A closer look at learning relations from text

Katrenko, S.

Citation for published version (APA):
Words are the coins making up the currency of sentences, and there are always too many small coins.

Jules Renard

ABSTRACT

In this chapter we address textual entailment by using a tree mining and matching technique. Our results show that accuracy can be improved when using a combination of lexical entailment with syntactic matching. Best results were obtained by combining two components is the following: 70% recall and 57.5% precision on the test set. Some parts of this chapter appeared previously in the publication “Using Maximal Embedded Syntactic Subtrees for Textual Entailment Recognition” co-authored with Pieter Adriaans in Proceedings of the RTE-2 workshop, 2006.

4.1 INTRODUCTION

Textual entailment is a complex task which can contribute to a variety of applications, such as question answering (QA) systems, information retrieval (IR) or information extraction (IE). Its main objective is to determine whether a meaning of one text (hypothesis) can be inferred from another one. An example of textual entailment is given below:

(4.1) (t) About two weeks before the trial started, I was in Shapiro’s office in Century City.

(h) Shapiro works in Century City.

To be able to infer the second sentence from the first one needs to know that offices are usually places where people work. Given that someone’s office is located in the Century City it leads to the conclusion that this person works there. Note that not all the information that can be found in (t) is actually needed to infer (h). The fact that there was a trial to be started is not relevant for (h). Example (4.1) shows that automatic recognition of textual entailment can benefit from NLP techniques (such as paraphrasing) and information found in various knowledge sources such as ontologies, taxonomies and others.

This chapter is organized as follows. In Section 4.2 we briefly discuss the notion of entailment the way it is defined in logic and in language and
review some requirements with respect to the latter. We then consider various methods that have been used in the research community to tackle textual entailment. Section 4.4 provides background information on tree mining. Sections 4.5-4.6 present our own contribution by applying tree mining techniques to the dependency structures. We conclude with the directions for the future work.

4.2 WHAT IS TEXTUAL ENTAILMENT?

In logic it is common to distinguish between inductive and deductive arguments. In valid deductive arguments the truth of the premises is sufficient to entail the conclusion while in inductive arguments it is only likely (to a certain degree) for a conclusion to be true. It is usually said that, for deductive arguments, the premises logically entail the conclusion. There exist two criteria for entailment. The first one is concerned with entailment as a proof of \( y \) from \( X \) and the second one is so-called semantic entailment. Semantic entailment is defined as follows: for any member of a set \( X (x \in X) \) and for any sentence \( y \), \( X \) semantically entails \( y \) \( (X \models y) \) iff \( y \) is true in all models in which all statements from \( X \) are true. This definition uses a notion of a model which is usually understood as a set of atomic sentences. Even though these two criteria can coincide, semantic entailment cannot always be proven (e.g., in incomplete logic systems).

Entailment has been studied not only by logicians or linguists, it has also gotten attention from psychologists. The central theme they consider is human thinking, and in particular how humans grasp simple and complex inferences. By distinguishing entailing and entailed statements, psychologists introduce a notion of support. Entailing statements are said to give a maximal support for the entailed statement if the support they provide is equal to the support of the entailing statements themselves. For instance, the support that the statement "A and B" provides to the statement ‘A’ is as strong as support provided by ‘A’ to itself. An intuitive example for weaker support could be an III syllogism (where I stands for the particular affirmatives) which is formulated as the follows:

<table>
<thead>
<tr>
<th>Major premise:</th>
<th>Some X are Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor premise:</td>
<td>Some Y are Z</td>
</tr>
<tr>
<td>Conclusion:</td>
<td>Some X are Z</td>
</tr>
</tbody>
</table>

Clearly, the conclusion is likely to be true but it is not guaranteed to be so. Psychological studies have shown that human subjects have difficulties with judging logically correct arguments [142]. The experiments reveal that humans tend to be misled by their beliefs if the argument is nondeductible. In other words, they attempt to define the degree of support rather than to give an answer based on the given premises only.
In linguistics, textual entailment is studied along with implicatures and presuppositions. Textual entailment does however differ from both. Presuppositions stand for anything that is taken for granted given an utterance and they remain under negation. In the well known example ‘the king of France is bold’ the existence of the king of France is a presupposition (which would change if we negate this utterance). This is clearly not the case with entailments. An implicature is anything that is inferred from an utterance without being a condition for the utterance to be true. A standard example of implicatures can be found in Example (4.2). Here, one infers that he can find petrol at the garage. Such inference is possible because (A) assumes that (B) is cooperative and provides therefore relevant information.

(4.2) (A) I’ve just run out of petrol.
(B) There is a garage just around the corner.

Textual entailment is an important component for language understanding in general and for more fine-grained tasks such as question answering in particular. Given the keen interest in handling textual entailment, the NLP community started organizing shared tasks, the first of which was held in 2005 and is being popular up to now. It was stated by its organizers that “the Recognizing Textual Entailment (RTE) Challenge is an attempt to promote an abstract generic task that captures major semantic inference needed across applications” [38]. For the Second PASCAL RTE Challenge in 2006, Bar-Heim et al. [7] refined a definition of textual entailment by taking into account the following requirements:

1. directionality We formalize a directionality criterion as follows: 
   \[ T \models H \Rightarrow H \models T. \]

2. full entailment If there is more information in \( H \) then it could be possibly entailed from \( T \), then \( H \) is not being considered as an inference from \( T \) (entailment is not full).

3. probability of the entailment to take place If it is very probable for entailment to occur, such \( T - H \) pairs are judged as positive examples.

4. presupposition of common knowledge Information in \( T \) might be not sufficient to infer \( H \) but as soon as one uses the background knowledge \( BK \), \( H \) is easily entailed (\( T \land BK \models H \)).

In our view, there is a link between entailing more information than is present in text and entailment with background knowledge. At first glance, it may seem that these two conditions are contradictory. Confusion may arise when the background knowledge is interpreted as a situational knowledge rather than generic information about the states.
of affairs. Consider two examples in (4.3) and (4.4). In the first case, (4.3) is inferred from (4.3) by using information about buying and selling which is located in the background knowledge base. The second example, however, is not an entailment because (4.4) contains more information than it can be inferred from (4.4) and this additional information (‘Ann went home’) is not part of the background knowledge.

