A Confidence Model for Syntactically-Motivated Entailment Proofs

Asher Stern
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, Israel
astern7@gmail.com

Ido Dagan
Dept. of Computer Science
Bar-Ilan University
Ramat Gan, Israel
dagan@cs.biu.ac.il

Abstract

This paper presents a novel method for recognizing textual entailment which derives the hypothesis from the text through a sequence of parse tree transformations. Unlike related approaches based on tree-edit-distance, we employ transformations which better capture linguistic structures of entailment. This is achieved by (a) extending an earlier deterministic knowledge-based algorithm with syntactically-motivated on-the-fly transformations, and (b) by introducing an algorithm that uniformly learns costs for all types of transformations. Our evaluations and analysis support the validity of this approach.

1 Introduction

Recognizing Textual Entailment (RTE) is the task of determining whether a given textual statement (a hypothesis), **H**, can be inferred by a given text passage, **T** (Dagan et al., 2005). In recent years, the task has attracted considerable interest, with research evolving around the six RTE challenges, organized by PASCAL[1] and later under the NIST Text Analysis Conference (TAC)[2]. While some of the proposed RTE systems employed quite shallow and ad-hoc techniques, a few principled approaches for modeling entailment inference began emerging as well.

This paper focuses on an appealing approach attempted in several previous works, which, like most RTE systems, utilizes parse-based representations of the text and hypothesis. Within this approach the parse-tree of **H** is explicitly generated from that of **T** by applying a sequence of tree transformation operations. In analogy to logic, that sequence can be referred to as a *proof*, by which the target proposition, represented as a parse tree, is generated from the given text propositions using appropriate proof steps.

In one line of these works (Wang and Manning, 2010; Heilman and Smith, 2010; Mehdad and Magnini, 2009) the tree-transformation operations followed mostly traditional *tree edit distance* operations, such as node insertion, deletion and substitution, and learned their costs according to the given RTE training data. As described in more detail in Section 2, these transformations do not necessarily capture the syntactic structure of entailment-preserving transformations.

On the other hand, a rich inventory of knowledge-based operations was employed by Bar-Haim et al. (2007a). Their operations enable transforming complete sub-trees which do capture the syntactic structure of entailment inferences. Nevertheless, their work did not include a learning component for estimating proof costs and their tree-transformations were based only on available knowledge resources, without providing on-the-fly operations that could compensate for some inevitably missing knowledge.

In this work we aim to combine the complementing advantages of the above mentioned works while filling in some missing gaps. We utilize knowledge-based sub-tree transformations, following (Bar-Haim et al., 2007a), but augment them with a set of *on the fly* transformations that correspond to syntactically-motivated entailment inferences, whose reliability can be learned using syntactic features. We further apply a cost model for entailment proofs and introduce an iterative learning scheme that estimates reliability weights based on the "best" (lowest-cost) proofs for the training pairs.

Evaluations show that our current implementation, including an initial set of knowledge resources and linguistic analysis, achieves compa-

[1] http://pascallin.ecs.soton.ac.uk/Challenges/RTE/
[2] http://www.nist.gov/tac/

Proceedings of Recent Advances in Natural Language Processing, pages 455–462,
Hissar, Bulgaria, 12-14 September 2011.

rable results to other proof-based systems. We conclude the paper by pointing at the generality and flexibility of our framework and suggest several research directions which can be naturally integrated into it.

2 Background

As pointed above, a promising approach in RTE research is applying a sequence of operations (a proof) on the given text, **T**, to reveal whether and how it entails the hypothesis, **H**. The advantage of this approach is that it allows composition of knowledge, where in many cases information from one knowledge resource becomes relevant only after a previous operation was performed.

Methods that follow this approach should deal with three main aspects. First, they have to decide how to represent **T** and **H**. Second, they have to define the set of proof operations. Third, they have to define a method to estimate the likelihood that a generated sequence of operations indeed preserves entailment.

Raina et al. (2005) used a logical representation, and accordingly defined the set of operations by a commonly used theorem proving method (resolution refutation). However, since state-of-the-art methods that transform a text into logical representation are less robust than syntactic parsers, the logical representation is rarely used.

Syntactic parse trees provide a common representation in text understanding systems in general, and for RTE in particular. The corresponding proof operations are thus tree-transformations that subsequently change the parse tree of **T** until **H**'s parse tree is obtained.

