

Knowledge and Tree-Edits in Learnable Entailment Proofs

Asher Stern¹, Amnon Lotan³, Shachar Mirkin¹, Eyal Shnarch¹, Lili Kotlerman¹, Jonathan Berant², and Ido Dagan¹

¹Computer Science Department, Bar-Ilan University, Ramat Gan 52900, Israel

²School of Computer Science, Tel-Aviv University, Ramat Aviv 69978, Israel

³Department of Linguistics, Tel-Aviv University, Ramat Aviv 69978, Israel

October 25, 2011

Abstract

This paper describes BIUTEE - Bar Ilan University Textual Entailment Engine. BIUTEE is a natural language inference system in which the hypothesis is proven by the text, based on linguistic- and world- knowledge resources, as well as syntactically motivated tree transformations. The main progress in BIUTEE in the last year is a new confidence model that estimates the validity of the proof found by BIUTEE.

1 Introduction

This paper describes the main concepts of BIUTEE - “Bar Ilan University Textual Entailment Engine” - in the Seventh Recognizing Textual Entailment (RTE-7) challenge. The RTE task is formalized as follows: Given a pair of natural language text fragments, named text and hypothesis, the system needs to recognize whether the hypothesis can be inferred from the text. Our approach for addressing the RTE task, as first described in (Bar-Haim et al., 2007a), is to explicitly find out how the hypothesis can be inferred from the text, based on linguistic- and world-knowledge. More concretely, our system finds a sequence of inference steps (a.k.a. *a proof*), such that each step would ideally be inferred from the previous one based on some knowledge of inference operation. Since the sequence begins with the text and ends with the hypothesis, we conclude that the hypothesis can be inferred from the text, based on the knowledge used in the sequence.

This approach, despite many efforts to acquire large and accurate knowledge resources, still suffers from two inherent difficulties. First, in most practical cases the knowledge bases available to the system are insufficient to construct a complete sequence of inference steps. Usually, only a partial sequence can be constructed. Second, most of the knowledge bases, especially those that were automatically acquired, contain also inaccurate knowledge, which may lead to wrong inferences.

In order to cope with practical RTE datasets, the developers of our earlier version of BIUTEE augmented the system with an *Approximate Match* mechanism (Bar-Haim et al., 2008). This mechanism extracts features from the consequents of partial proofs that were found by the system and from the hypothesis. The features were designed to capture entailment phenomena, like polarity, predicate-argument match, lexical coverage and tree structure similarity. The approximate match mechanism then uses a machine learning classifier to classify the text-hypothesis pair as entailing or non-entailing. However, that mechanism broke the main concept of a proof system. More concretely, the system became a hybrid system, with one mechanism that aims to find consequents derived from the text, and another separate post-processing mechanism that aims to find similarities and dissimilarities between the hypothesis, and the consequents derived from the text. One of the proof scheme advantages is its ability to provide a clear explanation of how the hypothesis can be inferred from the text. This explanation is the sequence of inference-steps. The hybrid system lacks

that advantage, since there was no longer a direct connection between the proof process and the final decision on entailment.

We now address this issue with our new confidence model, first introduced in (Stern and Dagan, 2011). In this model, the system is allowed to perform some operations even if they are not justified by the available knowledge resources. The validity of those operations, as well as that of knowledge-based operations, is estimated by the confidence model. It is expected that the proofs found for positive text-hypothesis pairs (i.e. pairs in which the text entails the hypothesis) will be assigned a high confidence, while the proofs found for negative pairs will be assigned a low confidence. Thus, the system can find a proof for any text-hypothesis pair, and evaluate its validity.

In the rest of this paper we survey the earlier version of BIUTEE, as well as other related works (Section 2). Then we describe our inference framework and the confidence model (Section 3), followed by the results obtained on the RTE-7 dataset (Section 4). In Section 5 we describe the advantages of BIUTEE as an open source system. Section 6 concludes this paper with suggestions for future work.

