Methods for Using Textual Entailment in Open-Domain Question Answering

Sanda Harabagiu and Andrew Hickl
Language Computer Corporation
1701 North Collins Boulevard
Richardson, Texas 75080 USA
sanda@languagecomputer.com

Abstract

Work on the semantics of questions has argued that the relation between a question and its answer(s) can be cast in terms of logical entailment. In this paper, we demonstrate how computational systems designed to recognize *textual entailment* can be used to enhance the accuracy of current open-domain automatic question answering (Q/A) systems. In our experiments, we show that when textual entailment information is used to either filter or rank answers returned by a Q/A system, accuracy can be increased by as much as 20% overall.

1 Introduction

Open-Domain Question Answering (Q/A) systems return a textual expression, identified from a vast document collection, as a response to a question asked in natural language. In the quest for producing accurate answers, the open-domain Q/A problem has been cast as: (1) a pipeline of linguistic processes pertaining to the processing of questions, relevant passages and candidate answers, interconnected by several types of lexico-semantic feedback (cf. (Harabagiu et al., 2001; Moldovan et al., 2002)); (2) a combination of language processes that transform questions and candidate answers in logic representations such that reasoning systems can select the correct answer based on their proofs (cf. (Moldovan et al., 2003)); (3) a noisy-channel model which selects the most likely answer to a question (cf. (Echihabi and Marcu, 2003)); or (4) a constraint satisfaction problem, where sets of auxiliary questions are used to provide more information and better constrain the answers to individual questions (cf. (Prager et al., 2004)). While different in their approach, each of these frameworks seeks to approximate the forms of semantic inference that will allow them to identify valid textual answers to natural language questions.

Recently, the task of automatically recognizing one form of semantic inference – textual entailment – has received much attention from groups participating in the 2005 and 2006 PASCAL Recognizing Textual Entailment (RTE) Challenges (Dagan et al., 2005). [1]

As currently defined, the RTE task requires systems to determine whether, given two text fragments, the meaning of one text could be reasonably inferred, or *textually entailed*, from the meaning of the other text. We believe that systems developed specifically for this task can provide current question-answering systems with valuable semantic information that can be leveraged to identify exact answers from ranked lists of candidate answers. By replacing the pairs of texts evaluated in the RTE Challenge with combinations of questions and candidate answers, we expect that textual entailment could provide yet another mechanism for approximating the types of inference needed in order answer questions accurately.

In this paper, we present three different methods for incorporating systems for textual entailment into the traditional Q/A architecture employed by many current systems. Our experimental results indicate that (even at their current level of performance) textual entailment systems can substantially improve the accuracy of Q/A, even when no other form of semantic inference is employed.

The remainder of the paper is organized as fol-

[1] http://www.pascal-network.org/Challenges/RTE

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 905–912,
Sydney, July 2006.