Mostly, the selected tree-transformations followed standard ("insert", "delete", "substitute") or custom tree edit distance operations (Mehdad and Magnini, 2009; Wang and Manning, 2010; Heilman and Smith, 2010) However, those sets of operations are often not linguistically-motivated and thus do not necessarily reflect the nature of the RTE problem. In addition, utilizing knowledge resources (both linguistic knowledge and world knowledge) is limited in such systems. Consider, for example, transformation of a parse-tree from a passive form to an active form. Such transformation can be done by a sequence of mostly deletion and insertion operations, however, such sequence misses to capture the syntactic structure of the transformation. Similarly, resources that indicate semantic similarity of two sub-trees, e.g. DIRT (Lin and Pantel, 2001), would be utilized naturally by substitution of a complete sub-tree by another, which cannot be performed by the above tree-edit-distance operations.

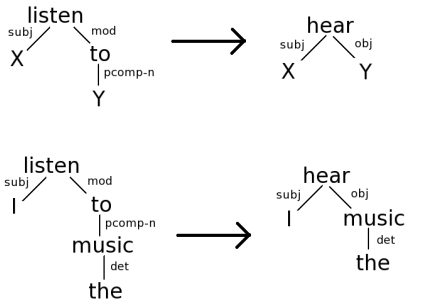

Figure 1: The figure demonstrates the rule *X listen to Y* → *X hear Y*, along with its application to the sentence "I listen to the music."

In contrast, a set of linguistically-motivated operations was proposed independently by Harmeling (2009) and by Bar-Haim et al. (2007a). While the set of operations defined by Harmeling (2009) was limited and included mostly ad-hoc heuristics, the operations defined by Bar-Haim et al. (2007a) were designed to capture a broad range of linguistic and world knowledge. Their primary operations are applications of *Entailment Rules*, which substitute complete sub-trees and generate new parse-trees, based on knowledge resources (see Figure 1 and Table 1).

However, unlike other methods that followed the proof-style approach, the method of Bar-Haim et al. (2007a) does not estimate the likelihood that a generated proof is valid. Another problem is that in most cases, there is no sequence of operations that completely generates **H**. Rather, starting from **T**, the operations generate new trees that become more similar, but not identical to **H**[3] (Bar-Haim, 2010).

To summarize, two main challenges are involved in transformational proof-style inferences over syntactic parse trees. The first is defining a method to estimate the likelihood that a given proof preserves entailment. The second is to define operations that are linguistically-motivated and reflect the RTE problem space. So far, all proof-style systems addressed either the first or the

[3]On the RTE datasets, a hybrid framework was introduced by Bar-Haim et al. (2007b), which uses an approximate match mechanism for final classifications.

Rule Type	Description	Examples
Lexical Rules	Substitution of a single node, capturing lexical entailment. Both *lhs* and *rhs* are single nodes.	novel → book walk → go
Lexical Syntactic Rules	Tree transformations that change the tree's lexical items as well as the tree's structure.	"X file lawsuit against Y" → "X accuse Y" "X listen to Y" → "X hear Y"
Generic Syntactic Rules	Tree structure transformations. Capture linguistic phenomena (e.g. passive-active).	X *V(active)* Y → Y is *V(passive)* by X

Table 1: Types of Entailment Rules. Note that for simplicity the examples are presented as strings, though the actual definition and implementation are based on sub-trees, as in Figure 1

second challenge, but not both. As described next, in this work we propose a principled integrated solution to those two challenges.

3 A Cost-based Proof Model

In our framework we adopt the linguistically-motivated entailment operations proposed by Bar-Haim et al. (2007a), and extend them with syntactically-motivated on-the-fly operations to enable generation of complete proofs (Sec. 3.1). The extended framework is then integrated with a learning method similar to the one proposed for logic representations by (Raina et al., 2005) as follows. We propose a cost model, which assigns a cost for each entailment proof (Sec. 3.2), and introduce a search algorithm that finds the "best proof" with respect to the cost model (Sec. 3.3). Finally we describe a method to iteratively learn the parameters of the cost model (Sec. 3.4).

3.1 Inference Formalism

The model presented here assumes a single-sentence hypothesis, similar to the RTE challenges, though it can be easily adjusted to multi-sentence hypotheses as well.