(4.3) (t) Ann bought flowers from John.
     (h) John sold flowers to Ann.

(4.4) (t) Ann bought flowers from John.
     (h) Ann bought flowers from John and went home.

4.3 RELATED WORK

Textual entailment has many applications, varying from question answering to story comprehension. In question answering, one is looking for a text segment such that the answer to a given question is subsumed by it. There is no guarantee that the answer will perfectly match a question and for this reason textual entailment becomes a necessity. Story comprehension is often seen as a process that maintains and updates a collection of propositions about the state of affairs and textual entailment is an important component of it as well.

Generally speaking, current approaches to textual entailment fall in three categories; syntactically oriented [158], knowledge based and one that combines both perspectives [72, 39]. These will be further described in the next sections. Even though it has been recognized in the early stages that such a complex task as textual entailment requires not only structural transformations but also availability of large knowledge bases, it is still unclear how to combine information from different sources in the best possible way. This is confirmed by the results obtained by various research groups on the RTE-2 data set. Several teams built the systems that incorporated different information sources but not all of them were equally successful. A group from Stanford [39] used a number of components such as graph alignment, lexical relations, background knowledge and a list of linguistic features that take account for polarity, modality, factivity an some others. Nevertheless, the results attained by this team do not differ from the performance provided by simpler methods [115, 92].

Complex systems are difficult to compare because the final performance is influenced not only by the individual components but also by their combination. Nevertheless, the conclusion drawn at the end of the RTE-2 was that the most crucial factors on the particular data sets were adding more data and using large knowledge bases. The most successful approach [72] also used more training data which was
collected from the Web (e.g., two sentences with explicit discourse connective for contrast such as but are considered negative examples of TE). Syntactically-oriented approaches surprisingly lead to similar results, no matter how syntactic information is employed. In most cases, they outperform lexical overlap and present an important component that has to be further integrated with other methods to fully tackle textual entailment [158, 115].

Recent RTE challenge [52] introduced several changes to the set-up of the task. Instead of the binary classification of entailment, three categories are considered, such as entailment, contradiction and unknown (the truth of hypothesis cannot be determined based on the given text fragment). Apart from this, the provided data sets became slightly larger. Similarly to the previous years, the IE subtask appears to be the most challenging. Three-way classification also seems to be more difficult as shown by the overall performance which is lower compared to the results on binary classification in previous years.

4.3.1 Syntactically-oriented methods

In the early seventies, Schank [166] proposed to view the language understanding problem as a complex enterprise and to look for a generic theory that would account for it as a whole rather than focusing on some parts of the understanding process. A key idea in his theory is an interlingual conceptual base such that all language units are mapped onto it. Schank [166] also emphasizes a prediction that can be made using conceptual structures by arguing that

\[\text{…humans engaged in the understanding process make predictions about a great deal more than the syntactic structure of a sentence, and any adequate theory must predict much of what is received as input in order to know how to handle it.}\]

The conceptual dependencies that Schank defines can be seen as an analogue to the syntactic dependencies. But unlike syntactic dependencies, conceptual dependencies take place between three types of concepts, nominals (“things that can be thought of by themselves without the need for relating them to some other concept”), actions (what is a nominal doing?) and modifiers (an attribute of a nominal or an action). To establish these dependencies one needs the background knowledge and the ability to make inferences. All these dependencies form a conceptual dependency network. Even though Schank [166] focused on conceptual nature of analysis, he nevertheless recognized a role that syntax plays in it. Most syntactically oriented approaches to textual entailment that are described below also assume that syntax takes an important part in solving this difficult task but acknowledge that syntactic information would most likely be insufficient and should be combined with other approaches. It is
however interesting to investigate to which extent syntactic information helps and whether the output of such methods is complementary.

Syntactic matching on two text fragments can be done in many different ways. The methods that are the closest to the approach described later in this chapter were proposed by Rus [158], Marsi et al. [115] and Kouylekov and Magnini [92]. The first not only uses the idea of matching of two dependency trees but also employs the same parser, namely Minipar [108]. Primarily, the goal is to find whether two graphs are isomorphic by subsumption. The degree of subsumption is quantified by the score that is computed as a sum over individual node (i.e., word or lemma) and relation scores. The author compared this approach against lexico-syntactic method that uses constituency parsing along with handling negation and synonymy, and concluded that the latter is superior to the former. However, the better results in this case are explained by adding extra features rather than by the differences in constituency and dependency parsing. Indeed, when the components responsible for synonymy and negation are removed, the performance slightly degrades.

The work of Marsi et al. [115] was inspired by the tree alignment algorithm that has been used for machine translation. This method starts with measuring similarity on nodes by matching two nodes and taking into account the best matching pairs of their descendants. This recursive definition allows for flexible alignment and guarantees low computational complexity. They also use a so-called ‘skip penalty’ to make sure that two trees do not necessarily have to be aligned perfectly. The classification then boils down to labeling all tree matches with a value higher than a threshold as positive and rejecting the others. In the implementation, Marsi et al. [115] use the MaltParser system [138] but their results are quite similar to other methods based on syntactic matching.

Kouylekov and Magnini [92] presented yet another approach for recognizing textual entailment by comparing dependency trees. Two trees are matched given the tree edit distance which, similarly to the edit distance on strings, is defined via operations of insertion, deletion and substitution. Kouylekov and Magnini [92] emphasize the importance of the insertion cost and consider several possibilities to compute it such as fixed insertion cost, setting it to the inverse document frequency (IDF) score or to a number of the children nodes. The intuition behind the last setting lies in the fact that nodes having more children are more relevant to “the meaning expressed by a certain phrase”.