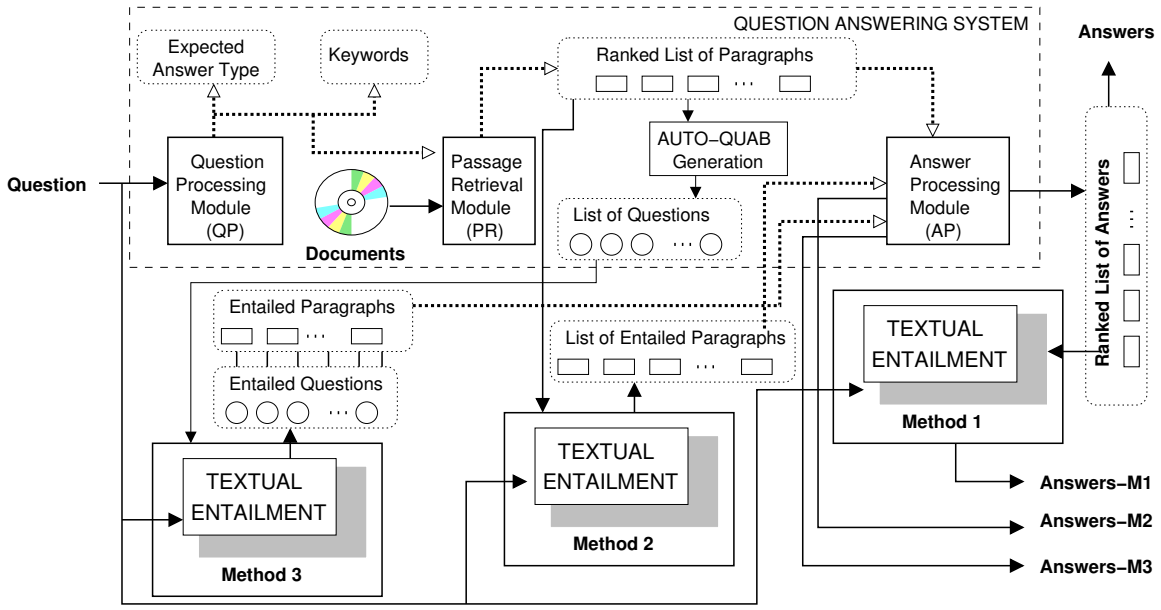

Figure 1: Integrating Textual Entailment in Q/A.

lows. Section 2 describes the three methods of using textual entailment in open-domain question answering that we have identified, while Section 3 presents the textual entailment system we have used. Section 4 details our experimental methods and our evaluation results. Finally, Section 5 provides a discussion of our findings, and Section 6 summarizes our conclusions.

2 Integrating Textual Entailment in Question Answering

In this section, we describe three different methods for integrating a textual entailment (TE) system into the architecture of an open-domain Q/A system.

Work on the semantics of questions (Groenendijk, 1999; Lewis, 1988) has argued that the formal answerhood relation found between a question and a set of (correct) answers can be cast in terms of logical entailment. Under these approaches (referred to as *licensing* by (Groenendijk, 1999) and *aboutness* by (Lewis, 1988)), p is considered to be an answer to a question $?q$ iff $?q$ logically entails the set of worlds in which p is true(i.e. $?p$). While the notion of *textual entailment* has been defined far less rigorously than logical entailment, we believe that the recognition of textual entailment between a question and a set of candidate answers – or between a question and questions generated from answers – can enable Q/A systems to identify correct answers with greater precision than current keyword- or pattern-based techniques.

As illustrated in Figure 1, most open-domain Q/A systems generally consist of a sequence of three modules: (1) a *question processing* (QP) module; (2) a *passage retrieval* (PR) module; and (3) an *answer processing* (AP) module. Questions are first submitted to a QP module, which extracts a set of relevant keywords from the text of the question and identifies the question's expected answer type (EAT). Keywords – along with the question's EAT – are then used by a PR module to retrieve a ranked list of paragraphs which may contain answers to the question. These paragraphs are then sent to an AP module, which extracts an exact *candidate answer* from each passage and then ranks each candidate answer according to the likelihood that it is a correct answer to the original question.

Method 1. In Method 1, each of a ranked list of answers that do not meet the minimum conditions for TE are removed from consideration and then re-ranked based on the *entailment confidence* (a real-valued number ranging from 0 to 1) assigned by the TE system to each remaining example. The system then outputs a new set of ranked answers which do not contain any answers that are not entailed by the user's question.

Table 1 provides an example where Method 1 could be used to make the right prediction for a set of answers. Even though A_1 was ranked in sixth position, the identification of a high-confidence positive entailment enabled it to be returned as the

top answer. In contrast, the recognition of a negative entailment for A_2 caused this answer to be dropped from consideration altogether.

Q_1: "*What did Peter Minuit buy for the equivalent of $24.00?*"				
	Rank_1	TE	Rank_2	Answer Text
A_1	6^{th}	YES (0.89)	1^{st}	Everyone knows that, back in 1626, Peter Minuit bought **Manhattan** from the Indians for $24 worth of trinkets.
A_2	1^{st}	NO (0.81)	–	In 1626, an enterprising Peter Minuit flagged down some passing locals, plied them with **beads, cloth and trinkets** worth an estimated $24, and walked away with the whole island.

Table 1: Re-ranking of answers by Method 1.