Given a (**T**,**H**) pair, the system first constructs the dependency parse trees[4] of **T** and **H**. Each node in those trees contains information about one lexical item (i.e. a word or a multi-word expression), which includes its lemma and its part-of-speech, and optionally other information, such as Named Entity type[5]. Each edge is labelled with a dependency relation (e.g. *subject*, *object*).

Let $\mathcal{T}$ be a set of dependency parse trees that were constructed for **T**'s sentences, and let h be the dependency parse tree constructed for **H**. The system iteratively extends $\mathcal{T}$ with additional trees, by applying *tree generation operations*, until there exists a tree $t \in \mathcal{T}$, such that h is embedded in t.

[4]We used the Minipar parser (Lin, 1998b)

[5]We used Stanford NE recognizer (Finkel et al., 2005)

We will use the following notations: Let $\mathcal{T}$ be a set of trees, o be a tree generation operation, and t be a tree. $\mathcal{T} \vdash_o t$ denotes that t can be generated from $\mathcal{T}$ using the operation o. We will use the $\vdash$ notation also for the resulting extended set of trees, that is:

$$\mathcal{T} \vdash_o \mathcal{T} \cup \{t\}$$

Let $O = (o_1, o_2 \ldots o_m)$ be a sequence of operations. The notation $\mathcal{T} \models_O \mathcal{T}'$ means that $\mathcal{T}'$ can be generated from $\mathcal{T}$ by applying iteratively the operations in O. Finally, a sequence of operations is called a *proof*, P, if $\mathcal{T} \models_P \mathcal{T}'$ such that h is embedded in one of the trees in $\mathcal{T}'$.

Although a more accurate definition of a proof would require that h would be identical to one of the trees in $\mathcal{T}'$, rather than being embedded in one of them, our relaxed definition is a common heuristic simplifying the proof construction process.

3.1.1 Entailment rules

The primary operations in Bar-Haim et al. (2007a) are applications of *Entailment Rules*. An entailment rule is composed of two sub-trees, named *left hand side* (*lhs*) and *right hand side* (*rhs*), intended to capture an entailment relation between its two sides (See Table 1). For example, a simple lexical rule is "music → art", where both sub-trees consist of single nodes.

Let *r = (lhs, rhs)* be a rule and t be a parse-tree, such that *lhs* is embedded in t. An *application* of r on t is a generation of a new tree, t', which is identical to t, but with the instance of *lhs* in t being replaced by *rhs*. If the underlying meaning of t entails the meaning of t', then we would consider the application of r as *valid*. It should be noted that in (Bar-Haim et al., 2007a) all rule-applications, based on the set of rules given to the system, were considered valid for any arbitrary (**T**,**H**) pair, an assumption which we relax in our cost-based model.

A rule's *lhs* and *rhs* may contain *variables*, i.e.

nodes in which the lemma is not specified. When such a rule is applied, the system first instantiates the variables with actual lemmas, according to the original tree, and then replaces the *lhs* by the instantiated *rhs* (As exemplified in Figure 1). As described in Section 2 and Table 1, such entailment rules are able to capture a broad range of linguistic and world knowledge. It should be noted that in our current implementation generic-syntactic rules were not integrated yet. Incorporating and extending the set of generic-syntactic rules is currently under work.

3.1.2 Co-reference Operations

Co-reference Substitution is a tree manipulation that is performed according to co-reference information, given by an external co-reference resolver[6]. Given two mentions m_1 and m_2 of the same entity, not necessarily in the same parse-tree, we define the operation of replacing the sub-tree rooted by m_1 by the sub-tree rooted by m_2 as *Co-reference Substitution*.

3.1.3 On The Fly operations

As described in Section 2, the original scheme of Bar-Haim et al. (2007a) recognized a (**T**,**H**) pair as entailing if and only if **H** could be generated by a sequence of co-reference substitutions and applications of rules from the given set of knowledge resources. Inevitably that scheme suffers very limited recall[7].

Utilizing our learning scheme as described below, we are able to overcome that difficulty, by adding an additional set of *on the fly* tree-transformations. Though those operations are not justified by a pre-given knowledge base, an estimation of their correctness likelihood can be learned, based on syntactic features. For example, moving a complete sub-tree is defined as an atomic operation, in contrast to the regular tree-edit-distance operations, in which such transformation requires a sequence of "insert" and "delete" operations.