2 Background and Related Works

The Bar-Ilan University approach to the task of Recognizing Textual Entailment is to prove the hypothesis by the text. This approach was developed in a continuous research line during the last years. The main idea, described in (Bar-Haim et al., 2007a), is an inference framework, in analogy to logic-based inference systems, which operates over syntactic-parse-trees. Like logic-based systems, the inference framework is composed of propositions and inference rules. The propositions include t (the given text), h (the hypothesis to be proven), and intermediate premises inferred during the proof. To construct a proof, the inference framework uses inference (entailment) rules that define how new propositions are derived from previously established ones.

At each step of the proof an entailment rule generates a derived tree (d) from a source tree s . A rule " $L \rightarrow R$ " is primarily composed of two templates, termed *left-hand-side* (L), and *right-hand-side* (R). Templates are dependency subtrees which may con-

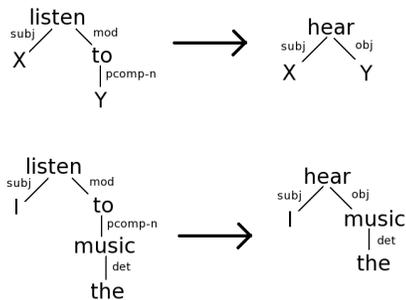


Figure 1: DIRT-style rule $X \text{ listen to } Y \rightarrow X \text{ hear } Y$, along with its application to the sentence “I listen to the music.”

tain variables. An example of such a rule is shown in Figure 1. A detailed definition of entailment rules and their application is given in Section 3.1.1.

Given a text-hypothesis (t, h) pair, the system detects whether t entails h as follows: The system starts with a set, \mathcal{T} , of the text’s parse trees. Then it generates all new consequents (which are themselves parse trees) that can be derived from the existing ones, using the available entailment-rules, and adds them to \mathcal{T} . The system continues to iteratively generate new consequents based on the existing ones, and adds them to \mathcal{T} , until no more consequents can be generated. Then, the system decides that the text entails the hypothesis, if and only if the hypothesis parse tree is included in \mathcal{T} .

Empirically, that scheme suffers from two significant limitations: First, the requirement of $h \in \mathcal{T}$ leads to very low recall, since complete derivation of h from t is not feasible for most RTE (t, h) pairs (Bar-Haim, 2010). Second, many knowledge resources from which entailment rules were acquired are far from being accurate. Nevertheless, the system did not take that inaccuracy into account.

A more realistic approach would treat the applied operations as *correct with some probability*. Such an approach was suggested by several works. For example, Harmeling (2009) introduced a calculus that can transform the given text, t , into the hypothesis h . His underlying assumption was that in practice such calculi will always remain imperfect due to the difficulty of correctly formalizing commonsense reasoning. Thus, the calculus is considered as *probabilistically* sound calculus, in which each atomic transformation preserves truth with some probability. Given

this calculus, one can assign a probability to a complete derivation in the calculus simply by multiplying all probabilities of preserving the truth along the derivation.

The core idea in the above mentioned work is not only constructing a proof, but also estimating the likelihood that a constructed proof is indeed valid (i.e. preserves truth). The same idea was proposed independently by other works: In the work of Raina et al. (2005), a logical proof is constructed, by a framework of *abductive reasoning*, and its validity is estimated using a machine learning algorithm. In another line of works, in which *Tree-Edit Distance* is used to recognize textual entailment (see, for example, (Mehdad, 2009) and (Heilman and Smith, 2010)) the proof is a sequence of Tree-Edit operations, and its validity is estimated by the cost assigned to that sequence.

However, in these works proofs were not constructed over entailment rules, which seem to provide a better formalism for textual inference. By using entailment rules, natural language inference phenomena can be formulated as atomic operations, in contrast to the operations suggested by the above systems, in which many inference phenomena require a sequence of operations. For example, transforming a text from passive form to active form is formalized by a single entailment rule, compared to a sequence of insertions and deletions required in the tree-edits models. Thus, our goal was to preserve an inference process over entailment rules, while addressing the limitations described above. In the new version of BIUTEE the core operations are entailment rule applications. Since our available set of entailment rules is inevitably incomplete, we defined a set of syntactically motivated on-the-fly tree edit operations that are used when knowledge is insufficient for a given (t, h) pair. The system finds a proof of the hypothesis from the text, and uses a confidence model to evaluate its validity, as follows: The system calculates for every operation of either type a cost, based on linguistic and world-knowledge features. Then it calculates a total cost for the complete proof, based on the costs of the proof’s operations. This cost is then used to estimate the proof validity.