Method 2. Since AP is often a resource-intensive process for most Q/A systems, we expect that TE information can be used to limit the number of passages considered during AP. As illustrated in Method 2 in Figure 1, lists of passages retrieved by a PR module can either be ranked (or filtered) using TE information. Once ranking is complete, answer extraction takes place only on the set of *entailed passages* that the system considers likely to contain a correct answer to the user's question.

Method 3. In previous work (Harabagiu et al., 2005b), we have described techniques that can be used to automatically generate well-formed natural language questions from the text of paragraphs retrieved by a PR module. In our current system, sets of automatically-generated questions (AGQ) are created using a stand-alone *AutoQUAB* generation module, which assembles question-answer pairs (known as *QUABs*) from the top-ranked passages returned in response to a question. Table 2 lists some of the questions that this module has produced for the question Q_2: "*How hot does the inside of an active volcano get?*".

Q_2: "*How hot does the inside of an active volcano get?*"	
A_2	Tamagawa University volcano expert Takeyo Kosaka said lava fragments belched out of the mountain on January 31 were as hot as **300 degrees Fahrenheit**. The intense heat from a second eruption on Tuesday forced rescue operations to stop after 90 minutes. Because of the high temperatures, the bodies of only five of the volcano's initial victims were retrieved.
Positive Entailment	
AGQ_1	What temperature were the lava fragments belched out of the mountain on January 31?
AGQ_2	How many degrees Fahrenheit were the lava fragments belched out of the mountain on January 31?
Negative Entailment	
AGQ_3	When did rescue operations have to stop?
AGQ_4	How many bodies of the volcano's initial victims were retrieved?

Table 2: TE between AGQs and user question.

Following (Groenendijk, 1999), we expect that if a question $?q$ logically entails another question $?q'$, then some subset of the answers entailed by $?q'$ should also be interpreted as valid answers to $?q$. By establishing TE between a question and AGQs derived from passages identified by the Q/A system for that question, we expect we can identify a set of answer passages that contain correct answers to the original question. For example, in Table 2, we find that entailment between questions indicates the correctness of a candidate answer: here, establishing that Q_2 entails AGQ_1 and AGQ_2 (but not AGQ_3 or AGQ_4) enables the system to select A_2 as the correct answer.

When at least one of the AGQs generated by the AutoQUAB module is entailed by the original question, all AGQs that do not reach TE are filtered from consideration; remaining passages are assigned an entailment confidence score and are sent to the AP module in order to provide an exact answer to the question. Following this process, candidate answers extracted from the AP module were then re-associated with their AGQs and resubmitted to the TE system (as in Method 1). Question-answer pairs deemed to be positive instances of entailment were then stored in a database and used as additional training data for the AutoQUAB module. When no AGQs were found to be entailed by the original question, however, passages were ranked according to their entailment confidence and sent to AP for further processing and validation.

3 The Textual Entailment System

Processing textual entailment, or recognizing whether the information expressed in a text can be inferred from the information expressed in another text, can be performed in four ways. We can try to (1) derive linguistic information from the pair of texts, and cast the inference recognition as a classification problem; or (2) evaluate *the probability* that an entailment can exist between the two texts; (3) represent the knowledge from the pair of texts in some representation language that can be associated with an inferential mechanism; or (4) use the classical AI definition of entailment and build models of the world in which the two texts are respectively true, and then check whether the models associated with one text are included in the models associated with the other text. Although we believe that each of these methods should be investigated fully, we decided to focus only on the first method, which allowed us to build the TE system illustrated in Figure 2.

Our TE system consists of (1) a *Preprocessing Module*, which derives linguistic knowledge from the text pair; (2) an *Alignment Module*, which takes advantage of the notions of lexical alignment