An initial set of on-the-fly operations which is implemented in our system is specified in Table 2. The validity of applying such operations is estimated by the cost-model, described next, using the features listed in Table 3. Those operations represent simple transformations required to handle differences between two dependency-parse-trees, and are applied when parts of the hypothesis tree are missing in a given tree in $\mathcal{T}$.

Operation-Name	Operation-Description
Insert Node	Insert a new node in an arbitrary position in a parse tree.
Move sub tree	Disconnect a sub tree rooted by n from its parent $p(n)$ and connect it as a child of another node in the tree, $p'(n)$.
Change Relation	Change the relation (the edge label) between a node n and its parent $p(n)$.
Flip Part-Of-Speech	Change a node's part-of-speech.
Cut Multi-Word	Remove some of the words from a multi-word expression, as identified by the parser
Single-Word to Multi-Word	Replace a word by a multi word expression containing it, e.g. "Bond" → "James Bond".

Table 2: *on-the-fly* operations in our system.

This set of operations can be extended in the future by using additional linguistic resources, e.g. by identifying the semantic role of the inserted and moved nodes, or by adding on-the-fly substitutions, scored by distributional similarity.

3.2 Cost Model

Given a proof P, we want to estimate its correctness likelihood. Under the assumption that some or all of the operations in P might be incorrect - for example due to inaccuracies of the knowledge bases, wrong co-reference resolution or incorrect on-the-fly operations - we define a *cost model* to quantify the proof's likelihood to be correct. Following the cost model applied by Raina et al. (2005) to logic proofs, we use an additive linear model in which each operation is characterized by a set of features and the operation's total cost is a weighted linear combination of those features. Formally, let $o \in P$, let $F^{(o)} = (F_1^{(o)}, F_2^{(o)}, \ldots F_D^{(o)})^T$ be a feature vector characterizing o, and let w be a corresponding *weight vector*. The total cost of o (denoted by $C_w(o)$) is defined as:

$$C_w(o) \triangleq \sum_{i=1}^{D} w_i \cdot F_i^{(o)} = w^T \cdot F^{(o)} \qquad (1)$$

The cost of a sequence of operations (and in particular of a proof) is naturally defined as the sum of costs of all operations. Thus, given a proof

[6] We used BART co-reference resolver (Versley et al., 2008)

[7] As mentioned earlier, to increase recall in practical RTE datasets, a hybrid framework was introduced by Bar-Haim et al. (2007b), which uses an approximate match mechanism for final classifications.

$P = (o_1, o_2, \ldots o_m)$, its total cost, denoted by $C_w(P)$, is:

$$C_w(P) \triangleq \sum_{j=1}^{m} C_w(o_j) \quad (2)$$

Let $F^{(P)} = \sum_{j=1}^{m} F^{(o_j)}$. Combining (1) and (2), we get:

$$C_w(P) \triangleq \sum_{i=1}^{D} w_i \cdot F_i^{(P)} = w^T \cdot F^{(P)} \quad (3)$$

The last equation provides a way to represent a complete proof by a single feature-vector, which is simply the sum of all operations' vectors. We will use this feature representation in the learning and classification phases.

For each (**T**,**H**) pair there might be many proofs. However, for positive pairs, we assume there exists a "correct" proof, i.e. a proof that is composed of only valid operations (though many other incorrect proofs exist as well), while for negative pairs non of the proofs is correct. An optimal weight vector, w^*, would assign low costs to correct proofs while incorrect proofs will be assigned high costs. Therefore, distinguishing between positive pairs and negative pairs should be done by examining their lowest-cost proofs.

In the next sub-sections we describe how to search for lowest-cost proofs ("best proofs") and how to learn the optimal weight vector.

3.2.1 Modelling Operations by Features

As a convention, all features are assigned zero-or-negative values, interpreted as penalty. For each value v_i assigned to a feature F_i, $v_i = 0$ means that no penalty is implied by that feature, while $|v_i| \gg 0$ implies a high penalty by that feature. Following that convention, all weights should be assigned zero-or-positive values, since adding an operation cannot improve the confidence of a proof. This implies that an operation's total cost $C_w(o)$, and a proof's total cost $C_w(P)$ are zero-or-negative. The higher the absolute cost value, the lower the likelihood of the proof's correctness.

