BIUTEE: A Modular Open-Source System for Recognizing Textual Entailment

Asher Stern
Computer Science Department
Bar-Ilan University
Ramat-Gan 52900, Israel
astern7@gmail.com

Ido Dagan
Computer Science Department
Bar-Ilan University
Ramat-Gan 52900, Israel
dagan@cs.biu.ac.il

Abstract

This paper introduces BIUTEE[1], an open-source system for recognizing textual entailment. Its main advantages are its ability to utilize various types of knowledge resources, and its extensibility by which new knowledge resources and inference components can be easily integrated. These abilities make BIUTEE an appealing RTE system for two research communities: (1) researchers of end applications, that can benefit from generic textual inference, and (2) RTE researchers, who can integrate their novel algorithms and knowledge resources into our system, saving the time and effort of developing a complete RTE system from scratch. Notable assistance for these researchers is provided by a visual tracing tool, by which researchers can refine and "debug" their knowledge resources and inference components.

1 Introduction

Recognizing Textual Entailment (RTE) is the task of identifying, given two text fragments, whether one of them can be inferred from the other (Dagan et al., 2006). This task generalizes a common problem that arises in many tasks at the semantic level of NLP. For example, in *Information Extraction (IE)*, a system may be given a template with variables (e.g., "X is employed by Y") and has to find text fragments from which this template, with variables replaced by proper entities, can be inferred. In *Summarization*, a good summary should be inferred from the given text, and, in addition, should not contain duplicated information, i.e., sentences which can be inferred from other sentences in the summary. Detecting these inferences can be performed by an RTE system.

Since first introduced, several approaches have been proposed for this task, ranging from shallow lexical similarity methods (e.g., (Clark and Harrison, 2010; MacKinlay and Baldwin, 2009)), to complex linguistically-motivated methods, which incorporate extensive linguistic analysis (syntactic parsing, coreference resolution, semantic role labelling, etc.) and a rich inventory of linguistic and world-knowledge resources (e.g., (Iftene, 2008; de Salvo Braz et al., 2005; Bar-Haim et al., 2007)). Building such complex systems requires substantial development efforts, which might become a barrier for new-comers to RTE research. Thus, flexible and extensible publicly available RTE systems are expected to significantly facilitate research in this field. More concretely, two major research communities would benefit from a publicly available RTE system:

1. Higher-level application developers, who would use an RTE system to solve inference tasks in their application. RTE systems for this type of researchers should be adaptable for the application specific data: they should be configurable, trainable, and extensible with inference knowledge that captures application-specific phenomena.

2. Researchers in the RTE community, that would not need to build a complete RTE system for their research. Rather, they may integrate

[1] www.cs.biu.ac.il/~nlp/downloads/biutee

their novel research components into an existing open-source system. Such research efforts might include developing knowledge resources, developing inference components for specific phenomena such as temporal inference, or extending RTE to different languages. A flexible and extensible RTE system is expected to encourage researchers to create and share their textual-inference components. A good example from another research area is the *Moses* system for *Statistical Machine Translation (SMT)* (Koehn et al., 2007), which provides the core SMT components while being extended with new research components by a large scientific community.

Yet, until now rather few and quite limited RTE systems were made publicly available. Moreover, these systems are restricted in the types of knowledge resources which they can utilize, and in the scope of their inference algorithms. For example, *EDITS*[2] (Kouylekov and Negri, 2010) is a distance-based RTE system, which can exploit only lexical knowledge resources. *NutCracker*[3] (Bos and Markert, 2005) is a system based on logical representation and automatic theorem proving, but utilizes only WordNet (Fellbaum, 1998) as a lexical knowledge resource.

Therefore, we provide our open-source textual-entailment system, BIUTEE. Our system provides state-of-the-art linguistic analysis tools and exploits various types of manually built and automatically acquired knowledge resources, including lexical, lexical-syntactic and syntactic rewrite rules. Furthermore, the system components, including pre-processing utilities, knowledge resources, and even the steps of the inference algorithm, are modular, and can be replaced or extended easily with new components. Extensibility and flexibility are also supported by a *plug-in mechanism*, by which new inference components can be integrated without changing existing code.

