Deeper Natural Language Processing for Evaluating Student Answers in Intelligent Tutoring Systems

Vasile Rus* **and Arthur C. Graesser****
Institute for Intelligent Systems
Department of Computer Science*
Department of Psychology**
373 Dunn Hall
Memphis, Tennessee 38152
vrus, a-graesser}@memphis.edu

Abstract

This paper addresses the problem of evaluating students' answers in intelligent tutoring environments with mixed-initiative dialogue by modelling it as a textual entailment problem. The problem of meaning representation and inference is a pervasive challenge in any integrated intelligent system handling communication. For intelligent tutorial dialogue systems, we show that entailment cases can be detected at various dialog turns during a tutoring session. We report the performance of a lexico-syntactic approach on a set of entailment cases that were collected from a previous study we conducted with AutoTutor.

Introduction

One fundamental problem in all Intelligent Tutoring Systems (ITS) with natural language interaction is to evaluate students' answers. During a dialogue that takes place between the system and the student, the system checks whether the student can articulate each of a set of ideal steps (called expectations) of the ideal answer to a given problem. Let us consider the following expectation and student answer, reproduced from a previous experiment from AutoTutor (Graesser, Hu, & McNamara 2005).

E:*The person and the object cover the same horizontal distance.*
A:*The two objects will cover the same distance.*

The challenge is to decide whether the expectation (E) can entail, or logically infer, the student answer A. This is similar to the task of Recognizing Textual Entailment (RTE) that was recently proposed (Dagan, Glickman, & Magnini 2005). RTE is the task of deciding, given two text fragments, whether the meaning of one text is entailed (can be inferred) from another text. We say that T (the entailing text) entails H (the entailed hypothesis). The task is relevant to a large number of applications, including machine translation, question answering, and information retrieval. However, as we show here, entailment cases are present in all ITS environments with tutorial dialogue in natural language.

In this paper we identify entailment cases in one ITS with dialogue in natural language, namely AutoTutor (Graesser *et al.* 2001), and discuss the peculiarities of applying entailment solutions to this class of tutoring systems. AutoTutor is an ITS that works by having a conversation with the learner. Students are encouraged to articulate lengthy answers that exhibit deep reasoning, rather than to recite small bits of shallow knowledge. It is in the deep knowledge and reasoning part where we explore the potential help of entailment.

Further, a novel approach to assess the correctness of students answers in intelligent tutoring systems is presented. The approach uses minimal knowledge resources. It relies on lexico-syntactic information and synonymy embedded in a thesaurus, an online dictionary. The minimal use of resources leads to fast entailment decisions which is important in intelligent tutoring systems where interactivity is crucial. A light entailer is absolutely necessary in interactive environments in general. This work provides an answer to the question how far one can go with a light system that uses a minimal set of resources.

In our approach each T-H pair is first mapped into two graphs, one for H and one for T. The nodes represent main concepts and the links indicate syntactic dependencies among concepts as encoded in H and T, respectively. An entailment score, $entail(T, H)$, is then computed quantifying the degree to which the T-graph subsumes the H-graph. The score is defined to be non-reflexive, i.e. $entail(T, H) \neq entail(H, T)$. We evaluated the approach on a set of data collected from AutoTutor sessions with students. The data exhibits a very different distribution of TRUE-FALSE cases from the 50-50 split used in the RTE challenge (Dagan, Glickman, & Magnini 2005). The results obtained are promising and at the same time challenging. According to our experiments, a method that embeds syntax leads to best precision scores. More details are offered throughout the paper and in particular in the *Conclusions* section.

The rest of the paper is structured as follows. *Related Work* outlines previous work in ITS and entailment. The third section makes the case for entailment in ITS whereas the fourth section describes our lexico-syntactic approach. The fifth section explains the scoring formula and its components and is followed by the section that presents the experiments we performed, the results, and a comparative view of different approaches. *Conclusions* ends the paper.