Features were defined for each knowledge resource, for co-reference substitution and for on-the-fly operations, as summarized in Table 3. For knowledge resources, features were defined as follows. Many knowledge resources provide numerical scores, indicating rules' reliability, which we use for the corresponding feature value. The knowledge resources that provide such scores and were used in the current system are DIRT (Lin and Pantel, 2001), Wikipedia rules (Shnarch et al., 2009), Lin similarity (Lin, 1998a), and Directional-Similarity[8] (Kotlerman et al., 2010). For knowledge resources that do not provide a numerical information about rule reliability, the corresponding feature-value is set to (-1). In the current system, WordNet[9] (Fellbaum, 1998; Miller, 1995), an in-house Geographical data-base, and VerbOcean[10] (Chklovski and Pantel, 2004) were included.

Some on-the-fly operations incorporate numerical information that reflects how likely it is that the meaning of the text is changed by applying them. As an example, for the insert-node operation we use the "Maximum Likelihood Estimation" (MLE) of the occurrence probability of the inserted word in a large news corpus[11]. The underlying assumption here is that it is more likely that inserting frequent words would still preserve entailment than inserting rare words.

3.3 Searching for the best proof

Searching for the best proof is done iteratively. Starting from $\mathcal{T}$ as the original text's trees, and a given weight vector, the system adds all the trees that can be generated by applying any generation-operation on $\mathcal{T}$. Since that scheme makes $\mathcal{T}$ grow exponentially, we use a simple beam search pruning approach as follows.

A constant beam size K is predetermined. In each iteration $\mathcal{T}$ is pruned such that its number of trees will be no more than K. Since every tree in $\mathcal{T}$ was generated by a sequence of operations, we define the cost of a tree as the cost of the sequence that was used to generate that tree. We use that cost, in addition to estimations about the difference between a given tree to the hypothesis tree, in order to decide which tree should be pruned out, such that after each iteration $|\mathcal{T}| \leq K$. Finally, the lowest cost generated tree which embeds h is returned.

[8] A rule-base of lexical entailment rules automatically extracted by means of directional distributional similarity.

[9] We used the following WordNet relations: *hypernymy*, *holonymy*, *verb-entailment* and *synonymy*

[10] Only the relation "stronger" was used.

[11] We used Reuters Corpus, Volume 1+2 (RCV1-2). Available at http://trec.nist.gov/data/reuters/reuters.html

#	Feature	Value
1	Wikipedia	$\log(m)$, where m is the estimated accuracy of the method used to learn the given Wikipedia rule, as described in (Shnarch et al., 2009). $0 \leq m \leq 1$.
2	Lin Similarity	$\log(sim)$, where sim is the similarity score given for that rule according to (Lin, 1998a). $0 \leq sim \leq 1$.
3	Directional-Similarity	$\log(sim)$, where sim is the similarity score given for that rule according to (Kotlerman et al., 2010). $0 \leq sim \leq 1$.
4	DIRT	$\log(sim)$, where sim is the similarity score given for that rule according to (Lin and Pantel, 2001). Note that $0 \leq sim \leq 1$.
5	WordNet	-1
6	VerbOcean	-1
7	Geographical Database	-1
8 **9** **10** **11**	Insert Verb Insert non-verb content word Insert non-content word insert Named Entity	$\log(f)$, where f is the MLE of the occurrence probability for the inserted lemma in the Reuters news corpus.
12, 13, 14, 15	Insert verb / content word / non-content word / Named Entity - that exist in other parts of the text	$\log(f)$, where f is the MLE of the occurrence probability for the inserted lemma in the Reuters news corpus.
16	Change relation of a node to its parent, from "subject" to "object" or vice versa	-1
17	Move Sub Tree rooted by n from $p(n)$ to $p'(n)$, s.t. the path from n to $p'(n)$ contains a verb	$-l$, where l is the length of the path between n and $p'(n)$ in the original tree.
18	All other "move Sub Tree" operations	
19	Single-word to Multi-word	$\log(\min_{f \in \mathcal{F}}(f))$ where $\mathcal{F}$ is the set of MLE of the occurrence probabilities corresponding to the added words. The probabilities were calculated using the Reuters News corpus.
20	Cut Multi-word	-1
21	Flip part-of-speech	-1
22	Co-reference	-1

Table 3: Features and their values for each (knowledge and on-the-fly) operation. Note that all values are negative.