Notable support for researchers is provided by *a visual tracing tool*, *Tracer*, which visualizes every step of the inference process as shown in Figures 2 and 3. We will use this tool to illustrate various inference components in the demonstration session.

[2]`http://edits.fbk.eu/`

[3]`http://svn.ask.it.usyd.edu.au/trac/candc/wiki/nutcracker`

2 System Description

2.1 Inference algorithm

In this section we provide a high level description of the inference components. Further details of the algorithmic components appear in references provided throughout this section.

BIUTEE follows the transformation based paradigm, which recognizes textual entailment by converting the text into the hypothesis via a sequence of transformations. Such a sequence is often referred to as a *proof*, and is performed, in our system, over the syntactic representation of the text - the text's parse tree(s). A transformation modifies a given parse tree, resulting in a generation of a new parse tree, which can be further modified by subsequent transformations.

Consider, for example, the following text-hypothesis pair:

Text: ... Obasanjo invited him to step down as president ... and accept political asylum in Nigeria.

Hypothesis: Charles G. Taylor was offered asylum in Nigeria.

This text-hypothesis pair requires two major transformations: (1) substituting "him" by "Charles G. Taylor" via a coreference substitution to an earlier mention in the text, and (2) inferring that if "X accept Y" then "X was offered Y".

BIUTEE allows many types of transformations, by which any hypothesis can be proven from any text. Given a T-H pair, the system finds a proof which generates H from T, and estimates the proof validity. The system returns a score which indicates how likely it is that the obtained proof is valid, i.e., the transformations along the proof preserve entailment from the meaning of T.

The main type of transformations is application of *entailment-rules* (Bar-Haim et al., 2007). An entailment rule is composed of two sub-trees, termed *left-hand-side* and *right-hand-side*, and is *applied* on a parse-tree fragment that matches its left-hand-side, by substituting the left-hand-side with the right-hand-side. This formalism is simple yet powerful, and captures many types of knowledge. The simplest type of rules is *lexical rules*, like `car` $\rightarrow$

`vehicle`. More complicated rules capture the entailment relation between predicate-argument structures, like `X accept Y` $\rightarrow$ `X was offered Y`. Entailment rules can also encode syntactic phenomena like the semantic equivalence of active and passive structures (`X Verb[active] Y` $\rightarrow$ `Y is Verb[passive] by X`). Various knowledge resources, represented as entailment rules, are freely available in BIUTEE's web-site. The complete formalism of entailment rules, adopted by our system, is described in (Bar-Haim et al., 2007).

Coreference relations are utilized via coreference-substitution transformations: one mention of an entity is replaced by another mention of the same entity, based on coreference relations. In the above example the system could apply such a transformation to substitute "him" with "Charles G. Taylor".

Since applications of entailment rules and coreference substitutions are yet, in most cases, insufficient in transforming T into H, our system allows *on-the-fly* transformations. These transformations include insertions of missing nodes, flipping parts-of-speech, moving sub-trees, etc. (see (Stern and Dagan, 2011) for a complete list of these transformations). Since these transformations are not justified by given knowledge resources, we use linguistically-motivated features to estimate their validity. For example, for on-the-fly lexical insertions we consider as features the named-entity annotation of the inserted word, and its probability estimation according to a unigram language model, which yields lower costs for more frequent words.

Given a (T,H) pair, the system applies a search algorithm (Stern et al., 2012) to find a proof $O = (o_1, o_2, \ldots o_n)$ that transforms T into H. For each proof step o_i the system calculates a *cost* $c(o_i)$. This cost is defined as follows: the system uses a weight-vector w, which is learned in the training phase. In addition, each transformation o_i is represented by a *feature vector* $f(o_i)$ which characterizes the transformation. The cost $c(o_i)$ is defined as $w \cdot f(o_i)$. The *proof cost* is defined as the sum of the costs of the transformations from which it is composed, i.e.:

$$c(O) \triangleq \sum_{i=1}^{n} c(o_i) = \sum_{i=1}^{n} w \cdot f(o_i) = w \cdot \sum_{i=1}^{n} f(o_i) \quad (1)$$

If the proof cost is below a threshold b, then the system concludes that T entails H. The complete description of the cost model, as well as the method for learning the parameters w and b is described in (Stern and Dagan, 2011).