More on a general note, Vanderwende et al. [195] observed that syntactic information alone is sufficient to determine whether there is an entailment for 37% of examples in the RTE-1 data set. Moreover, it mostly helps to recognize false entailments (27% of cases). This led to introduction of several false entailment features (unaligned entity, negation mismatch, modal mismatch, antonym match, argument movement, su-
perative mismatch and conditional mismatch) that proved to be accurate both on training and test sets [180].

### 4.3.2 Logical inference

Only two systems presented at the Second RTE Challenge were based on logical inference. Bos and Markert [11] considered first-order theorem proving tool and finite model building. To take into account common sense reasoning, they employed WordNet hyponymy relations and pre-coded inference rules for general knowledge (such as possessiveness, family relations, spatial knowledge and others). All text and hypothesis fragments were first parsed by the combinatory categorial grammar (CCG) parser whose output was presented as Discourse Representation Theory (DRS) fragments. Later, DRS was translated to first-order logic to be used for the reasoning tools. The output of reasoning was further integrated with the shallow features such as lemmata, text length and some others. By discussing performance of a combination of the shallow and deep approaches, Bos and Markert [11] noticed that reasoning did not improve the results achieved by the shallow approach alone. This is mainly due to the fact that there was substantial agreement between two methods and deep approach did not complement the shallow one.

Similarly to Bos and Markert [11], Tatu et al. [185] relied on the logical inference by using a natural language prover COGEX. But in contrast to previous work, they used a larger set of axioms. In this approach, WordNet was employed to find lexical chains between a pair of synsets and to detect nominalizations of verbs. In addition, their system included what they called NLP axioms (that could be used for noun compounds), world knowledge axioms (generated from WordNet and designed manually) and semantic calculus. The latter was used to add some logical properties of semantic relations. For example, semantic calculus introduces transitivity and symmetricity of the kinship relation and provides some constraints on other relations. The final system of Tatu et al. [185] consists of three modules. One of them is based on lexical alignment, and the other two use logic forms derived either from the constituency trees or from the dependency trees. Each module is assigned a weight and a linear combination of the components is used to yield the final classification label.

Even though two approaches that we described above are similar in nature, the difference in results is quite substantial (around 15% in accuracy, where Tatu et al. [185]’s system is the best performing among these). An explanation for it may lie in the number of axioms that were used in the second method and in their diversity as well. The authors point themselves that logic forms that they use accurately capture such phenomena as negation and quantification but also other axioms...
(which are referred to as the ‘axioms on demand’) play a significant role contributing to the overall performance.

4.4 TREE MINING

The need to process and analyze large data sets in many domains has led to the extensive research in the data mining field. While some approaches handle simple data representations such as item-sets or sequences, others tackle the problem of mining complex data structures. Examples of the latter are graph and tree mining. Graph mining has to deal with a high complexity since some graph-based operations are known to be NP-hard or NP-complete \[33\]. Contrary to this, trees are acyclic graphs and therefore there exist efficient algorithms for tree mining. The choice of a tree mining algorithm is usually guided by the data structure. For instance, mining in the biological domain would likely require graph mining techniques, whereas for mining of syntactically analyzed sentences tree mining methods are considered to be sufficient.

Tree mining and matching methods have been used for many tasks in different domains, including user navigation \[207\], mining in molecular databases \[137\], text mining \[130\], study of topological patterns in RNA, and many others. Since the syntactic structure of a sentence can be represented as a dependency tree, it is of considerable interest to study whether tree matching can improve results of the textual entailment recognition. In particular, we hypothesize that if two sentences have similar structures, it is more likely for one of them to be entailed from the other.

4.4.1 Definitions

In this section we provide several definitions for tree types. By definition, a tree is an acyclic graph. Further, there exist different types of trees that are distinguished based on whether there is a root vertex, an ordering imposed on a set of siblings. The simplest would be an unordered tree without a root which is referred to as a free tree. The trees that are considered in linguistics are mostly labeled rooted trees (Def. 8).

**Definition 8 (Labeled rooted tree).** A labeled rooted tree \( T = (V, E, \Sigma, L) \) is a directed acyclic connected graph of which one vertex is distinguished. It consists of a set of vertices \( V \), a set of edges \( E \), an alphabet \( \Sigma \) for labels of both, vertices and edges, and a labeling function \( L : V \cup E \rightarrow \Sigma \) assigning labels to vertices and edges.

When comparing two trees, different types of subtrees can be extracted. These types are defined according to the restrictions on the tree nodes ordering. The most specific case is a subtree where for a given node all its descendants must appear in both trees (bottom-up tree). This restriction
can be relaxed by allowing to remove some children (induced subtree) and requesting the ancestor-descendant ordering to be preserved even if some other descendants are missing (embedded subtree). The existence of different types of subtrees makes the use of tree mining more attractive since it is possible to define the granularity of the tree match.

Definition 9 (Bottom-up tree). Given a rooted labeled tree $T$, a rooted labeled tree $T'$ is a bottom-up subtree of $T$, iff:

(a) for each $v' \in V'$, there exists a vertex $v \in V$ such, that all descendants of $v$ are descendants of $v'$
(b) for a set of edges $E'$ it holds that $E' \subseteq E$ where $E$ and $E'$ are the sets of edges in $T$ and $T'$, respectively
(c) the labeling of $E'$ and $V'$ is preserved in $T'$

Definition 10 (Induced tree). Given a rooted labeled tree $T$, a rooted labeled tree $T'$ is an induced subtree of $T$, iff:

(a) $V' \subseteq V$
(b) for a set of edges $E'$ it holds that $E' \subseteq E$ where $E$ and $E'$ are the sets of edges in $T$ and $T'$, respectively
(c) the labeling of $V'$ is preserved in $T'$

Definition 11 (Embedded tree). A rooted labeled tree $T' = (V', E', \Sigma', L')$ is an embedded subtree of a rooted labeled tree $T = (V, E, \Sigma, L)$, iff:

(a) $V' \subseteq V$
(b) $v', w' \in V'$ and $(v', w') \in E'$ iff there exist $v, w \in V$ and $v$ is an ancestor of $w$ in $T$
(c) the labeling of $V'$ is preserved in $T'$

Examples of the bottom-up, induced and embedded subtrees for a tree depicted on Fig. 3a are given in Fig. 3b, Fig. 3c and Fig. 3d.