3.4 Iterative Weight Estimation

We would like to classify a proof P, represented by a feature vector F, as "correct" if its cost is low.

Formally, let (w, b) be a weight vector and a threshold. P is classified as correct if and only if

$$w \cdot F + b \geq 0 \tag{4}$$

and as incorrect otherwise. The goal of parameter estimation is thus finding optimal (w^*, b^*).

If our training set was a set of binary-labelled vectors (F_i, l_i), $i \in \{1 \dots n\}$, we could apply directly a linear training algorithm to find (w^*, b^*). However, our training set is a set of labelled text pairs, for which the proofs that determine the corresponding feature vectors should be constructed by the system. Yet, as explained at the end of Section 3.2, only the lowest-cost proofs should be considered to distinguish between positive and negative pairs, while finding those proofs through the search algorithm of Section 3.3 requires knowing the optimal weight vector.

We therefore use an iterative learning scheme to overcome this circularity problem, as follows (see Algorithm 1). We start with an initial weight vector and threshold, (w_0, b_0), set manually by a reasonable guess. Using the algorithm in Section 3.3 we find a lowest-cost proof for each pair, resulting in n labelled feature vectors, $(F_1, l_1) \dots (F_n, l_n)$, where l_i is the binary entailment annotation. Next, we use a standard linear learning algorithm to learn new parameters, (w_1, b_1). We iteratively improve the weights vectors and the proofs until con-

Algorithm 1 Parameters Estimation

Require: Training set: ($\mathbf{T}_1$,$\mathbf{H}_1$,l_1) ... ($\mathbf{T}_n$,$\mathbf{H}_n$,l_n)
1: $(w_0, b_0) \leftarrow$ a reasonable guess of weights vector and threshold
2: $i \leftarrow 0$
3: **repeat**
4: Find $P_1 \dots P_n$ by the method described in 3.3, using (w_i, b_i)
5: Construct the corresponding feature vectors $F^{(P_1)} \dots F^{(P_n)}$.
6: use $(F^{(P_1)}, l_1) \dots (F^{(P_n)}, l_n)$ as a training set to a linear classifier, resulting new parameters (w_{i+1}, b_{i+1}).
7: $i \leftarrow i + 1$
8: **until** convergence

System	RTE-1	RTE-2	RTE-3	RTE-5
Learning and abductive reasoning (Raina et al., 2005)	57.0 %			
Probabilistic Calculus of Tree Transformations (Harmeling, 2009)		56.39 %	57.88 %	
Probabilistic Tree Edit model (Wang and Manning, 2010)		63.0 %	61.10 %	
Deterministic Entailment Proofs (Bar-Haim et al., 2007b)			61.12 %	63.80 %
Our System Accuracy (Recall % / Precision %)	57.13% (81.0/54.8)	61.63% (76.2/59.0)	67.13% (87.2/63.3)	63.50% (75.7/60.9)
Median of all submissions in challenge	55.20 %	58.13 %	61.75 %	61.00 %
Best System in challenge	58.6 %	75.3 %	80.0 %	73.5 %

Table 4: Accuracy of proof-based systems on RTE datasets, followed by median results and best results of all systems participated in those challenges.

vergence. Since there is no theoretical bound on the convergence rate, we limit the number of iterations by a predefined constant. In practice, however, only few iterations are required for convergence.

4 Evaluation

We ran experiments on the first, second, third and fifth RTE datasets[12] (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) and compared our system to other proof-style systems. Each dataset consists of 600 to 800 (**T**,**H**) pairs, half of them are positive, and the other half are negative. For the underlying linear classifier, required by Algorithm 1, we used linear-SVM[13]. The value of K, described in Section 3.3, was set to 135, according to tuning done on the training set. The results for our system, presented in Table 4, show comparable performance to other systems on most datasets, with notably higher performance in RTE-3.

Operation	avg. in positives	avg. in negatives	ratio
Insert Named Entity	-0.006	-0.016	2.67
Insert content word	-0.038	-0.094	2.44
DIRT	-0.013	-0.023	1.73
"subject" ↔ "object"	-0.025	-0.040	1.60
Flip part-of-speech	-0.098	-0.101	1.03
Lin similarity	-0.084	-0.072	0.86
WordNet	-0.064	-0.052	0.81

Table 5: The average value of certain features in positive pairs and negative pairs, taken from an experiment on the RTE-2 test set.