2.2 System flow

The BIUTEE system flow (Figure 1) starts with pre-processing of the text and the hypothesis. BIUTEE provides state-of-the-art pre-processing utilities: *Easy-First parser* (Goldberg and Elhadad, 2010), *Stanford named-entity-recognizer* (Finkel et al., 2005) and *ArkRef coreference resolver* (Haghighi and Klein, 2009), as well as utilities for sentence-splitting and numerical-normalizations. In addition, BIUTEE supports integration of users' own utilities by simply implementing the appropriate interfaces. Entailment recognition begins with a *global processing phase* in which inference related computations that are not part of the proof are performed. Annotating the negation indicators and their scope in the text and hypothesis is an example of such calculation. Next, the system constructs a proof which is a sequence of transformations that transform the text into the hypothesis. Finding such a proof is a sequential process, conducted by the search algorithm. In each step of the proof construction the system examines all possible transformations that can be applied, generates new trees by applying selected transformations, and calculates their costs by constructing appropriate feature-vectors for them.

New types of transformations can be added to BIUTEE by a *plug-in* mechanism, without the need to change the code. For example, imagine that a researcher applies BIUTEE on the medical domain. There might be some well-known domain knowledge and rules that every medical person knows. Integrating them is directly supported by the plug-in mechanism. A plug-in is a piece of code which implements a few interfaces that detect which transformations can be applied, apply them, and construct appropriate feature-vectors for each applied transformation. In addition, a plug-in can perform computations for the global processing phase.

Eventually, the search algorithm finds a (approximately) lowest cost proof. This cost is normalized as a score between 0 and 1, and returned as output.

Training the cost model parameters w and b (see subsection 2.1) is performed by a linear learn-

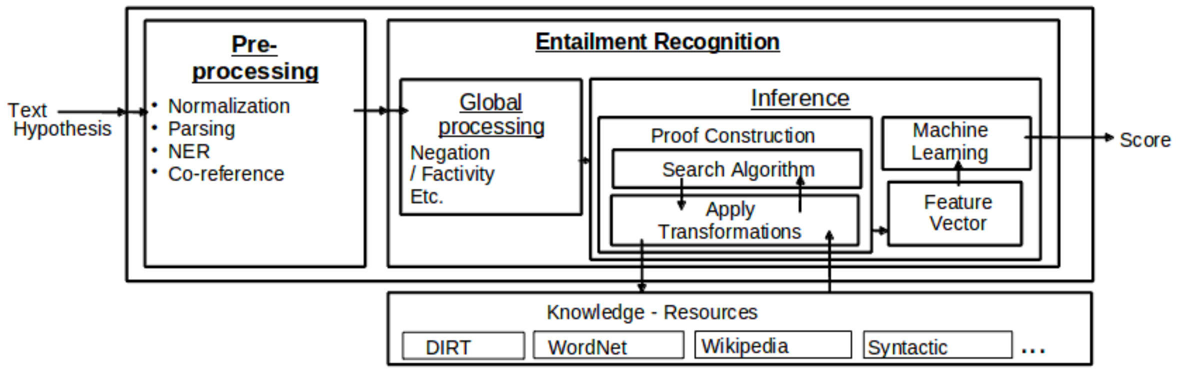


Figure 1: System architecture

RTE challenge	Median	Best	BIUTEE
RTE-6	33.72	48.01	49.09
RTE-7	39.89	48.00	42.93

Table 1: Performance (F1) of BIUTEE on RTE challenges, compared to other systems participated in these challenges. *Median* and *Best* indicate the median score and the highest score of all submissions, respectively.

ing algorithm, as described in (Stern and Dagan, 2011). We use a *Logistic-Regression* learning algorithm, but, similar to other components, alternative learning-algorithms can be integrated easily by implementing an appropriate interface.