By definition, a tree consisting of $K$ nodes is referred to as $K$-tree. The tree on Fig. 3b is an example of 4-tree. In order to take into account the frequency of a subtree in question, the notion of a minimum support has been used. Support is defined as percentage of trees in a tree database (or a tree forest) that contain at least one occurrence of this subtree [207].

There have been several well-accepted canonical formats for the tree representation proposed. The choice of the canonical representation depends on the type of trees. For the rooted ordered trees, the string encoding has been used which resembles depth-first search. Following this encoding, the tree on Fig. 3a is represented as follows: ABC-1D-1EF-1G-1I-1. ‘-1’ in this sequence stands for backtracking to the parent node (for instance, traversing the tree from C to B in order to reach D). In order to perform mining efficiently, it has been proposed to use the vertical representation. We refer an interested reader to [207], where this format has been extensively discussed and we give just a brief definition and an example here. Trees in the vertical format are represented by the scope-list of a tree which consists of the tree identifier, match labels and
the scope of the last node of the tree. Let \( n_1 \) be a node in a tree \( T \) (where \( l \) is a node’s number according to the pre-order traversal of \( T \)). Then, the scope of \( n_1 \) is defined as an interval \([l, r]\) where \( r \) is a number of the right-most leaf node \( n_r \) in a subtree rooted by \( n_1 \). All nodes in Fig. 3a are accompanied by their scopes. Here, for the root node A (\( n_0 \)) its scope is determined by the right-most leaf I and is equal \([0, 7]\). Similarly, for the node B the scope is defined via its number (1) and the number of the node D and is given by \([1, 3]\).

Depending on the type of subtrees to be found, there are several algorithms which can be used for tree mining and matching. The excellent overview of different mining techniques is given in Chi et al. \cite{28}. For the Textual Entailment challenge we decided to search for the maximal embedded subtrees between the hypothesis and the text fragment. In our view, embedded tree mining can be useful when applied to the dependency structures. It relaxes constraints by allowing some nodes (words) to be different. Such types of tree matching can also be referred to as fuzzy matching.

There have been several methods proposed for mining rooted ordered embedded trees. Zaki \cite{207} has compared two of them, one of which relies on the breadth-first iterative search (PatternMatcher) and the other employing depth-first search. It has been shown that TreeMiner signifi-
cantly outperforms PatternMatcher, especially provided the low support level.

4.4.2 The TreeMiner algorithm

TreeMiner is an algorithm for mining embedded rooted ordered subtrees originally proposed by Zaki [207]. As most other algorithms for tree or itemset mining, it makes use of the anti-monotone property which can be stated as follows: for every frequent K-tree it holds that all 1..K-1-trees are frequent. In such a way, a search space is limited and one needs to build potential candidates of being frequent K-subtrees based on the frequent K-1 trees.

Given a forest of trees D and the minimum support m, the method starts with finding the set of the most frequent nodes F1 and the set of the most frequent 2-trees F2. After computing the F2 set, the algorithm proceeds with the enumeration of the K-frequent subtrees. The enumeration of the frequent subtrees employs the idea of the prefix equivalence classes of the trees. Two K-subtrees are said to belong to the same equivalence class iff they share the same prefix up to the (k-1)th node. For instance, for the trees in Fig. 5a and Fig. 5d, their prefix subtree is defined by A while for the trees in Fig. 5c and Fig. 5b, the prefix subtree P is equal EF. Note that the support value allows for flexible mining by either restricting a set of detected subtrees only to such that occur in all trees in the forest D (support of 100%) or selecting only subtrees that can be found in at least m trees.

In order to find all frequent K-subtrees, the set of operations on the subtrees in the same equivalence class has to be performed. This step is repeated iteratively until no new frequent patterns can be obtained. For more detail on TreeMiner method, we refer to Zaki [207].

4.5 System description

Our system consists of two parts, lexical overlap and syntactic matching. Given two sentences $S_h$ (hypothesis) and $S_t$ (text), the lexical overlap is calculated as a ratio of the number of lemmata two sentences share to the length of the shortest sentence (usually, hypothesis). $|S_h|$ and $|S_t|$ stand for the length of the hypothesis and the text fragment, respectively.

$$\text{lex} = \frac{\text{overlap}(S_h, S_t)}{\min(|S_h|, |S_t|)}$$  \hspace{1cm} (4.5)

For the syntactic matching we have designed a module which includes parsing and matching steps. We have used Minipar whose output can be presented either by dependency trees or by constituency structures. The output is also accompanied by lemmata of all words in a sentence.
We have chosen to work on lemmata only. As a next step, the depth first search has been performed so all trees have been presented in a pre-order format.

Further, syntactic matching has been carried out on each pair of syntactic trees, $T_h$ and $T_t$. Note that contrary to the case of the large forest, the input data in our case consists of two trees. For this reason, a natural choice is the matching that outputs a maximal rooted ordered embedded subtree, $T_e$. We set the support level to 100% (requesting all nodes in the resulting subtree to be present in two trees to be matched) and searched for the maximal subtree only. Since most methods in tree mining and matching work on the labeled trees where only vertices are labeled, we incorporated the labels of edges into labels of nodes.

**Example 2** (Mining on dependency paths). *Let us consider the following example of textual entailment in (4.6). First, we need to parse both sentences and to represent them in the preorder form. As a result, we obtain the representations in (4.7). Having run TreeMiner on these paths with the maximum support, we obtain the subtree in (4.8).*

(4.6) (t) Tony Shalhoub won best actor in a comedy “Monk” and James Spader won best actor in a drama for “Boston Legal”.

(h) James Spader won best actor for “Boston Legal”.