Related Work

AutoTutor is a computer tutor that holds conversations with students in natural language (Graesser *et al.* 2001)(Graesser, Hu, & McNamara 2005). AutoTutor simulates the discourse patterns of human tutors and a number of ideal tutoring strategies. It presents a series of challenging problems (or questions) from a curriculum script and engages in collaborative, mixed initiative dialog while constructing answers. So far, AutoTutor has been developed and tested for topics in Newtonian physics (VanLehn, Graesser et al., 2004) and computer literacy (Graesser, Lu et al., 2004), showing impressive learning gains compared to pretest measures and suitable control conditions. We use in this paper a subset of the physics domain to explore entailment in ITS. Namely, we analyzed a corpus of 125 pairs of expectation-student answer (called E-S pairs) from sessions between students and AutoTutor. In most of the existing applications of AutoTutor, Latent Semantic Analysis (LSA) (Landauer, Foltz, & Laham 1998) has been used to evaluate student contributions during the dialog between the student and AutoTutor. A couple of problems with LSA is that it does not encode word order and it ignores negation. Our lexico-syntactic approach tries to overcome this drawback by taking into account syntactic information. We also consider negation handling although in the experiments presented here we did not emphasized the challenges of negation.

The task of textual entailment was recently proposed (Dagan, Glickman, & Magnini 2005) to be studied in a standardized framework but work on the problem can be found even before the advent of RTE.

In one of the earliest explicit treatments of entailment (Monz & de Rijke 2001) proposed a weighted bag of words approach to entailment. They argued that traditional inference systems based on first order logic are restricted to yes/no decisions when it comes to entailment tasks. In contrast, their approach delivered 'graded outcomes'. They established entailment relations among larger pieces of text (i.e. on average segments of 4 sentences) than the proposed RTE setup where the text size is a sentence (seldom two) or part of a sentence (phrase).

A closely related effort is presented in (Moldovan & Rus 2001). They show how to use unification and matching to address the answer correctness problem. Answer correctness can be viewed as entailment: Is a candidate answer entailing the ideal answer to the question? Initially, the question is paired with an answer from a list of candidate answers (obtained through some keyword proximity and shallow semantics methods). The resulting pair is mapped into a first-order logic representation and an unification process between the question and the answer follows. As a back-off step, for the case when no full unification is possible, the answer with highest unification score is ranked at the top. The task they describe is different than the RTE task because a list of candidate answers to rank are available. The granularity of candidate answers and questions is similar to the RTE data.

Pazienza and colleagues (Pazienza, Pennacchiotti, & Zanzotto 2005) use syntactic graph distance approach for the task of textual entailment. Their approach is closest to ours. Among the drawbacks of their work, as compared to ours, are the lack of negation handling and of surface-level representation of syntactic relationships. (Pazienza, Pennacchiotti, & Zanzotto 2005) also treat the problem as a graph-edit problem, which depends heavily on the type of edit commands one defines, while our approach focuses on simple node and edge matching.

The Case for Entailment in Intelligent Tutoring Systems

We show in this section that entailment cases can be found in Intelligent Tutoring Systems. The goal of the paper is to identify such cases and then use approaches to entailment to address them. The cases we identify are when the ITS needs to evaluate a student answer against an ideal answer, i.e. expectation. Let us look at how AutoTutor operates so that we can better understand where entailment cases occur.

During a session with AutoTutor the student is challenged with solving a problem. It typically takes 50-200 conversational turns between the tutor and student to provide an adequate answer. This is approximately the same number of turns it takes between human tutors and students who collaboratively answer similar problems that involve deep reasoning.

AutoTutor's curriculum script contains a set of problems about a subject matter that requires deep reasoning. Various forms of content are affiliated with each problem of which the most important are: the main question or the problem, expectations (ideal answer), misconceptions. The ideal answer can be viewed as the goal and expectations and misconceptions as subgoals. To achieve the goal of obtaining the ideal answer from students AutoTutor's strategy is to reach the component subgoals. AutoTutor requires that the learner articulate each of the expectations before it considers the question answered. The system periodically identifies a missing expectation during the course of the dialogue and posts the goal of covering the expectation. When expectation E is missed and therefore posted, AutoTutor attempts to get the student to articulate it by generating hints and prompts to encourage the learner's filling in missing words and propositions. Student answers have been evaluated against the expectation using LSA.