2.3 Experimental results

BIUTEE's performance on the last two RTE challenges (Bentivogli et al., 2011; Bentivogli et al., 2010) is presented in Table 1: BIUTEE is better than the median of all submitted results, and in RTE-6 it outperforms all other systems.

3 Visual Tracing Tool

As a complex system, the final score provided as output, as well as the system's detailed logging information, do not expose all the decisions and calculations performed by the system. In particular, they do not show all the potential transformations that could have been applied, but were rejected by the search algorithm. However, such information is crucial for researchers, who need to observe the usage and the potential impact of each component of the system.

We address this need by providing an interactive *visual tracing tool*, *Tracer*, which presents detailed information on each proof step, including potential steps that were not included in the final proof. In the demo session, we will use the visual tracing tool to illustrate all of BIUTEE's components[4].

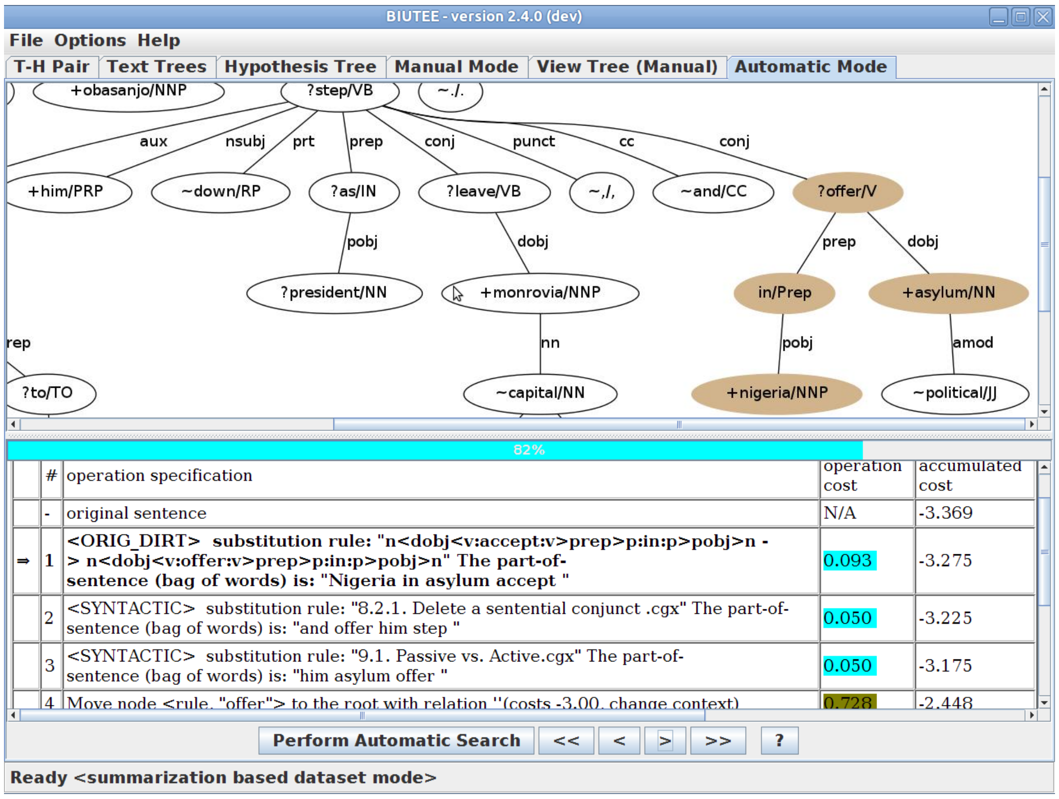
3.1 Modes

Tracer provides two modes for tracing proof construction: *automatic mode* and *manual mode*. In automatic mode, shown in Figure 2, the tool presents the complete process of inference, as conducted by the system's search: the parse trees, the proof steps, the cost of each step and the final score. For each transformation the tool presents the parse tree before and after applying the transformation, highlighting the impact of this transformation. In manual mode, the user can invoke specific transformations proactively, including transformations rejected by the search algorithm for the eventual proof. As shown in Figure 3, the tool provides a list of transformations that match the given parse-tree, from which the user chooses and applies a single transformation at each step. Similar to automatic mode, their impact on the parse tree is shown visually.