(4.7) (4.7) $(T_h) \ast fin\_U\ win\_i\ Tony\ Shalhoub\_s\ Tony\_lex-mod\ -1\ -1\ Tony\ Shalhoub\_subj\ -1\ actor\_obj\ best\_mod\ -1\ in\_mod\ comedy\_pcomp-n\ a\_det\ -1\ for\_mod\ ”_punc\ -1\ monk\_pcomp-n\ -1\ -1\ -1\ -1\ -1\ ”_punc\ -1\ and\_punc\ -1\ fin\_conj\ win\_i\ James\ Spader\_s\ James\_lex-mod\ -1\ -1\ James\ Spader\_subj\ -1\ actor\_obj\ best\_mod\ -1\ in\_mod\ drama\_pcomp-n\ a\_det\ -1\ for\_mod\ ”_punc\ -1\ Boston\ Legal\_pcomp-n\ Boston\_lex-mod\ -1\ -1\ -1\ -1\ -1\ -1\ ”_punc\ -1$.

(4.7) $(T_t) \ast fin\_U\ win\_i\ James\ Spader\_s\ James\_lex-mod\ -1\ -1\ James\ Spader\_subj\ -1\ actor\_obj\ best\_mod\ -1\ for\_mod\ ”_punc\ -1\ Boston\ Legal\_pcomp-n\ Boston\_lex-mod\ -1\ -1\ -1\ -1\ -1\ -1\ ”_punc\ -1$.

(4.7) $(T_e) \ast fin\_U\ win\_i\ actor\_obj\ best\_mod\ -1\ for\_mod\ ”_punc\ -1\ Boston\_lex-mod\ -1\ -1\ -1\ -1\ ”_punc$

To calculate the similarity, the following measure was used:

$$\text{syn} = \frac{|T_e|}{\min(|T_h|, |T_t|)} \quad (4.9)$$

Similarly to the previous example, $|T_h|$, $|T_e|$ and $|T_t|$ stand for the size of a tree which equals to the number of nodes each tree contains. The
more similar syntactic structure of two sentences is, the higher this score of syntactic matching will be.

To determine thresholds that should guide classifying the test data, the rule based classifier PART [204] has been trained based on the scores received by lexical overlap and syntactic matching. This classifier is based on partial decision trees and adopts separate-and-conquer strategy. We trained the classifier using 10-cross fold validation. To classify unseen example pairs tree mining was run on the test data in the same way as it was done for the training set. The second step included calculating the lexical overlap and syntactic matching scores as in Eq. 4.5 and in Eq. 4.9. Finally, the threshold that was obtained on the training set was used to classify the example pairs into positive and negative instances.

4.6 EXPERIMENTAL SET-UP

In what follows we describe the RTE-2 data set, give some pointers on how the data was collected and examine a few examples of the text-hypothesis pairs. An experimental part presents the results and discusses merits and limitations of our method. We conclude by comparing our approach against some other methods that rely on syntactic matching.

4.6.1 Data

The RTE-2 data set consists of 800 pairs in the development set and, similarly, 800 pairs in the test set. The data was collected aiming at four possible applications such as question answering, summarization, information extraction, and information retrieval. The way in which data examples were collected depended on the particular task. Even within a single application there could be several approaches to collect the data. For the IE application, there were 4 methods employed, two of which were relying on the output of the IE systems, and in the other two the data pairs were created manually. The output of the IE systems on two data sets, the ACE-2004 collection and the MUC-4 TST3 data set, was used to generate hypotheses. The MUC-4 collection (together with a data set of news articles) was also used for the entailment pairs generated by hand Bar-Heim et al. [7].

For the IR subset, the hypotheses were collected from TREC and CLEF data sets in the form of propositional queries. Texts were correspondingly retrieved by using some well-known search engines such as Google and Yahoo. Similarly to the IR collection, the QA data set was built by using queries from the TREC-QA and QA@CLEF data sets and by retrieving the corresponding answers from the Web. The input queries were subsequently modified from questions to affirmative sentences and served as hypotheses (H). The actual answers retrieved from the Web were used as
text fragments (T). The SUM data set was created by considering news clusters and their corresponding summaries which were automatically generated by summarization systems. If needed, additional transformations have been made to comply with the requirements mentioned in Section 4.2. For instance, a hypothesis could be simplified so that it could be fully inferred from the text fragment.

Some positive and negative examples per application (task) are given in Table 2. It is easy to observe that there are numerous factors to be taken into account. Semantic relations of synonymy and antonymy are important when considering IR examples. Knowing that a deterrent is an antonym of a catalyst should give a negative answer while being able to relate to go up and to rise should facilitate positive answer on whether an entailment takes place. Moreover, one should be capable of performing numerical comparison as in sentences (18), (256) and (674). A match in (18) is a good indicator of a possible entailment while a mismatch does not guarantee its absence. This is shown in (674) where a phrase ‘with more than 220,000 refugees’ can be entailed from ‘with more than 223,000 refugees’. Note that converse is not true.

4.6.2 Experiments and discussion

Being aware of the complexity of textual entailment task, we focus only on the syntactic part of it and are interested in whether syntactic matching provides useful information.

In this section, we address the following two research questions:

RQ1 How does a combination of syntactic matching rooted in tree mining and lexical overlap influence the overall performance on the RTE-2 data set?

RQ2 Do the syntactically oriented systems which have participated in the RTE-2 challenge provide similar predictions? Is there any way of combining the outputs of these systems to boost performance?

To answer the first question, we consider two runs. The first (run1) combines lexical overlap and syntactic matching, whereas the second (run2) is a simple lexical overlap only. Table 3 presents the results on the RTE-2 test data set in terms of accuracy while Table 5 provides the results in terms of precision and recall. As Table 3 suggests, the overall accuracy is higher for the run1.

We have carried out the analysis of how well the methods work on each topic in particular (Table 5). In general, run1 provides higher precision, while run2 gives better results on recall. This supports the observations made by Vanderwende et al. [195] (Section 4.3.1) who argued for usefulness of syntactic information in detection of false entailments.