We show here that evaluating student answers is analogous to an entailment problem. To illustrate this analogy let us pick the *Pumpkin* problem from AutoTutor library: *Suppose a runner is running in a straight line at constant speed, and the runner throws a pumpkin straight up. Where will the pumpkin land? Explain why.* Table 1 shows E-S pairs for the problem. The last column is the rating on a scale from 1 to 4, with 1 meaning very bad and 4 very good, of the student answer with regard to the expectation. The ratings were assigned by Physics experts holding PhD degrees or by physics graduate students. Let us now look at the analogy on this particular problem. An E-S pair can be regarded as an entailment pair of Text-Hypothesis. The expectation is what we know is true, which is similar to the Text component of a Text-Hypothesis pair in RTE. The student answer contribution is a hypothetical statement which can be true or

Expectation	Student Answer	Expert Rating
The person and the object cover the same horizontal distance.	The two objects will cover the same distance.	4
The person and the object cover the same horizontal distance.	The horizontal displacements must be different because the runner continues to run and the pumpkin only has a vertical component to its motion.	1
The object will continue to move at the same horizontal velocity as the person when it is thrown.	The pumpkin and the runner have the same horizontal velocity before and after release.	4
The object will continue to move at the same horizontal velocity as the person when it is thrown.	The pumpkin's horizontal velocity will decrease after it is released.	1

Table 1: Example pairs of expectation-student answer from the *Pumpkin Problem* in AutoTutor.

false. The task is to find the truth value of the student answer based on the true fact encoded in the expectation (and background knowledge, which is required when evaluating the physics content). Given this analogy, we examined how it performs on a test set of 125 E-S pairs collected from a sample of AutoTutor tutorial dialogues.

Approach To Textual Entailment

The task of entailment requires an arsenal of language processing components. For a detailed discussion see (Rus, Graesser, & Desai 2005). In this paper we study the impact of a lexico-syntactic graph-based approach on evaluation of E-S pairs. The graph-based approach is based on the notion of subsumption. In general, an object X subsumes an object Y if X is more general than or identical to Y. Or, alternatively we say Y is more specific than X. The same idea applies to more complex objects and structures of interrelated objects. When applied to textual entailment, subsumption translates into the following: hypothesis H is entailed from T if and only if T subsumes H.

The solution has two phases: (I) map both T and H into graph structures and (II) perform a subsumption operation between the T-graph and H-graph.

Phase I: From Text To Graph Representations

The two text fragments involved in a textual entailment decision are initially mapped into a graph representation that has its roots in the dependency-graph formalisms of (Mel'cuk 1998). The mapping process has three phases: preprocessing, dependency graph generation and final graph generation.

In the preprocessing phase we perform tokenization (separation of punctuation from words), lemmatization (map morphological variations of words to their base or root form), part-of-speech tagging (assign parts of speech to each word) and parsing (discover major phrases and how they relate to each other, with the phrases being grouped into a parse tree). The preprocessing continues with a step in which parse trees are transformed in a way that helps the graph generation process in the next phase. For example, auxiliaries and passive voice are eliminated but their important information is kept: voices are marked as additional labels to the tag that identifies the verb; verb aspect information for the verb a modal (may, must, can) acts upon is recorded as an extra marker of the node in the graph that is generated for the verb. An important step, part of the preprocessing phase, identifies major concepts in the input: named entities, compound nouns and collocations, postmodifiers, existentials, etc. This step is important because, for instance, entities may appear as composed of multiple words in T, e.g. *Overture Services Inc*), and as a single word concept in H, e.g. (*Overture*). To assure a proper treatment of those cases only common collocations, namely those composed of a sequence of common nouns in the input, are represented as a single concept by replacing the consecutive words forming a collocation with a new concept composed of the individual words glued with an underscore. A dictionary of collocations (compiled from WordNet (Miller 1995)) and a simple algorithm help us detect collocations in the input.

The actual mapping from text to the graph representation is based on information from parse trees. A parse tree groups words in a sentence into phrases and organizes phrases in hierarchical tree structures from where we can easily detect syntactic dependencies among concepts. We use Charniak's (Charniak 2000) parser to obtain parse trees and head-detection rules (Magerman 1994) to obtain the head of each phrase. A dependency tree is generated by linking the head of each phrase to its modifiers in a straightforward mapping step. The problem with the dependency tree is that it only encodes local dependencies (head-modifiers). Remote dependencies are not marked in such dependency trees. An extra step transforms the previous dependency tree into a dependency graph (Figure 1) in which remote dependencies are explicitly marked and further into a final graph in which direct relations among content words are coded. For instance, a *mod* dependency between a noun and its attached preposition is replaced by a direct dependency between the prepositional head and prepositional object.