3.2 Use cases

Developers of knowledge resources, as well as other types of transformations, can be aided by *Tracer* as follows. Applying an entailment rule is a process of first *matching* the rule's left-hand-side to the text parse-tree (or to any tree along the proof), and then substituting it by the rule's right-hand-side. To test a

[4]Our demonstration requirements are a large screen and Internet connection.



	#	operation specification	operation cost	accumulated cost
	-	original sentence	N/A	-3.369
⇒	1	**<ORIG_DIRT> substitution rule: "n<dobj<v:accept:v>prep>p:in:p>pobj>n -> n<dobj<v:offer:v>prep>p:in:p>pobj>n" The part-of-sentence (bag of words) is: "Nigeria in asylum accept "**	0.093	-3.275
	2	<SYNTACTIC> substitution rule: "8.2.1. Delete a sentential conjunct .cgx" The part-of-sentence (bag of words) is: "and offer him step "	0.050	-3.225
	3	<SYNTACTIC> substitution rule: "9.1. Passive vs. Active.cgx" The part-of-sentence (bag of words) is: "him asylum offer "	0.050	-3.175
	4	Move node <rule, "offer"> to the root with relation ''(costs -3.00, change context)	0.728	-2.448

Ready <summarization based dataset mode>

Figure 2: Entailment Rule application visualized in tracing tool. The upper pane displays the parse-tree generated by applying the rule. The rule description is the first transformation (printed in **bold**) of the proof, shown in the lower pane. It is followed by transformations 2 and 3, which are syntactic rewrite rules.

rule, the user can provide a text for which it is supposed to match, examine the list of potential transformations that can be performed on the text's parse tree, as in Figure 3, and verify that the examined rule has been matched as expected. Next, the user can apply the rule, visually examine its impact on the parse-tree, as in Figure 2, and validate that it operates as intended with no side-effects.

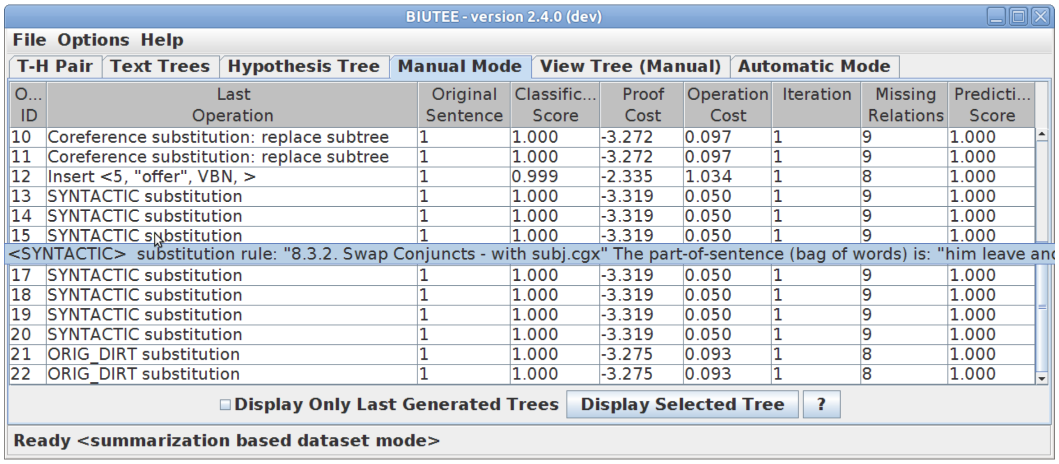
The complete inference process depends on the parameters learned in the training phase, as well as on the search algorithm which looks for lowest-cost proof from T to H. Researchers investigating these algorithmic components can be assisted by the tracing tool as well. For a given (T,H) pair, the automatic mode provides the complete proof found by the system. Then, in the manual mode the researcher can try to construct alternative proofs. If a proof with lower cost can be constructed manually it implies a limitation of the search algorithm. On the other hand, if the user can manually construct a better linguistically motivated proof, but it turns out that this proof has higher cost than the one found by the system, it implies a limitation of the learning phase which may be caused either by a limitation of the learning method, or due to insufficient training data.