The results of challenge also suggest that the system performs worst on the IE topic. Our analysis shows that although recall on the IE topic is
### Table 2: Some examples from the RTE-2 data set

<table>
<thead>
<tr>
<th>ID</th>
<th>T</th>
<th>H</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>He met U.S. President, George W. Bush, in Washington and British Prime Minister, Tony Blair, in London.</td>
<td>Washington is part of London.</td>
<td>NO IE</td>
</tr>
<tr>
<td>18</td>
<td>Muslims make up some 3.2 million of Germany’s 82 million people, and Turks represent two thirds of the minority.</td>
<td>82 million people live in Germany.</td>
<td>YES</td>
</tr>
<tr>
<td>150</td>
<td>Capital punishment is a catalyst for more crime.</td>
<td>Capital punishment is a deterrent to crime.</td>
<td>NO IR</td>
</tr>
<tr>
<td>307</td>
<td>The cost for some 30 million sheets of paper used each year by Cal State Long Beach colleges and departments went up Wednesday for the first time in four years.</td>
<td>The cost of paper is rising.</td>
<td>YES</td>
</tr>
<tr>
<td>256</td>
<td>Brian Brohm, the Louisville quarterback, threw for 368 yards and five touchdowns as the Cardinals beat visiting Oregon State 63-27.</td>
<td>The quarterback threw for 413 yards and three touchdowns, and then ran to the end zone two more times.</td>
<td>NO SUM</td>
</tr>
<tr>
<td>674</td>
<td>With more than 223,000 refugees already in Texas, Perry said officials at relief centers, around the state, say they are running out of room.</td>
<td>With more than 220,000 refugees in Texas, Perry warned that his state was running out of room.</td>
<td>YES</td>
</tr>
<tr>
<td>95</td>
<td>The Chicago White Sox are a major league baseball team based in Chicago, Illinois.</td>
<td>The Bulls basketball team is based in Chicago, Illinois.</td>
<td>NO QA</td>
</tr>
<tr>
<td>38</td>
<td>John Lennon’s widow, Yoko Ono, approved this museum as the world’s first museum to honor John Lennon.</td>
<td>Yoko Ono is John Lennon’s widow.</td>
<td>YES</td>
</tr>
</tbody>
</table>
one of the highest for two runs, precision is low. For instance, for run2, only 24 positive instances out of 100 have been misclassified. Similarly to Bar-Heim et al. [6], we have analyzed the misclassified examples looking for the type of information which might improve classification (Table 4). We have noticed that in most cases a combination of several information sources will be needed. In particular, in the mentioned fragments different types of paraphrasing apart from pure structural ones are involved. We assume therefore that most misclassified fragments would benefit from both, paraphrasing and using additional knowledge sources.

<table>
<thead>
<tr>
<th>Task</th>
<th>Run1</th>
<th>Run2</th>
</tr>
</thead>
<tbody>
<tr>
<td>QA</td>
<td>60.50</td>
<td>58.00</td>
</tr>
<tr>
<td>UM</td>
<td>69.50</td>
<td>67.00</td>
</tr>
<tr>
<td>IR</td>
<td>62.00</td>
<td>56.50</td>
</tr>
<tr>
<td>IE</td>
<td>44.00</td>
<td>47.00</td>
</tr>
<tr>
<td>Total</td>
<td>59.00</td>
<td>57.13</td>
</tr>
<tr>
<td>Litkowski [110]</td>
<td>56.63</td>
<td></td>
</tr>
<tr>
<td>Rus [158]</td>
<td>58.37</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Accuracy on the RTE-2 test set (official results)

Moreover, some of the examples on the IE topic clearly reflect patterns often used in the information extraction task. One of such examples is the snippet 358 with the organization-location relation presented below.

(4.10) (h) The declaration was the first from PepsiCo to damp speculation, after two weeks of press reports and mounting concern from politicians, including French President Jacques Chirac, that Paris-based Danone would be acquired.

(t) Danone headquarters are located in Paris.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>OCCURRENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paraphrases</td>
<td>11</td>
</tr>
<tr>
<td>Semantic information and</td>
<td>18</td>
</tr>
<tr>
<td>Background knowledge</td>
<td></td>
</tr>
<tr>
<td>Anaphora resolution</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4: Misclassified examples in the IE category

Since the topic annotations were missing in the RTE-2 test set, we trained the classifier on the whole training corpus making no distinction between topics the text snippets belong to. After we have received the
annotated test data, we also carried out additional test on the information extraction topic. When trained on IE topic only, the accuracy for IE on the test data increases but the overall accuracy decreases.

<table>
<thead>
<tr>
<th>Task</th>
<th>Run1</th>
<th>Run2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>59.50</td>
<td>58.91</td>
</tr>
<tr>
<td>QA</td>
<td>57.89</td>
<td>77.00</td>
</tr>
<tr>
<td>SUM</td>
<td>76.71</td>
<td>56.00</td>
</tr>
<tr>
<td>IR</td>
<td>60.35</td>
<td>43.00</td>
</tr>
<tr>
<td>IE</td>
<td>45.59</td>
<td>62.00</td>
</tr>
</tbody>
</table>

Table 5: Precision and recall on the RTE-2 test set (for TRUE category)

As mentioned in Section 4.5, we incorporated the labels of edges into the node labels. Consequently, nodes such as Botswana_subj and Botswana_pcomp_n have been considered to be different and they were not matched by our method. One way to overcome this is to discard syntactic functions and to consider the labels of vertices only. We have conducted this experiment similarly to run1, combining the results of lexical overlap and syntactic matching (we refer to this experiment as to run3).

We also trained different classifiers but the best result has been received by using Naive Bayes approach. In comparison to run2, both, recall and precision are higher. Recall equals 70% and precision increases to 57.50%.

1 where subj and pcomp-n are syntactic functions
This run can be considered as a trade-off between two first runs, where recall was high but precision low (run2) and in an opposite way (run1). The overall accuracy in run3 is slightly higher than in run1. Further investigation shows that, in comparison to run1, run3 provides the same accuracy for QA and SUM topics, higher accuracy for IE topic (46.50%) and lower accuracy for the IR topic (60.00%). The precision and recall plots are given on Fig. 4 and Fig. 5 respectively.