3 The Cost-based Proof Model

In our framework we use entailment rules 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 trees¹ of t and h . Each node in those trees contains information about one lexical item (i.e. a word), which includes its lemma and its part-of-speech, and optionally other information, such as Named Entity type². Each edge is labeled 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 .

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 $O = (o_1, o_2 \dots o_m)$ 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

¹We used the EasyFirst parser (Goldberg and Elhadad, 2010)

²We used Stanford NE recognizer (Finkel et al., 2005)

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

Our primary operations, as described in (Bar-Haim et al., 2007a) are applications of entailment rules. An entailment rule is composed of two subtrees, 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 \rightarrow art”, where both subtrees 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).

A broad range of linguistic- and world- knowledge can be captured by entailment rules. In general, there are three main types of entailment rules: *Lexical*, *Lexical-Syntactic* and *Generic Syntactic*.³

Lexical Rules: These are the simplest rules, in which both *lhs* and *rhs* are single words (or multi-word expressions), for example, “music \rightarrow art”. The meaning of this rule is that given a tree with a node with the lemma “music”, this node can be replaced

³Most of the resources used by our system are available in <http://www.cs.biu.ac.il/~nlp/downloads/>. Some of them are part of BIUTEE package in <http://www.cs.biu.ac.il/~nlp/downloads/biutee>

by a similar node with the lemma “art”, while the resulting tree would be entailed from the original one.

We used the following lexical resources: Wikipedia rules (Shnarch et al., 2009), Lin similarity⁴ (Lin, 1998), Directional-Similarity⁵ (Kotlerman et al., 2010), WordNet⁶ (Fellbaum, 1998; Miller, 1995), a geographic resource based on TRECs TIPSTER gazetteer (see (Mirkin et al., 2009)) and VerbOcean⁷ (Chklovski and Pantel, 2004)

Lexical Syntactic Rules: These rules not only take into account the lexical items that can be replaced, but also the syntactic structure in which they are organized in the parse tree. Usually, those rules deal with entailment of predicates (either as verbs or as nominalizations), and contain variables that specify their arguments, which are instantiated during rule application. An example of a lexical-syntactic rule is “buy Y for Z \rightarrow pay Z for Y”. Applying this rule on the sentence “Microsoft bought YaData for \$25 million.” would result in “Microsoft paid \$25 million for YaData.”.

We used the following lexical-syntactic resources: DIRT⁸ (Lin and Pantel, 2001) and FrameNet based rules (Ben-Aharon et al., 2010)

Generic Syntactic Rules: The generic-syntactic rule base is comprised of a few dozen manually written inference rules that are syntactically oriented, i.e. they do not involve open class lexical items. They are based on a manual analysis of the RTE dataset, as well as on previous work included in older versions of our system (Bar-Haim et al., 2007a; Bar-Haim, 2010), and on other related works such as (Amoia and Gardent, 2008) and (Hearst, 1992).

Generic syntactic rules capture entailment patterns associated with common syntactic constructs, as summarized in Table 2,

Table 1 summarizes the types of entailment rules.

⁴retrieving for each left-hand-side only the top 30 right-hand-sides

⁵A rule-base of lexical entailment rules automatically extracted by means of directional distributional similarity.

⁶We used the following WordNet relations: *hypernymy*, *holonymy*, *verb-entailment* and *synonymy* - all with depth 2

⁷Only the relation “stronger” was used.