4 Conclusions

In this paper we described BIUTEE, an open-source textual-inference system, and suggested it as a research platform in this field. We highlighted key advantages of BIUTEE, which directly support researchers' work: (a) modularity and extensibility, (b) a plug-in mechanism, (c) utilization of entailment rules, which can capture diverse types of knowledge, and (d) a visual tracing tool, which visualizes all the details of the inference process.

Acknowledgments

This work was partially supported by the Israel Science Foundation grant 1112/08, the PASCAL-



O... ID	Last Operation	Original Sentence	Classific... Score	Proof Cost	Operation Cost	Iteration	Missing Relations	Predicti... Score
10	Coreference substitution: replace subtree	1	1.000	-3.272	0.097	1	9	1.000
11	Coreference substitution: replace subtree	1	1.000	-3.272	0.097	1	9	1.000
12	Insert <5, "offer", VBN, >	1	0.999	-2.335	1.034	1	8	1.000
13	SYNTACTIC substitution	1	1.000	-3.319	0.050	1	9	1.000
14	SYNTACTIC substitution	1	1.000	-3.319	0.050	1	9	1.000
15	SYNTACTIC substitution	1	1.000	-3.319	0.050	1	9	1.000
17	SYNTACTIC substitution	1	1.000	-3.319	0.050	1	9	1.000
18	SYNTACTIC substitution	1	1.000	-3.319	0.050	1	9	1.000
19	SYNTACTIC substitution	1	1.000	-3.319	0.050	1	9	1.000
20	SYNTACTIC substitution	1	1.000	-3.319	0.050	1	9	1.000
21	ORIG_DIRT substitution	1	1.000	-3.275	0.093	1	8	1.000
22	ORIG_DIRT substitution	1	1.000	-3.275	0.093	1	8	1.000

<SYNTACTIC> substitution rule: "8.3.2. Swap Conjuncts - with subj.cgx" The part-of-sentence (bag of words) is: "him leave an

Ready <summarization based dataset mode>

Figure 3: List of available transformations, provided by *Tracer* in the manual mode. The user can manually choose and apply each of these transformations, and observe their impact on the parse-tree.

2 Network of Excellence of the European Community FP7-ICT-2007-1-216886, and the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 287923 (EXCITEMENT).

References

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

Luisa Bentivogli, Peter Clark, Ido Dagan, Hoa Dang, and Danilo Giampiccolo. 2010. The sixth pascal recognizing textual entailment challenge. In *Proceedings of TAC*.

Luisa Bentivogli, Peter Clark, Ido Dagan, Hoa Dang, and Danilo Giampiccolo. 2011. The seventh pascal recognizing textual entailment challenge. In *Proceedings of TAC*.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of EMNLP*.

Peter Clark and Phil Harrison. 2010. Blue-lite: a knowledge-based lexical entailment system for rte6. In *Proceedings of TAC*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Quionero-Candela, J.; Dagan, I.; Magnini, B.; d'Alch-Buc, F. (Eds.) Machine Learning Challenges. Lecture Notes in Computer Science*.

Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. 2005. An inference model for semantic entailment in natural language. In *Proceedings of AAAI*.

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

Jenny Rose Finkel, Trond Grenager, and Christopher 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*.

Adrian Iftene. 2008. Uaic participation at rte4. In *Proceedings of TAC*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL*.

Milen Kouylekov and Matteo Negri. 2010. An open-source package for recognizing textual entailment. In *Proceedings of ACL Demo*.

Andrew MacKinlay and Timothy Baldwin. 2009. A baseline approach to the rte5 search pilot. In *Proceedings of TAC*.

Asher Stern and Ido Dagan. 2011. A confidence model for syntactically-motivated entailment proofs. In *Proceedings of RANLP*.

Asher Stern, Roni Stern, Ido Dagan, and Ariel Felner. 2012. Efficient search for transformation-based inference. In *Proceedings of ACL*.