Besides this, the results obtained on the test set correspond to the 10-fold cross validation results on the training data (precision: 60.3%, recall: 69.1%). Interestingly, no matter how the classifiers have been trained and how two components (lexical and syntactic) have been combined, the highest recall reaches 72% only. This suggests that lexical and syntactic components have limitations and should possibly be combined with a deeper semantic analysis.

As reported by Kouylekov and Magnini [91] who also have used dependency parsing, the results can be affected by accuracy of parsing. In some cases a sentence with a complex structure leads not to one rooted tree but more. We encountered two such cases in the training set and none of them in the test data. In the two cases on the training data set, the first tree was selected for matching and the second one was ignored.

The second research question addresses performances of the individual systems evaluated on the same data sets. It is of interest whether syntactically oriented approaches complement each other or plainly produce the same labelings on the test data. We conducted additional analyses by considering the output of several methods that were said to use some kind of syntactic matching. As most systems components included a syntactic module, we limited ourselves only to those approaches that
either employed syntactic matching alone or were relying on the combination of syntactic matching with lexical relations (Table 6). For the more complex systems the contribution of their components is very often unknown and for this reason other methods were left out. In total, we examined the output of 9 runs and calculated agreement [47] for each pair of runs. The kappa agreement values (where kappa varies from -1 to 1) are given in Table 7.

<table>
<thead>
<tr>
<th>abbrev</th>
<th>reference</th>
<th>parser</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>KA</td>
<td>Katrenko and Adriaans [79]</td>
<td>Minipar</td>
<td>59.0</td>
</tr>
<tr>
<td>VR1</td>
<td>Rus [158]</td>
<td>Minipar</td>
<td>59.0</td>
</tr>
<tr>
<td>VR2</td>
<td>Rus [158]</td>
<td>Minipar</td>
<td>58.4</td>
</tr>
<tr>
<td>KL</td>
<td>Litkowski [110]</td>
<td>unknown</td>
<td>56.6</td>
</tr>
<tr>
<td>MK</td>
<td>Kouylekov and Magnini [92]</td>
<td>Minipar</td>
<td>57.3</td>
</tr>
<tr>
<td>EM</td>
<td>Marsi et al. [115]</td>
<td>MaltParser</td>
<td>60.5</td>
</tr>
<tr>
<td>OF1</td>
<td>Óscar Ferrández et al. [141]</td>
<td>unknown</td>
<td>55.6</td>
</tr>
<tr>
<td>OF2</td>
<td>Óscar Ferrández et al. [141]</td>
<td>unknown</td>
<td>54.8</td>
</tr>
<tr>
<td>LV</td>
<td>Vanderwende et al. [196]</td>
<td>NLPwin</td>
<td>62.5</td>
</tr>
</tbody>
</table>

Table 6: Abbreviations for 9 runs

<table>
<thead>
<tr>
<th>KA</th>
<th>VR1</th>
<th>VR2</th>
<th>KL</th>
<th>MK</th>
<th>EM</th>
<th>OF1</th>
<th>OF2</th>
<th>LV</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>.32</td>
<td>.36</td>
<td>.20</td>
<td>.37</td>
<td>.28</td>
<td>.16</td>
<td>.13</td>
<td>.16</td>
</tr>
<tr>
<td>1.00</td>
<td>.21</td>
<td>.21</td>
<td>.38</td>
<td>.38</td>
<td>.22</td>
<td>.18</td>
<td>.15</td>
<td>.14</td>
</tr>
<tr>
<td>1.00</td>
<td>.11</td>
<td>.32</td>
<td>.25</td>
<td>.15</td>
<td>.15</td>
<td>.13</td>
<td>.11</td>
<td>.11</td>
</tr>
<tr>
<td>1.00</td>
<td>.29</td>
<td>.19</td>
<td>-.01</td>
<td>-.01</td>
<td>-.01</td>
<td>.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>.27</td>
<td>.07</td>
<td>.04</td>
<td>.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>.07</td>
<td>.08</td>
<td>.12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>.86</td>
<td>.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Pairwise agreement for 9 runs

It can be observed that substantial agreement (.86) is achieved only between two runs by Óscar Ferrández et al. [141]. In this approach, two submitted runs were using information from WordNet (either relations or the Lin relatedness measure). Even though performance varies, the actual predictions overlap. As to the rest, the results do not seem to overlap very much. Our method attains fair agreement (≥.20) with approaches proposed by Rus [158], Kouylekov and Magnini [92] and Marsi et al. [115]. The agreement with the first two methods can be explained by the
fact that all three approaches use Minipar to analyze the data. However, they all use syntactic structure in a different way. While Rus [158] seeks isomorphic structures, Kouylekov and Magnini [92] focus on the various operations that can be performed on trees. In our approach we go further by searching for the most generic common substructure that two trees share (a maximal embedded tree). In contrast to the approaches mentioned above, Vanderwende et al. [196]'s method was fully based on heuristics, starting from the rules to align nodes in two dependency structures to the syntactic heuristics to rule out false entailment. Overall, there is an agreement among syntactic matching methods (especially those using Minipar) but it is clear that this agreement is far from substantial which leads us to the conclusion that these approaches can complement each other.