The remote dependencies are obtained using a naive-Bayes functional tagger. The naive Bayesian model relies on more than a dozen linguistic features automatically extracted from parse trees (phrase label, head, part of speech, parent's head, parent's label, etc.). The model was trained on annotated data from Wall Street Journal section of Penn Treebank (Marcus, Santorini, & Marcinkiewicz 1993). The

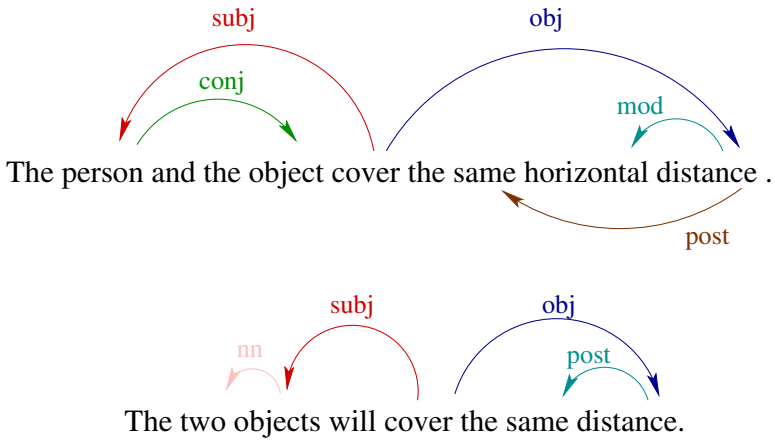

Figure 1: Example of graph representation for an Expectation (top) - Student Answer (bottom) pair.(Edges are colour-coded to better visualize the correspondence.)

accuracy of the functional tagger is in the 90-th percentile (Rus & Desai 2005).

As soon as graph representations are obtained, a graph matching operation is initialized. The operation is detailed in the next section.

Phase II: Graph Subsumption

Let us remember core concepts from graph theory before we proceed with modelling subsumption for textual entailment.

A graph $G = (V, E)$ consists of a set of nodes or vertices V and a set of edges E. Isomorphism in graph theory is the problem of testing whether two graphs are really the same (Skiena 1998). Several variations of graph isomorphism exist in practice of which the subsumption or containment problem best fits our task. Is graph H contained in (not identical to) graph T? Graph subsumption consists of finding a mapping from vertices in H to T such as edges among nodes in H hold among mapped edges in T. In our case the problem can be further relaxed: attempt a subsumption and if that is not possible back-off to a partial subsumption. The important aspect is to quantify the degree of subsumption of H by T.

The subsumption algorithm for textual entailment has three major steps: (1) find an isomorphism between H_v (set of vertices of the Hypothesis graph) and T_v (2) check whether the labelled edges in H, E_H, have correspondents in E_T (3) compute score. Step 1 is more than a simple word-matching method since if a node in H does not have a direct correspondent in T a thesaurus is used to find all possible synonyms for nodes in T. Nodes in H have different priorities: head words are most important followed by modifiers. Modifiers that indicate negation are handled separately from the bare lexico-syntactic subsumption since. If H is subsumed at large by T, and T is not negated but H is or viceversa (see example in Figure), the overall score should be dropped, with high confidence, to indicate no entailment. Step 2 takes each relation in H and checks its presence in T. It is augmented with relation equivalences among appositions, possessives and linking verbs (*be, have*). Lastly, a normalized score for node and edge mapping is computed. The score for the entire entailment is the sum of each individual node and relation matching score. The node match consists of lexical matching and aspect matching (for verbs). The overall score is sanctioned by negation relations. More details on scoring are found in *The Scoring* Section.

Negation

Negation is important for our approach because we hope our system yields an improvement over LSA. Consider the ideal case of a pair for which all nodes in H and their relations can be perfectly mapped in T. According to our approach, one would decide, with maximum confidence, that T entails H. The decision is wrong simply because T is negated and H is not. To handle such cases negation treatment is necessary and this section declares one solution.

We pay special attention to two broad types of negation: explicit and implicit. Explicit negation is indicated by particles such as: *no, not, neither ... nor* and their shortened forms *'nt*. Implicit negation is present in text via deeper lexico-semantic relations among different linguistic expressions, the most obvious example is the *antonymy* relation among lemmas which can be retrieved from WordNet. Negation is regarded as a feature of both text and hypothesis; it is handled in the score after the entailment decision for the Text-Hypothesis pair without negation is made. If one of the text fragments is negated, the decision is reversed, but if both are negated the decision is retained (double-negation), and so forth. In Equation 1 the term *#neg_rel* represents the number of negation relations between T and H.