⁸retrieving for each left-hand-side only the top 20 right-hand-sides

Rule Type	Description	Examples
Lexical Rules	Substitution of a single node, capturing lexical entailment. Both <i>lhs</i> and <i>rhs</i> are single nodes.	novel \rightarrow book walk \rightarrow 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” \rightarrow “X accuse Y” “X listen to Y” \rightarrow “X hear Y”
Generic Syntactic Rules	Tree structure transformations. Capture linguistic phenomena (e.g. passive-active).	X <i>V(active)</i> Y \rightarrow Y is <i>V(passive)</i> 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

Syntactic Construct	Example
Apposition	<i>Ted, the boss, is coming</i> \rightarrow <i>Ted is the boss</i>
Conjunction	<i>Ted has been in meetings and appointments</i> \rightarrow <i>Ted has been in appointments</i>
Genitive	<i>Ted’s wife is also coming</i> \rightarrow <i>The wife of Ted is also coming</i>
Implications	<i>Ted likes apples and other fruit</i> \rightarrow <i>Apples are fruit</i>
Relative Clauses	<i>I’m looking for someone invited by Paula</i> \rightarrow <i>someone was invited by Paula</i>
Passive-Active	<i>Ted was invited by Paula</i> \rightarrow <i>Paula invited Ted</i>

Table 2: Generic Syntactic Rules

3.1.2 Co-reference substitutions

Co-reference Substitution is a tree manipulation that is performed according to co-reference information, given by an external co-reference resolver⁹. 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 from a very limited recall¹⁰.

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 cor-

rectness 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.

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” \rightarrow “James Bond”.

Table 3: *on-the-fly* operations in our system.

An initial set of *on-the-fly* operations which is implemented in our system is specified in Table 3. The validity of applying such operations is estimated by

⁹We used ArkRef co-reference resolver (www.ark.cs.cmu.edu/ARKref/). See (Haghighi and Klein, 2009)

¹⁰As mentioned earlier, to increase recall in practical RTE datasets, a hybrid framework was introduced in (Bar-Haim et al., 2007b; Bar-Haim et al., 2008), which uses an approximate match mechanism for final classifications.

the cost-model, described next, using the features listed in Table 4. 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} .

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 $F^{(o)} = (F_1^{(o)}, F_2^{(o)}, \dots, F_D^{(o)})^T$ 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:

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

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 $P = (o_1, o_2, \dots, o_m)$, 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 may 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 sections 3.3 and 3.4 we describe how to search for lowest-cost proofs (“best proofs”) and how to learn the optimal weight vector.

3.2.1 Modeling 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 4. 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. For knowledge resources that do not provide a numerical information about rule reliability, the corresponding feature-value is set to (-1) .

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

#	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, 1998). $0 \leq sim \leq 1$.
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	
7	Geographical Database	
8	FrameNet Rules	
9	Generic Syntactic Rules	
10	Insert Verb	$\log(f)$, where f is the MLE of the occurrence probability for the inserted lemma in the Reuters news corpus.
11	Insert non-verb content word	
12	Insert non-content word	
13	Insert Named Entity	
14	Insert Verb - that exist in other parts of the text	
15	Insert non-verb content - that exist in other parts of the text	
16	Insert non-content word - that exist in other parts of the text	
17	Insert Named Entity - that exist in other parts of the text	
18	Change relation of a node to its parent, from "subject" to "object" or vice versa	-1
19	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.
20	All other "move Sub Tree" operations	
21	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.
22	Cut Multi-word	-1
23	Flip part-of-speech	-1
24	Co-reference	-1

Table 4: Features and their values for each (knowledge-based, on-the-fly and coreference) operation. Note that all values are negative.

a large news corpus¹¹. 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

Equipped with a large set of knowledge resources and on-the-fly operations, our system can typically generate, in every single step, dozens of generated trees (~ 30 trees) for any given tree, yielding an exponential number of possible derivations. One solution to that exponential explosion, given by Bar-

Haim et al. (2009), is to store all derivations in a polynomial-space compact representation, utilizing the fact that many trees share the same parts. That solution, named *Compact Forest* was used in earlier versions of BIUTEE system. However, the problem we face in our system is not only a storage problem, but also a search problem: We have to find which derivation (out of the exponential number of possible derivations) is the best one (that is, has the lowest cost). In addition, we have to store a feature vector for each derivation, which is not supported by that compact representation.

On the other hand, we can utilize our cost model characteristics, which were not available in the ear-

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

lier version of BIUTEE used by Bar-Haim et al. (2009), in order to define a polynomial time and space search scheme. These characteristics are:

1. Each intermediate tree has a cost (calculated by the sequence of operations by which that tree was generated).
2. The gap (the difference) between an intermediate tree and the hypothesis tree is a relevant factor, since intermediate trees play no role in the final entailment decision.