To check whether any improvement can be gained by combining results obtained by different groups, we carried out majority voting. During this procedure, every test example was assigned a label which most systems selected. In the first experiment we considered three approaches that used Minipar (Rus [158]'s, Kouylekov and Magnini [92] and ours) and later we added all other methods listed in Table 6. The voting results show only slight improvement in the first case (accuracy of 59.63%, precision of 59.65% and recall of 59.5%) while combining all methods leads to 63% of accuracy, 60.08% of precision and 77.5% of recall. We used the “stratified shuffling” method to test statistical significance [143]. Here, the null hypothesis states that two methods produce the same results or, in other words, scores they provide for a single instance (a text-hypothesis pair, in our case) are equally likely. The main idea behind this method is to randomly shuffle the individual scores between two methods and recompute the evaluation metric. Further, the difference in a metric after shuffling is compared to the original observed difference. Our tests show that the difference between voting results on 3 methods and performance of any of these individual methods is not statistically significant. Interestingly, the difference in results of voting on 9 runs and all individual runs but one is statistically significant (for our run at $p \leq .025$, for LV run at $p \leq .03$). The only run where no statistically significant difference was observed is the one by Marsi et al. [115]. Tables 8a-8b show how contingency tables change from voting on 3 methods’ outputs to 9. We note that a number of false positives and false negatives in Table 8a is almost the same while voting on 9 outputs leads to the considerable reduction in false negatives but increases a number of false positives. This finding reveals that a combination of syntactically oriented approaches results in high recall but precision is not very much affected. We would initially expect syntactic information to be helpful by boosting precision and it does so when single approaches are considered (e.g., in Table 5, precision of lexical overlap vs. precision of run1). Nevertheless, combining several methods by voting contributes to recall rather than
precision which suggests that more knowledge has to be used to filter out false positives.

<table>
<thead>
<tr>
<th></th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>238</td>
<td>162</td>
</tr>
<tr>
<td>-</td>
<td>161</td>
<td>239</td>
</tr>
</tbody>
</table>

(a) Contingency table for voting on 3 runs

<table>
<thead>
<tr>
<th></th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>310</td>
<td>90</td>
</tr>
<tr>
<td>-</td>
<td>206</td>
<td>194</td>
</tr>
</tbody>
</table>

(b) Contingency table for voting on 9 runs

Table 8: Contingency tables for voting schemes

4.7 CONCLUSIONS

Tree mining has not been widely used for the natural language processing applications yet. Some steps in this direction have been taken by Morinaga [130] who proposed applying the tree mining algorithms to the dependency trees. Later on, the frequent subtrees are intended to be used for the natural language generation. In their experiments, Morinaga [130] did not consider syntactic functions focusing on the node labels, i.e. words and respective parts of speech. In our view, the syntactic functions are important since they carry information about the way words are related to each other. As mentioned above, all applications and methods aiming at the mining embedded trees have dealt with trees whose nodes are labeled but there is no information about the labels of the edges given. One possible extension would be incorporating labels of edges in the nodes.

The chapter addresses tree mining where each node has an atomic label (combination of the node label together with the syntactic function would be considered to be atomic too, so gene-obj and gene-subj are different node labels). However, if one wishes to use more information, be it linguistic (e.g., parts of speech) or semantic (e.g., concept labels), it would become crucial to perform mining on trees with the complex node labels.

It is worth noting that tree mining is sensitive to the output of the employed syntactic parsers. Analyzers other than Minipar would most likely produce different output which in turn will influence the results of tree mining. Parsers as Stanford analyzers do incorporate information about prepositions into the syntactic functions (e.g., prep-in or prep-for) whereas Minipar does not do it. The parses for the same sentence provided by these two tools structurally may resemble each other but the differences in labelling would lead to other results. We would expect syntactically oriented methods that use the same parser to label textual entailment pairs similarly. This was proved by calculating pairwise agreement between approaches relying on the Minipar’s output. Even though
the agreement is not very high, it is still higher than the one on methods using different parsers. We also discovered that even with low pairwise agreement, a simple voting only slightly improves performance which leads us to the conclusion that either the outputs should be combined in a more elaborate manner by meta-learning or there are examples in the data that cannot be accurately classified by any syntactically inspired method.

Our analysis shows that, when using syntactic matching, precision and accuracy increase but recall drops. According to our assumptions, one reason for this is the use of our method producing ordered embedded trees. Pairs of sentences such as the one below do not receive high similarity scores and may be misclassified.

(4.11) (t) The currency used in China is the Renminbi Yuan.
(h) The Renminbi Yuan is the currency used in China.

The choice of ordered embedded trees was motivated by the fact that we worked with the English data sets. For the free word order languages it may be necessary to relax the requirements concerning order and use other mining solutions such as mining of unordered embedded trees.

Apart from this, we have used the most general subtrees. Although in most cases it allows to filter out possible false positives and to increase precision, there are also cases (e.g., in Example (4.12)) when that sentences have very complex structures but the text fragment and the hypothesis are correctly matched.

(4.12) (t) Four US cable companies, including industry leaders Comcast Corp and Time Warner Cable, have entered the fast-growing wireless arena through a joint venture with Sprint Nextel.
(h) Four US cable companies formed a joint venture with Sprint Nextel.

There are several ways to extend this approach in the future. First, it is possible to modify the syntactic matching component. Although the method we proposed performs a relaxed matching of trees, it is in some cases too restrictive since it does not make use of the additional semantic information (e.g., synonyms). In addition, syntactic matching can be incorporated in the larger system as one of the components. While examining training and test examples we also noticed that many of them include paraphrases. It has already been shown by Bar-Heim et al. [6] that adding paraphrases contributes to recall. Yet another module can make use of the existing resources, such as WordNet.

Textual entailment is a very complex task and it would be useful to find out which linguistic phenomena play the most important role for its successful detection. This need was recently recognized by a
number of researchers who analyzed existing data sets with respect to the phenomena which can be found there. More specifically, Vanderwende et al. [196] focused on such linguistic cues as synonymy, antonymy, superlative mismatch (i.e., two phrases “the world’s largest media and Internet company” and “the world’s largest company” should not match), conditional mismatch (a sentence “X is Y” cannot be inferred from “if X is doing Z, then Y”), modal mismatch (can does not imply must) and some others. Their findings were that such heuristics are useful but taken alone they do not guarantee high accuracy. The cues that performed relatively well were modal mismatch and prepositional antonyms (i.e., before/after). Another work done on textual entailment corpora was detection of contradictions [30]. It is argued here that contradictions require deeper inferences compared to some entailments but solutions to many types of contradictions (antonymy, negation, factive) are needed when working with textual entailment in general.