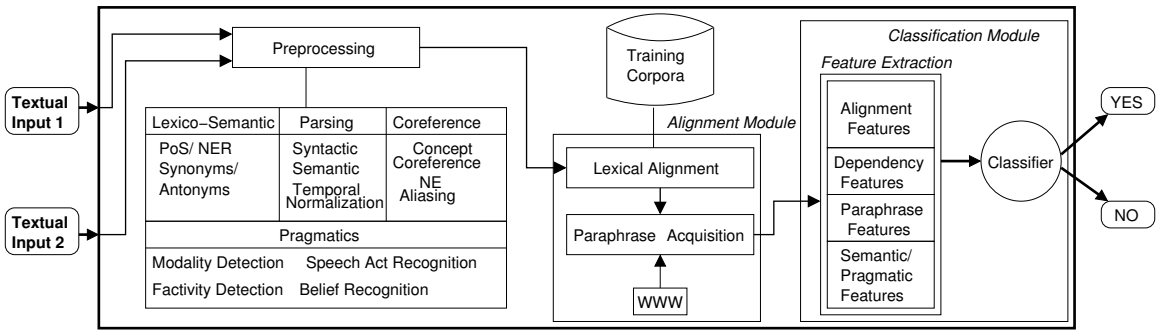

Figure 2: Textual Entailment Architecture.

and textual paraphrases; and (3) a *Classification Module*, which uses a machine learning classifier (based on decision trees) to make an entailment judgment for each pair of texts.

As described in (Hickl et al., 2006), the Preprocessing module is used to syntactically parse texts, identify the semantic dependencies of predicates, label named entities, normalize temporal and spatial expressions, resolve instances of coreference, and annotate predicates with polarity, tense, and modality information.

Following preprocessing, texts are sent to an *Alignment Module* which uses a Maximum Entropy-based classifier in order to estimate the probability that pairs of constituents selected from texts encode corresponding information that could be used to inform an entailment judgment. This module assumes that since sets of entailing texts necessarily predicate about the same set of individuals or events, systems should be able to identify elements from each text that convey similar types of presuppositions. Examples of predicates and arguments aligned by this module are presented in Figure 3.

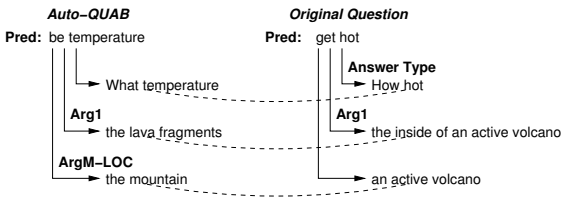

Figure 3: Alignment Graph

Aligned constituents are then used to extract sets of phrase-level alternations (or "paraphrases") from the WWW that could be used to capture correspondences between texts longer than individual constituents. The top 8 candidate paraphrases for two of the aligned elements from Figure 3 are presented in Table 3.

Finally, the *Classification Module* employs a decision tree classifier in order to determine whether an entailment relationship exists for each pair of texts. This classifier is learned using features extracted from the previous modules, including features derived from (1) the (lexical) alignment of the texts, (2) syntactic and semantic dependencies discovered in each text passage, (3) paraphrases derived from web documents, and (4) semantic and pragmatic annotations. (A complete list of features can be found in Figure 4.) Based on these features, the classifier outputs both an entailment judgment (either *yes* or *no*) and a confidence value, which is used to rank answers or paragraphs in the architecture illustrated in Figure 1.

Judgment	Paraphrase
YES	*lava fragments* in pyroclastic flows can reach *400 degrees*
YES	*an active volcano* can get up to *2000 degrees*
NO	*an active volcano* above you are slopes of *30 degrees*
YES	*the active volcano* with steam reaching *80 degrees*
YES	*lava fragments* such as cinders may still be as hot as *300 degrees*
NO	*lava* is a liquid at high temperature: typically from *700 degrees*

Table 3: Phrase-Level Alternations

3.1 Lexical Alignment

Several approaches to the RTE task have argued that the recognition of textual entailment can be enhanced when systems are able to identify – or align – corresponding entities, predicates, or phrases found in a pair of texts. In this section, we show that by using a machine learning-based classifier which combines lexico-semantic information from a wide range of sources, we are able to accurately identify aligned constituents in pairs of texts with over 90% accuracy.

We believe the alignment of corresponding entities can be cast as a classification problem which uses lexico-semantic features in order to compute an alignment probability *p(a)*, which corresponds to the likelihood that a term selected from one text entails a term from another text. We used constituency information from a chunk parser to decompose the pair of texts into a set of disjoint seg-

ALIGNMENT FEATURES: These three features are derived from the results of the lexical alignment classification.
◇1◇ LONGEST COMMON STRING: This feature represents the longest contiguous string common to both texts.
◇2◇ UNALIGNED CHUNK: This feature represents the number of chunks in one text that are not aligned with a chunk from the other
◇3◇ LEXICAL ENTAILMENT PROBABILITY: This feature is defined in (Glickman and Dagan, 2005).