We developed a search scheme using those two factors: The cost paid until now to generate a tree t is denoted by $g(t)$, and a measurement for the gap between t and the hypothesis tree is denoted by $h(t)$. Using $g(t)$ and $h(t)$ we can define a function $f(t)$ that estimates the “attractiveness” of t (a simple example is the famous A^* function: $f(t) = g(t) + h(t)$).

The general idea of the search scheme is as follows: In each iteration, pick up a subset of \mathcal{T} , based on their $f(t)$ value, generate all of their possible single-step consequents, and add them to \mathcal{T} . Then, again, based on the values of $f(t)$, prune \mathcal{T} , such that $|\mathcal{T}| \leq K$ (where K is a predefined parameter). Those iterations are performed in a loop until h is embedded in one of \mathcal{T} ’s trees. We empirically tested several variations of the search scheme. A detailed description of those variations is beyond the scope of this paper, and we plan to describe it in a separate publication.

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

$$w \cdot F + b \geq 0 \quad (4)$$

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

If our training set was a set of binary-labeled vectors (F_i, l_i) , $i \in \{1 \dots n\}$, we could apply directly a supervised linear learning algorithm to find (w^*, b^*) . However, our training set is a set of labeled text-hypothesis 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.

Algorithm 1 Parameters Estimation

Require: Training set: $(\mathbf{T}_1, \mathbf{H}_1, l_1) \dots (\mathbf{T}_n, \mathbf{H}_n, l_n)$

- 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

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 convergence. 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.

3.4.1 Negative feature weights and redundant operations

As explained in Section 3.2.1, all feature weights should be non-negative, reflecting the assumption that each proof-step can only reduce the confidence in the proof’s correctness. However, features that occur more frequently in the positive examples might be assigned negative weights by the learning algorithm, as it tries to separate the positive pairs from the negative ones (recall that feature values are

negative, hence multiplying them by a negative feature weight in the linear classifier would increase the classification score).

This conceptual anomaly causes problematic behaviour when searching for the best proof for a pair. With negative weights for certain inference operations, the search algorithm would always favor adding these operations to the proof, even if they are not necessary, as this would reduce the cost of the proof instead of raising it.

To prevent this situation, before running the search algorithm we first set all negative feature weights to zero, as was similarly done in (Raina et al., 2005). Next, we consistently increase all feature weights by a small constant, thus enforcing a non-zero cost for all operations in the search process.

Finally, we note that the modified weight vector is used only for the search algorithm, but not for actual classification. For classification (Formula (4)) we use the original weights as learned during training, to ensure consistency between the learning and classification scores (in synch with the optimal threshold learned).

4 Results

4.1 Main task

In the seventh RTE challenge, the main task is *Recognizing Textual Entailment within a Corpus*, defined as follows:

Given a corpus, a hypothesis H , and a set of “candidate” sentences retrieved by Lucene from that corpus for H , RTE systems are required to identify all the sentences that entail H among the candidate sentences.

The main task’s dataset is composed of 10 topics and each of them is composed of:

1. A number of hypotheses (between 25 and 45) referring to the topic. H ’s are standalone sentences taken from the TAC Update Summarization corpus¹².
2. A corpus of 10 documents.

3. For each H , a list of up to 100 candidate entailing sentences from the corpus. The candidate sentences are the 100 top-ranked sentences retrieved by Lucene, using H verbatim as the search query.

System results were compared to a human-annotated gold standard and the metrics used to evaluate system performances were Micro-Averaged Precision, Recall, and F1. To maximize the F1-measure we used *SVM-Perf*¹³ implementation of Support-Vector-Machine, which provides an option to maximize F1 (Joachims, 2005).

Using several preliminary experiments on RTE-6 datasets and RTE-7 development set, we decided to use only a subset of our knowledge resources. In particular, the following resources were used: WordNet, Directional-Similarity, Wikipedia, FrameNet and Geographical rule base.

