies, the number of covered cases is low, and hence the backup systems will deal with most of the cases for IR and SUM. According to the experiments, IE and QA pairs mostly choose TSM as a backup strategy, while IR pairs choose BoW; and for SUM pairs, BoW has already achieved the best performance, cf. Table 2.

## Experiments on Additional Data

In order to get a deeper view of our method, we evaluated our systems using additional data. The results of the experiments achieved so far suggest that our method works well for the IE and QA tasks. Therefore, we decided to collect additional data from relevant sources (MUC, BinRel, and TREC2003) in order to test how our method performs for larger training sets.

For IE pairs, we use Named-Entities (NEs) and their corresponding relation to construct **H**, and the sentence(s) containing them to construct **T**. Negative examples will be either incorrect NEs or incorrect relations. For QA pairs, **H** consists of the question and the answer, and **T** is just the original sentence(s) containing the answer. Negative examples are those which do not have correct answers.

In all, there are new 750 **T**-**H** pairs from which our patterns can match 460 pairs. Table 4 displays the results:

| Systems | IE (MUC,BinRel) | QA (TREC2003) | Overall |
|---|---|---|---|
| BoW | 62.9 | 61.4 | 62.3 |
| TSM | 64.9 | 62.3 | 63.8 |
| SK | **76.3** | **65.7** | **74.5** |

Table 4: Results on Extra Data

The subsequence kernel (SK) method improves in both of the tasks, especially for the IE pairs. It seems that although our method can cover most of the QA pairs, the precision of the EPs still need improvements. We will discuss the gain and loss in detail in the following subsection.

## Discussions

It seems to be a promising direction to develop task specific entailment operators. Our structure-oriented method actually performs a classification on the entailment cases before doing the predictions. The results have shown differences in accuracy among pairs of different tasks, cf. Figure 2. Our subsequence kernel method works successfully on structure-oriented **T**-**H** pairs, most of which come from IE and QA tasks. If both $TS_T$ and $TS_H$ can be transformed into two CCS sequences, the comparison performs well, as is the case for the last example (id=534) in Table 1. Note that "*Dooley Foyer*" and "*Dooley Hall*" are a coordination, conveyed by the conjunction "*and*". The similar cases, "*work for*", a relation of a person and a company, or "*is located in*", a relation

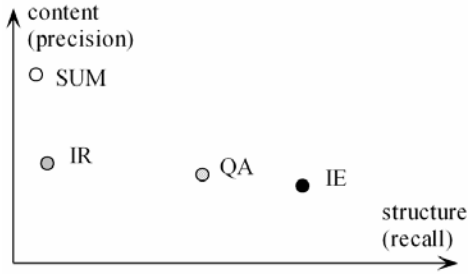


Figure 2: Distribution of task-specific pairs.

between two location names are normally indicated by the preposition "*of*". Based on these findings, taking into account the meaning of functional words more carefully might be helpful in improving RTE. Since the current RTE results have not been impressive when applying WordNet, lexical semantics of *Closed-Class Word* are worth considering.

Some incorrect cases like the third example (id=133) in Table 1 suggest that extending the tree skeleton might be useful. The trick for correctly predicting the entailment relation in this example is the word "*said*", which is not included in our current version, as we stop at the first common parent node in the dependency tree.

Of course, our method still has low coverage, even for IE pairs. Since we have applied some fuzzy matching techniques on the substring level, "*Southern Indian*" and "*Southern India*" in the first example (id=13) in Table 1 could be matched successfully. However, for person name abbreviations like "*J.D.E.*", it is not trivial to make a connection with "*J.D. Edwards*". A similar problem happens in temporal expressions like "*13th of January 1990*" with "*Jan. 13, '90*", as well. Obviously, we have to improve the matching strategy.

Besides that, some other missed cases (mostly in the IR and SUM tasks) contain more than two foot nodes. For instance, consider the third example (id=133) in Table 1. The keyword set contains at least two company names, and other nouns like "*telephone*" and "*industry*". Cases like this also account for low coverage.

## Conclusion and Future Work

Applying different RTE strategies for different NLP task is a reasonable solution. According to our observations, IE and QA pairs are more structure-oriented cases, while IR and SUM pairs are not. We have utilized a subsequence kernel method to deal with the former group, and applied backup strategies for the latter group. The result shows the advantages of our method, as well as areas of future work. In particular, we will extend the structure of tree skeletons for handling more complex cases, and will consider lexical semantic aspects of functional words in more depth.

## Acknowledgements

The work presented here was partially supported by a research grant from BMBF to the DFKI project HyLaP (FKZ: 01 IW F02) and the EC-funded project QALL-ME.

## References

Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B. and Szpektor, I. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In *Proc. of the PASCAL RTE-2 Challenge*, pp1-9.

Bos, J. and Markert, K. 2005. Combining Shallow and Deep NLP Methods for Recognizing Textual Entailment. In *Proc. of the PASCAL RTE Challenge*, pp65-68.

Bunescu, R. and Mooney, R. 2006. Subsequence Kernels for Relation Extraction. In *Proc. of the 19th Conference on Neural Information Processing Systems.*

Burger, J. and Ferro, L. 2005. Generating an Entailment Corpus from News Headlines. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pp49–54.

Corley, C. and Mihalcea, R. 2005. Measuring the Semantic Similarity of Texts. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pp13-18.

de Salvo Braz, R., Girju, R., Punyaka-nok, V., Roth, D., and Sammons, M. 2005. An Inference Model for Semantic Entailment in Natural Language. In *Proc. of the PASCAL RTE Challenge*, pp29-32.

Glickman, O., Dagan, I., and Koppel, M. 2005. Web based Probabilistic Textual Entailment. In *Proc. of the PASCAL RTE Challenge*, pp33-36.

Hickl, A., Williams, J., Bensley, J., Roberts, K., Rink, B. and Shi, Y. 2006. Recognizing Textual Entailment with LCC's GROUNDHOG System. In *Proc. of the PASCAL RTE-2 Challenge*, pp80-85.

Kouylekov, M. and Magnini, B. 2006. Tree Edit Distance for Recognizing Textual Entailment: Estimating the Cost of Insertion. In *Proc. of the PASCAL RTE-2 Challenge*, pp68-73.

Lin, D. 1998. Dependency-based Evaluation of MINIPAR. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC 1998*.

Roth, D. and Yih, W. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of the 8th Conference on Computational Natural Language Learning*, pp1-8.

Tatu, M., Iles, B., Slavik, J., Novischi, A. and Moldovan, D. 2006. COGEX at the Second Recognizing Textual Entailment Challenge. In *Proc. of the PASCAL RTE-2 Challenge*, pp104-109.

Witten, I. H. and Frank, E. 1999. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann.

Zanzotto, F.M. and Moschitti, A. 2006. Automatic Learning of Textual Entailments with Cross-pair Similarities. In *Proc. of ACL2006*, pp401-408.