DEPENDENCY FEATURES: These four features are computed from the PropBank-style annotations assigned by the semantic parser.
◇1◇ ENTITY-ARG MATCH: This is a boolean feature which fires when aligned entities were assigned the same argument role label.
◇2◇ ENTITY-NEAR-ARG MATCH: This feature is collapsing the arguments Arg_1 and Arg_2 (as well as the Arg_M subtypes) into single categories for the purpose of counting matches.
◇3◇ PREDICATE-ARG MATCH: This boolean feature is flagged when at least two aligned arguments have the same role.
◇4◇ PREDICATE-NEAR-ARG MATCH: This feature is collapsing the arguments Arg_1 and Arg_2 (as well as the Arg_M subtypes) into single categories for the purpose of counting matches.

PARAPHRASE FEATURES: These three features are derived from the paraphrases acquired for each pair.
◇1◇ SINGLE PATTERN MATCH: This is a boolean feature which fired when a paraphrase matched either of the texts.
◇2◇ BOTH PATTERN MATCH: This is a boolean feature which fired when paraphrases matched both texts.
◇3◇ CATEGORY MATCH: This is a boolean feature which fired when paraphrases could be found from the same paraphrase cluster that matched both texts.

SEMANTIC/PRAGMATIC FEATURES: These six features are extracted by the preprocessing module.
◇1◇ NAMED ENTITY CLASS: This feature has a different value for each of the 150 named entity classes.
◇2◇ TEMPORAL NORMALIZATION: This boolean feature is flagged when the temporal expressions are normalized to the same ISO 9000 equivalents.
◇3◇ MODALITY MARKER: This boolean feature is flagged when the two texts use the same modal verbs.
◇4◇ SPEECH-ACT: This boolean feature is flagged when the lexicons indicate the same speech act in both texts.
◇5◇ FACTIVITY MARKER: This boolean feature is flagged when the factivity markers indicate either TRUE or FALSE in both texts simultaneously.
◇6◇ BELIEF MARKER: This boolean feature is set when the belief markers indicate either TRUE or FALSE in both texts simultaneously.

CONTRAST FEATURES: These six features are derived from the opposing information provided by antonymy relations or chains.
◇1◇ NUMBER OF LEXICAL ANTONYMY RELATIONS: This feature counts the number of antonyms from WordNet that are discovered between the two texts.
◇2◇ NUMBER OF ANTONYMY CHAINS: This feature counts the number of antonymy chains that are discovered between the two texts.
◇3◇ CHAIN LENGTH: This feature represents a vector with the lengths of the antonymy chains discovered between the two texts.
◇4◇ NUMBER OF GLOSSES: This feature is a vector representing the number of Gloss relations used in each antonymy chain.
◇5◇ NUMBER OF MORPHOLOGICAL CHANGES: This feature is a vector representing the number of Morphological-Derivation relations found in each antonymy chain.
◇6◇ NUMBER OF NODES WITH DEPENDENCIES: This feature is a vector indexing the number of nodes in each antonymy chain that contain dependency relations.
◇7◇ TRUTH-VALUE MISMATCH: This is a boolean feature which fired when two aligned predicates differed in any truth value.
◇8◇ POLARITY MISMATCH: This is a boolean feature which fired when predicates were assigned opposite polarity values.

Figure 4: Features Used in Classifying Entailment

ments known as "alignable chunks". Alignable chunks from one text (C_t) and the other text (C_h) are then assembled into an alignment matrix ($C_t \times C_h$). Each pair of chunks ($p \in C_t \times C_h$) is then submitted to a Maximum Entropy-based classifier which determines whether or not the pair of chunks represents a case of lexical entailment.

Three classes of features were used in the Alignment Classifier: (1) a set of statistical features (e.g. cosine similarity), (2) a set of lexico-semantic features (including WordNet Similarity (Pedersen et al., 2004), named entity class equality, and part-of-speech equality), and (3) a set of string-based features (such as Levenshtein edit distance and morphological stem equality).