To increase precision, we adopted the idea of Mirkin et al. (2009) to filter the candidate sentences by the Lucene IR system, in the following way: The hypothesis is given to Lucene as a query, which returns an ordered list of sentences as a result, ordered by relevance. Then, only the top N candidate sentences were judged by our system, while all the rest were classified as “non-entailing”. Following several preliminary experiments based on the RTE-6 datasets and the RTE-7 development dataset, we have chosen $N = 27$ by

We submitted 3 runs (i.e. 3 answer files) that differ from each other only in the set of knowledge resources that were used. Table 5 summarizes our 3 runs’ configurations and results.

4.2 Novelty detection subtask

In addition to the main RTE task, we participated in the novelty-detection subtask. The Novelty Detection subtask is based on the Main task and is aimed at specifically addressing the interests of the Summarization community, with regard to the Update Summarization task. The task consists of judging whether the information contained in each H is novel with respect to the information contained in the set of the candidate entailing sentences. If for a given H one or more entailing sentences are found then it

¹²TAC 2008 and 2009 Update Summarization Task

¹³http://svmlight.joachims.org/svm_perf.html

Id	Knowledge resources	Precision %	Recall %	F1 %
BIU1	WordNet, Directional Similarity	38.97	47.40	42.77
BIU2	WordNet, Directional Similarity, Wikipedia	41.81	44.11	42.93
BIU3	WordNet, Directional Similarity, Wikipedia, FrameNet, Geographical database	39.26	45.95	42.34

Table 5: Main task submissions

Id	Knowledge resources	Novelty-Detection Score			Justification Score		
		Precision %	Recall %	F1 %	Precision %	Recall %	F1 %
BIU1	WordNet, Directional Similarity	90.74	75.38	82.35	31.47	43.26	36.43
BIU2	WordNet, Directional Similarity, Wikipedia	91.61	72.82	81.14	32.41	41.85	36.53
BIU3	WordNet, Directional Similarity, Wikipedia, FrameNet, Geographical database	90.12	74.87	81.79	36.34	40.31	38.22

Table 6: Novelty-Detection subtask submissions

Tested Resource	% F1 with resource	% F1 without resource	% Δ
WordNet	42.93	42.98	-0.05
Directional Similarity	42.93	41.99	0.94
Wikipedia	42.93	41.37	1.56
Coreference Substitutions	42.93	42.24	0.69

Table 7: Ablation tests. Each row corresponds to one ablation test in which one resource was removed. The Δ column expresses the contribution of that resource as the difference between the F1 achieved with that resource to the F1 achieved without that resource.

means that the content of the H is not novel. On the contrary, if no entailing sentences are detected, it means that the information contained in H is regarded as novel.

While the dataset structure¹⁴ is identical to the main task, the scoring is different, and focuses on novelty detection. In particular, systems are judged by two scoring mechanisms:

1. The primary score is Precision, Recall and F1 computed on the binary novel/non-novel decision. The novelty detection decision is derived automatically from the number of justifications provided by the system (i.e. the entailing sentences retrieved for each H) - where 0 implies “novel”, 1 or more “non-novel”.
2. The secondary score measures the quality of the justifications provided for non-novel H’s,

¹⁴but not the data itself. In particular, only a subset of the main task’s hypotheses were included

that is the set of all the sentences extracted as entailing the H’s. The metrics used to this purpose are Micro-averaged Precision, Recall and F1.

We made no specific optimizations for the novelty-detection subtask, but rather ran our system the same way and with the same configurations used in the main task. Table 6 summarizes our novelty-detection subtask submissions.

4.3 Ablation tests

Ablation tests aim to collect data to better understand the impact of knowledge resources and tools used by the RTE system and evaluate the contribution of each resource to system performance. An ablation test consists of removing one module from the complete system, and re-running the system on the test set with the other modules. Comparing the results to those obtained by the complete system, it is possible to assess the actual contribution of the

individual module.

Our ablation tests were based on the second submission (BIU2), which obtained the highest value of F1 measure. Table 7 summarizes the ablation tests.