The Scoring

Our formula to obtain an overall score aims to deliver both a numerical value for the degree of entailment between T and H and a degree of confidence in the decision. The score can have values between 0 and 1, with 1 meaning TRUE entailment with maximum confidence and 0 meaning FALSE

$$entscore(T,H) = (\alpha \times \frac{\sum_{V_h \in H_v} max_{V_t \in T_v} match(V_h, V_t)}{|V_h|} + \beta \times \frac{\sum_{E_h \in H_e} max_{E_t \in T_v} synt\_match(E_h, E_t)}{|E_h|} + \gamma) \times \frac{(1 + (-1)^{\#neg\_rel})}{2} \quad (1)$$

entailment with maximum confidence.

The formula to compute the overall score is provided by Equation 1. There are three important components of the score: lexical or node matching, syntactic or relational matching, and negation. The weights of lexical and syntactic matching are given by parameters α and β, respectively. The effect of negation on entailment decision is captured by the last term of the equation. An odd number of negation relations between T and H, denoted *#neg_rel*, would lead to an entailment score of 0 while an even number will not change the bare lexico-semantic score. The choice of α, β and γ can have a great impact on the overall score. The *Experiments and Results* section discusses how to estimate those parameters. From the way the score is defined it is obvious that $entscore(H,T) \neq entscore(T,H)$.

Experimental Setup and Results

We modelled our experiments after the experimental setup used by the RTE Challenge (Dagan, Glickman, & Magnini 2005). We collected a set of 125 E-S pairs and the correct entailment judgments, TRUE or FALSE, as assigned by human experts. The pairs were then fed into our system which produces a result in the form of a judgment, TRUE or FALSE, and a confidence score on the judgement. The system's judgment was automatically compared with the correct judgment. Several evaluation metrics are reported as proposed in (Dagan, Glickman, & Magnini 2005). The percentage of matching judgements provides the *accuracy* of the run, i.e. the fraction of correct responses.

As a second measure, a *Confidence-Weighted Score* (CWS, also known as average precision) is computed using the formula below. Judgements of the test examples are sorted by their confidence in decreasing order from the most certain to the least certain.

$$\frac{1}{n} * \sum_{i=1}^{n} \frac{\#-correct-up-to-pair-i}{i} \quad (2)$$

In the formula, n is the number of the pairs in the test set, and i ranges over the pairs. The Confidence-Weighted Score rewards the systems' ability to assign a higher confidence score to the correct judgements.

We also report *precision*, the accuracy for positive entailment pairs, *recall*, the percentage of positive entailment cases that are recognized, and the *F-measure*, a combination of precision and recall.

Collecting Hypothesis-Text Pairs from AutoTutor

One expert physicist rated the degree to which particular speech acts expressed during AutoTutor training matched particular expectations. These judgments were made on a sample of 25 physics expectations and 5 randomly sampled student answers per expectation, yielding a total of 125 pairs of expressions. The learner answers were always responses to the first hint for that expectation. The E-S pairs were graded by Physics experts on a scale of 1-4 (4 being perfect answer). This rubric could be used to prepare entailment tasks that deliver not only TRUE-FALSE decisions but also more fine-grained outputs. However, we followed the current RTE guidelines and transformed these numerical values to a discrete metric: scores 3 and 4 equal a TRUE decision and 1 and 2 equal a FALSE decision. We ended up with 36 FALSE and 89 TRUE entailment pairs, i.e. a 28.8% versus 71.2% split (as compared to the 50-50% split of RTE data). It should be noted that our data sample was somewhat different from the RTE data. Our data was a distribution over 5 students and a set of 5 problems. It would be interesting in the future to examine the individual distribution per student over a set of problems and the distribution per problem over a set of individuals.

Results

We conducted an evaluation of several methods in order to get a better idea how the presented lexico-syntactic approach works. The alternative approaches can be viewed as simplified approaches of our graph-based approach. First we tried a simple word overlap method: tokenize, lemmatize (using *wnstemm* in wn library), ignore punctuation, and compute the degree of lexical overlap between H and T. We normalized the result by dividing the lexical overlap by the total number of words in H. Then, if the normalized score was greater than a threshold (.50 in our case) we assigned a TRUE value (meaning T entails H), and otherwise we assigned FALSE. The normalized score also plays the role of the confidence score that is necessary to compute in the CWS metric. We next added synonymy to this word overlap method and measured the new performance. Those two alternatives are basically equivalent to the particular case of our lexico-syntactic method when the lexical component is the only one used ($\alpha = 1$) and the syntactic component is ignored ($\beta = 0$). The third alternative approach is a purely syntactic one, i.e. $\alpha = 0$ and $\beta = 1$. The evaluation of the latter allows us to see how much syntax alone can do for entailment. Table 2 shows a summary of the results.