As in (Hickl et al., 2006), we used a two-step approach to obtain sufficient training data for the Alignment Classifier. First, humans were tasked with annotating a total of 10,000 alignment pairs (extracted from the 2006 PASCAL Development Set) as either positive or negative instances of alignment. These annotations were then used to train a hillclimber that was used to annotate a larger set of 450,000 alignment pairs selected at random from the training corpora described in Section 3.3. These machine-annotated examples were then used to train the Maximum Entropy-based classifier that was used in our TE system. Table 4 presents results from TE's linear- and Maximum Entropy-based Alignment Classifiers on a sample of 1000 alignment pairs selected at random from the 2006 PASCAL Test Set.

Classifier	Training Set	Precision	Recall	F-Measure
Linear	10K pairs	0.837	0.774	0.804
Maximum Entropy	10K pairs	0.881	0.851	0.866
Maximum Entropy	450K pairs	0.902	0.944	0.922

Table 4: Performance of Alignment Classifier

3.2 Paraphrase Acquisition

Much recent work on automatic paraphrasing (Barzilay and Lee, 2003) has used relatively simple statistical techniques to identify text passages that contain the same information from parallel corpora. Since sentence-level paraphrases are generally assumed to contain information about the same event, these approaches have generally assumed that all of the available paraphrases for a given sentence will include at least one pair of entities which can be used to extract sets of paraphrases from text.

The TE system uses a similar approach to gather phrase-level alternations for each entailment pair. In our system, the two highest-confidence entity alignments returned by the Lexical Alignment module were used to construct a query which was used to retrieve the top 500 documents from *Google*, as well as all matching instances from our training corpora described in Section 3.3. This method did not always extract true paraphrases of either texts. In order increase the likelihood that

only true paraphrases were considered as phrase-level alternations for an example, extracted sentences were clustered using complete-link clustering using a technique proposed in (Barzilay and Lee, 2003).

3.3 Creating New Sources of Training Data

In order to obtain more training data for our TE system, we extracted more than 200,000 examples of textual entailment from large newswire corpora.

Positive Examples. Following an idea proposed in (Burger and Ferro, 2005), we created a corpus of approximately 101,000 textual entailment examples by pairing the headline and first sentence from newswire documents. In order to increase the likelihood of including only positive examples, pairs were filtered that did not share an entity (or an NP) in common between the headline and the first sentence

Judgment	Example
YES	**Text-1:** Sydney newspapers made a secret deal not to report on the fawning and spending during the city's successful bid for the 2000 Olympics, former Olympics Minister Bruce Baird said today.
	Text-2: Papers Said To Protect Sydney Bid
YES	**Text-1:** An IOC member expelled in the Olympic bribery scandal was consistently drunk as he checked out Stockholm's bid for the 2004 Games and got so offensive that he was thrown out of a dinner party, Swedish officials said.
	Text-2: Officials Say IOC Member Was Drunk

Table 5: Positive Examples

Negative Examples. Two approaches were used to gather negative examples for our training set. First, we extracted 98,000 pairs of sequential sentences that included mentions of the same named entity from a large newswire corpus. We also extracted 21,000 pairs of sentences linked by connectives such as *even though, in contrast* and *but*.

Judgment	Example
NO	**Text-1:** One player losing a close friend is Japanese pitcher Hideki Irabu, who was befriended by Wells during spring training last year.
	Text-2: Irabu said he would take Wells out to dinner when the Yankees visit Toronto.
NO	**Text-1:** According to the professor, present methods of cleaning up oil slicks are extremely costly and are never completely efficient.
	Text-2: *In contrast*, he stressed, Clean Mag has a 100 percent pollution retrieval rate, is low cost and can be recycled.

Table 6: Negative Examples

4 Experimental Results

In this section, we describe results from four sets of experiments designed to explore how textual entailment information can be used to enhance the quality of automatic Q/A systems. We show that by incorporating features from TE into a Q/A system which employs no other form of textual inference, we can improve accuracy by more than 20% over a baseline.