5 Open Source

The progress of RTE research in recent years made the leading RTE systems more and more complicated. Several preprocessing utilities (tokenization, part-of-speech tagging, parsing, named-entity-recognition, coreference resolution and more) as well as a large number of knowledge resources are used. If newcomers to the RTE community have to develop their own RTE systems from scratch, they will have to spend substantial effort in development before being able to evaluate their new ideas, let alone comparing them to other algorithms.

Addressing this issue, we released our system as open-source (www.cs.biu.ac.il/~nlp/downloads/biutee). The advantage of BIUTEE is in its extensibility and flexibility. Integration of new knowledge resources into BIUTEE can be done naturally by formalizing them as entailment rules. Furthermore, recognizing textual-entailment by generation of entailment proof is a flexible framework for exploiting additional natural-language phenomena, by defining appropriate operations and features.

6 Conclusions and Future Work

The new version of BIUTEE, presented in this paper, is an implementation of the new model described in (Stern and Dagan, 2011). The main contribution is a principled method to use knowledge resources, which have different levels of accuracy, along with coreference information and heuristic ad-hoc operations, in one integral framework. Moreover, the framework employs learning algorithms to decide how to assess the various resources and ad-hoc operations and how to estimate the correctness of their applications.

As for future work, we would like to have a better solution for the problem of learning negative weights (see Section 3.4.1). The current solution, which changes the weights as a post-processing step to the learning algorithm suffers from the limitation that the modified weight vector is no longer valid

for predictions of new instances ((t,h) pairs). Thus it only prevents redundant operations in the search process, but ignores the inherent inaccuracy of the model itself which did not take into account the fact that no operation can increase the confidence of any proof. Our planned solution is to develop a learning method in which only positive values can be learned for the feature weights.

Another future goal is to improve the search algorithm. A good search algorithm should fulfill two properties. First - *accuracy*: the proofs found by the algorithm should have as low cost as possible. Second - *run-time*: the required run-time to find the (approximated) best proof should be as short as possible.

Another future improvement is using additional sentence analysis: Currently, we make syntactic analysis as well as named-entity-recognition and coreference resolution for the input data. However, many other analyses could be performed, for example: semantic-role labeling, temporal expression detection, numerical expression detection, etc. Such information can be utilized as features for on-the-fly operations, in a similar way to our current use of part-of-speech and named-entity-recognition.

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. This work was developed under the ITCH collaboration project with FBK-irst/University of Haifa.

References

- Marilisa Amoia and Claire Gardent. 2008. A test suite for inference involving adjectives. In *Proceedings of LREC*.
- 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 Szepktor, 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, Jonathan Berant, Ido Dagan, Iddo Greental, Shachar Mirkin, Eyal Shnarch, and Idan Szepktor.

2008. Efficient semantic deduction and approximate matching over compact parse forests. In *Proceedings of TAC*.
- Roy Bar-Haim, Jonathan Berant, and Ido Dagan. 2009. A compact forest for scalable inference over entailment and paraphrase rules. In *Proceedings of EMNLP*.
- Roy Bar-Haim. 2010. *Semantic Inference at the Lexical-Syntactic Level*. Ph.D. thesis, Bar-Ilan University.
- Roni Ben-Aharon, Idan Szpektor, and Ido Dagan. 2010. Generating entailment rules from framenet. In *Proceedings of ACL*.
- Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of EMNLP*.
- 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*.
- Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of NAACL*.
- Aria Haghighi and Dan Klein. 2009. Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of EMNLP*.
- Stefan Harmeling. 2009. Inferring textual entailment with a probabilistically sound calculus. *Natural Language Engineering*.
- Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*.
- Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL*.
- T. Joachims. 2005. A support vector method for multivariate performance measures. In *ICML*.
- Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*.
- 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. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*.
- Yashar Mehdad. 2009. Automatic cost estimation for tree edit distance using particle swarm optimization. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*.
- George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*.
- Shachar Mirkin, Roy Bar-Haim, Jonathan Berant, Ido Dagan, Eyal Shnarch, Asher Stern, and Idan Szpektor. 2009. Addressing discourse and document structure in the rte search task. In *Proceedings of TAC*.
- 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*.
- Asher Stern and Ido Dagan. 2011. A confidence model for syntactically-motivated entailment proofs. In *Proceedings of RANLP*.