As indicated by the results, our system indeed assigns, on average, higher costs to negative pairs than to positive ones. Further insight into this behavior is obtained by Table 5. The table presents a sample of features' average values. The upper rows of the table present features whose average absolute value in negative pairs is significantly higher than in positive pairs, while the features in the lower rows have similar average values in positive and negative pairs.

The former features indicate that there are some operations that tend to be part of the "best" proof for negative pairs more frequently than for positive pairs. A reasonable explanation for this phenomenon is that the system learned that some operations are less reliable than other operations, and tried to avoid them whenever possible. However, these operations could not be avoided in negative pairs, resulting in higher feature values.

5 Conclusions and Future Work

Two main concepts underlie this paper. The first is automatic estimation of the quality of proofs required to recognize textual-entailment. The second concept is a complete framework of linguistically-motivated proof operations for recognising textual-entailment. The main contribution of this paper is showing how those two concepts can be integrated, to leverage the advantages of both.

The linguistically-motivated framework presented here is based on the framework proposed by (Bar-Haim et al., 2007a), with a significant extension of on-the-fly operations required for making it robust and complete. Many additional linguistically-motivated entailment operations can be naturally integrated into this framework. For example, lexical, syntactic and semantic attributes like verb-tense and polarity (negation) can be easily handled, much like part-of-speech and named-entity (Bar-Haim et al., 2007a). Another example is temporal inference (e.g. "this afternoon → today") which can be integrated easily by proper substitutions based on an appropriate knowledge

[12] The RTE-4 dataset had no training dataset

[13] We used SVM-Light, available at http://svmlight.joachims.org/

resource (similar to the one proposed by Wang and Zhang (2008)). Yet another example is addressing more types of co-reference based operations (Mirkin et al., 2010). Finally, as noted earlier, the current set of on-the-fly operations may be extended, which will likely improve the system's performance.

Acknowledgements

This work was partially supported by the Israel Science Foundation grant 1112/08 and by the PASCAL-2 Network of Excellence of the European Community FP7-ICT-2007-1-216886.

References

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Roy Bar-Haim, Ido Dagan, Iddo Greental, and Eyal Shnarch. 2007a. Semantic inference at the lexical-syntactic level. In *Proceedings of AAAI*.

Roy Bar-Haim, Ido Dagan, Iddo Greental, Idan Szpektor, and Moshe Friedman. 2007b. Semantic inference at the lexical-syntactic level for textual entailment recognition. In *Proceedings of ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Roy Bar-Haim. 2010. *Semantic Inference at the Lexical-Syntactic Level*. Ph.D. thesis, Bar-Ilan Univeristy.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, H.T. Dang, and D. Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *proceeding of TAC*.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of EMNLP*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In Joaquin Quiñonero Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors, *MLCW*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs Sampling. In *Proceedings of ACL*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *RTE '07 Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Stefan Harmeling. 2009. Inferring textual entailment with a probabilistically sound calculus. *Natural Language Engineering*, 15(4):459–477.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases and answers to questions. In *HLT-NAACL*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16:359–389.

Dekang Lin and Patrick Pantel. 2001. DIRT - discovery of inference rules from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Dekang Lin. 1998a. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*.

Dekang Lin. 1998b. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on Evaluation of Parsing Systems at LREC 1998*, Granada, Spain.

Yashar Mehdad and Bernardo Magnini. 2009. Optimizing textual entailment recognition using particle swarm optimization. In *Proceedings of the 2009 Workshop on Applied Textual Inference*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Shachar Mirkin, Ido Dagan, and Sebastian Pado. 2010. Assessing the role of discourse references in entailment inference. In *Proceedings of ACL*.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI*.

Eyal Shnarch, Libby Barak, and Ido Dagan. 2009. Extracting lexical reference rules from Wikipedia. In *ACL-IJCNLP*.

Yannick Versley, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Ro Moschitti. 2008. BART: A modular toolkit for coreference resolution. In *Proceedings of ACL, Demo Session*.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of COLING*.

Rui Wang and Yajing Zhang. 2008. Recognizing textual entailment with temporal expressions in natural language texts. In *Proceedings of the IWSCA*.