We conducted our evaluations on a set of 500 factoid questions selected randomly from questions previously evaluated during the annual TREC Q/A evaluations. [2] Of these 500 questions, 335 (67.0%) were automatically assigned an answer type from our system's answer type hierarchy ; the remaining 165 (33.0%) questions were classified as having an unknown answer type. In order to provide a baseline for our experiments, we ran a version of our Q/A system, known as FERRET (Harabagiu et al., 2005a), that does not make use of textual entailment information when identifying answers to questions. Results from this baseline are presented in Table 7.

Question Set	Questions	Correct	Accuracy	MRR
Known Answer Types	335	107	32.0%	0.3001
Unknown Answer Types	265	81	30.6%	0.2987

Table 7: Q/A Accuracy without TE

The performance of the TE system described in Section 3 was first evaluated in the 2006 PASCAL RTE Challenge. In this task, systems were tasked with determining whether the meaning of a sentence (referred to as a *hypothesis*) could be reasonably inferred from the meaning of another sentence (known as a *text*). Four types of sentence pairs were evaluated in the 2006 RTE Challenge, including: pairs derived from the output of (1) automatic question-answering (QA) systems, (2) information extraction systems (IE), (3) information retrieval (IR) systems, and (4) multi-document summarization (SUM) systems. The accuracy of our TE system across these four tasks is presented in Table 8.

		Training Data	
		Development Set	Additional Corpora
Number of Examples		800	201,000
Task	QA-test	0.5750	0.6950
	IE-test	0.6450	0.7300
	IR-test	0.6200	0.7450
	SUM-test	0.7700	0.8450
Overall Accuracy		**0.6525**	**0.7538**

Table 8: Accuracy on the 2006 RTE Test Set

In previous work (Hickl et al., 2006), we have found that the type and amount of training data available to our TE system significantly ($p < 0.05$) impacted its performance on the 2006 RTE Test Set. When our system was trained on the training corpora described in Section 3.3, the overall accuracy of the system increased by more than 10%,

[2] Text Retrieval Conference (http://trec.nist.gov)

from 65.25% to 75.38%. In order to provide training data that replicated the task of recognizing entailment between a question and an answer, we assembled a corpus of 5000 question-answer pairs selected from answers that our baseline Q/A system returned in response to a new set of 1000 questions selected from the TREC test sets. 2500 positive training examples were created from answers identified by human annotators to be correct answers to a question, while 2500 negative examples were created by pairing questions with incorrect answers returned by the Q/A system.

After training our TE system on this corpus, we performed the following four experiments:

Method 1. In the first experiment, the ranked lists of answers produced by the Q/A system were submitted to the TE system for validation. Under this method, answers that were not entailed by the question were removed from consideration; the top-ranked entailed answer was then returned as the system's answer to the question. Results from this method are presented in Table 9.

Method 2. In this experiment, entailment information was used to rank passages returned by the PR module. After an initial relevance ranking was determined from the PR engine, the top 50 passages were paired with the original question and were submitted to the TE system. Passages were re-ranked using the entailment judgment and the entailment confidence computed for each pair and then submitted to the AP module. Features derived from the entailment confidence were then combined with the keyword- and relation-based features described in (Harabagiu et al., 2005a) in order to produce a final ranking of candidate answers. Results from this method are presented in Table 9.

Method 3. In the third experiment, TE was used to select AGQs that were entailed by the question submitted to the Q/A system. Here, AutoQUAB was used to generate questions for the top 50 candidate answers identified by the system. When at least one of the top 50 AGQs were entailed by the original question, the answer passage associated with the top-ranked entailed question was returned as the answer. When none of the top 50 AGQs were entailed by the question, question-answer pairs were re-ranked based on the entailment confidence, and the top-ranked answer was returned. Results for both of these conditions are presented in Table 9.

Hybrid Method. Finally, we found that the best results could be obtained by combining aspects of each of these three strategies. Under this approach, candidate answers were initially ranked using features derived from entailment classifications performed between (1) the original question and each candidate answer and (2) the original question and the AGQ generated from each candidate answer. Once a ranking was established, answers that were not judged to be entailed by the question were also removed from final ranking. Results from this hybrid method are provided in Table 9.