system	cws	accuracy	precision	recall	F-measure
word overlap	0.6364	0.6160	0.8060	0.6067	0.6923
word overlap+synonymy	0.8008	0.7520	0.8222	0.8315	0.8268
syntax-only	0.6864	0.6400	0.8667	0.5843	0.6980
lexico-syntactic	0.7663	0.6960	0.8493	0.6966	0.7654

Table 2: Performance and comparison of different approaches on AutoTutor data.

The first three rows of the table present the results of the three alternative methods. The last row in the table shows the results on test data obtained with the lexico-syntactic approach. All of the results do not include negation handling. Due to space constraints we will address that problem in future publications.

We used the RTE-1 development data to estimate the parameters of the score equation and then applied the equation with the best fit parameters to the new test data. We used linear regression to estimate the values of the parameters and also experimented with balanced weighting ($\alpha = \beta = 0.5$, $\gamma = 0$). The balanced weighted scheme provided better results, so the results reported in Table 2 were computed with this scheme.

Conclusions

The results revealed that syntax leads to best precision on the test data. The word overlap and synonymy approach delivered the best accuracy and average precision(cws). The poorer results for accuracy and average precision of the lexico-syntactic approach should be regarded as a worst case scenario. That is simply because the syntactic information is extracted from partially correct parse trees. If the input from the parse tree were perfect, we might expect the gap (in cws and accuracy) between the lexico-syntactic approach and the word overlap and synonymy approach to be reduced or even reversed. Moreover, the word overlap approach is not affected by any type of errors because we use pure words. The results for the word overlap approach manifests its maximum potential.

Acknowledgements

This research was partially funded by The University of Memphis and AutoTutor project. The research on AutoTutor was supported by the National Science Foundation (REC 106965, ITR 0325428) and the DoD Multidisciplinary University Research Initiative (MURI) administered by ONR under grant N00014-00-1-0600.

Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of The University of Memphis, DoD, ONR, or NSF. We are also grateful to four anonymous reviewers for their valuable comments.

References

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of North American Chapter of Association for Computational Linguistics (NAACL-2000)*.

Dagan, I.; Glickman, O.; and Magnini, B. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the Recognizing Textual Entaiment Challenge Workshop*.

Graesser, A.; VanLehn, K.; Rose, C.; Jordan, P.; and Harter, D. 2001. Intelligent tutoring systems with conversational dialogue. *AI Magazine* 22:39–51.

Graesser, A.; Hu, X.; and McNamara, D. 2005. Computerized learning environments that incoporate research in discourse psychology, cognitive science, and computational linguistics. In Healy, A., ed., *Experimental Cognitive Psychology and its Applications*, 59–72. Washington, D.C.: American Psychological Association.

Landauer, T. K.; Foltz, P. W.; and Laham, D. 1998. Introduction to latent semantic analysis. *Discourse Processes* 25:259–284.

Magerman, D. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. Dissertation, Stanford University.

Marcus, M.; Santorini, B.; and Marcinkiewicz. 1993. Building a large annotated coprus of english: the penn treebank. *Computational Linguistic* 19(2):313–330.

Mel'cuk, I. 1998. *Dependency Syntax: theory and practice*. Albany, NY: State University of New York Press.

Miller, G. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Moldovan, D., and Rus, V. 2001. Logic form transformation of wordnet and its applicability to question answering. In *Proceedings of the ACL Conference (ACL-2001)*.

Monz, C., and de Rijke, M. 2001. *Light-Weight Entailment Checking for Computational Semantics*. 59–72.

Pazienza, M.; Pennacchiotti, M.; and Zanzotto, F. 2005. Textual entailment as syntactic graph distance: A rule based and svm based approach. In *Proceedings of the RTE Challenge Workshop*.

Rus, V., and Desai, K. 2005. Assigning function tags with a simple model. In *Proceedings of Conference on Intelligent Text Processing and Computational Linguistics (CICLing) 2005*.

Rus, V.; Graesser, A.; and Desai, K. 2005. A lexico-syntactic approach to textual entailment. In *Proceedings of Recent Advances in Natural Language Processing (RANLP) 2005*.

Skiena, S. 1998. *The Algorithm Design Manual*. Springer-Verlag.