	Known EAT		Unknown EAT	
	Acc	MRR	Acc	MRR
Baseline	32.0%	0.3001	30.6%	0.2978
Method 1	44.1%	0.4114	39.5%	0.3833
Method 2	52.4%	0.5558	42.7%	0.4135
Method 3	41.5%	0.4257	37.5%	0.3575
Hybrid	53.9%	0.5640	41.9%	0.4010

Table 9: Q/A Performance with TE

5 Discussion

The experiments reported in this paper suggest that current TE systems may be able to provide open-domain Q/A systems with the forms of semantic inference needed to perform accurate answer validation. While probabilistic or web-based methods for answer validation have been previously explored in the literature (Magnini et al., 2002), these approaches have modeled the relationship between a question and a (correct) answer in terms of *relevance* and have not tried to approximate the deeper semantic phenomena that are involved in determining answerhood.

Our work suggests that considerable gains in performance can be obtained by incorporating TE during both answer processing and passage retrieval. While best results were obtained using the Hybrid Method (which boosted performance by nearly 28% for questions with known EATs), each of the individual methods managed to boost the overall accuracy of the Q/A system by at least 7%. When TE was used to filter non-entailed answers from consideration (Method 1), the overall accuracy of the Q/A system increased by 12% over the baseline (when an EAT could be identified) and by nearly 9% (when no EAT could be identified). In contrast, when entailment information was used to rank passages and candidate answers, performance increased by 22% and 10% respectively. Somewhat smaller performance gains were achieved when TE was used to select

amongst AGQs generated by our Q/A system's AutoQUAB module (Method 3). We expect that by adding features to TE system specifically designed to account for the semantic contributions of a question's EAT, we may be able to boost the performance of this method.

6 Conclusions

In this paper, we discussed three different ways that a state-of-the-art textual entailment system could be used to enhance the performance of an open-domain Q/A system. We have shown that when textual entailment information is used to either filter or rank candidate answers returned by a Q/A system, Q/A accuracy can be improved from 32% to 52% (when an answer type can be detected) and from 30% to 40% (when no answer type can be detected). We believe that these results suggest that current supervised machine learning approaches to the recognition of textual entailment may provide open-domain Q/A systems with the inferential information needed to develop viable answer validation systems.

7 Acknowledgments

This material is based upon work funded in whole or in part by the U.S. Government and any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Government.

References

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *HLT-NAACL*.

John Burger and Lisa Ferro. 2005. Generating an Entailment Corpus from News Headlines. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 49–54.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop*.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

Oren Glickman and Ido Dagan. 2005. A Probabilistic Setting and Lexical Co-occurrence Model for Textual Entailment. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, USA.

Jeroen Groenendijk. 1999. The logic of interrogation: Classical version. In *Proceedings of the Ninth Semantics and Linguistics Theory Conference (SALT IX)*, Ithaca, NY.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunsecu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2001. The Role of Lexico-Semantic Feedback in Open-Domain Textual Question-Answering. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*.

S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, A. Hickl, and P. Wang. 2005a. Employing Two Question Answering Systems in TREC 2005. In *Proceedings of the Fourteenth Text REtrieval Conference*.

Sanda Harabagiu, Andrew Hickl, John Lehmann, and Dan Moldovan. 2005b. Experiments with Interactive Question-Answering. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing Textual Entailment with LCC's Groundhog System. In *Proceedings of the Second PASCAL Challenges Workshop*.

David Lewis. 1988. Relevant Implication. *Theoria*, 54(3):161–174.

Bernardo Magnini, Matteo Negri, Roberto Prevete, and Hristo Tanev. 2002. Is it the right answer? exploiting web redundancy for answer validation. In *Proceedings of the Fortieth Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, PA.

Dan Moldovan, Marius Pasca, Sanda Harabagiu, and Mihai Surdeanu. 2002. Performance Issues and Error Analysis in an Open-Domain Question Answering System. In *Proceedings of the 4Oth Meeting of the Association for Computational Linguistics*.

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. 2003. COGEX: A Logic Prover for Question Answering. In *Proceedings of HLT/NAACL-2003*.

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, San Jose, CA.

John Prager, Jennifer Chu-Carroll, and Krzysztof Czuba. 2004. Question answering using constraint satisfaction: Qa-by-dossier-with-contraints. In *Proceedings of the ACL-2004*, pages 574–581, Barcelona, Spain, July.