

Applying COGEX to Recognize Textual Entailment

Abraham Fowler, Bob Hauser, Daniel Hodges, Ian Niles, Adrian Novischi, Jens Stephan

Language Computer Corporation
Richardson, Texas, 75080

{abraham,hauser,daniel,adrian,jens}@languagecomputer.com

Abstract

The PASCAL RTE challenge has helped LCC to explore the applicability of enhancements that have been made to our logic form representation and WordNet lexical chains generator. Our system transforms each T-H pair into logic form representation with semantic relations. The system automatically generates NLP axioms serving as linguistic rewriting rules and lexical chain axioms that connect concepts in the hypothesis and text. A light set of simple hand-coded world knowledge axioms are also included. Our COGEX logic prover is then used to attempt to prove entailment. Semantic relations, WordNet lexical chains, and NLP axioms all helped the logic prover detect entailment.

1 Introduction

Textual entailment occurs when one text can be inferred from the meaning/contents of another text passage. All assertions made in an entailed sentence must be made in the text passage directly, or be logically derivable from it. Our approach attempts to recognize textual entailment by determining if the hypothesis sentence can be logically derived from the text passage using a logic prover. The goal of a logic prover is to determine if some hypothetical statement can be proven given a set of other known true statements. Our logic prover operates by “reductio ad absurdum” or “proof by contradiction”

(Wos, 1988). The hypothesis is negated, and if it then contradicts anything in the text or anything inferred from the text, the prover concludes that the original hypothetical statement is derivable from the text (thus, entailment exists).

A description of our system’s implementation is provided in Section 2, our results and some performance analysis are included in Section 3, and some final concluding remarks are made in Section 4.

2 System Description

2.1 Logic Form Transformation

In the first stage of our system, the input text and hypothesis are converted into logic forms (Moldovan and Rus, 2001). This conversion process includes part-of-speech tagging, parse tree generation, word sense disambiguation, and semantic relations detection. In the final representation, word senses are removed from the predicates. We found that the inaccuracy of the word sense disambiguator was so great that it prevented many of our other tools from being properly utilized.

2.2 Axiom Generation

We have implemented our COGEX (Moldovan et al., 2003) logic prover into the entailment recognition system. COGEX is a modified version of the OTTER (McCune, 1994) logic prover that has been adapted for natural language processing. The prover requires a list of clauses called the “set of support” which is used to initiate the search for inferences. The set of support is loaded with the negated form of the hypothesis as well as the predicates that make up the text passage. A second list, called the usable

list, contains clauses used by OTTER to generate inferences. In our system the usable list consists of all the axioms that have been generated either automatically or by hand. Axioms in our system are utilized to provide external world knowledge, knowledge of syntactic equivalence between logic form predicates, and lexical knowledge in the form of lexical chains.

2.2.1 World Knowledge Axioms

We incorporate a small common-sense knowledge base of 310 world knowledge axioms, where 80 have been manually designed based on the development set data, and 230 originate from previous projects. Currently, this data set is too small to have a significant impact, but in combination with Lexical Chains, the coverage of these axioms will grow.

2.2.2 NLP Axioms

Our NLP Axioms are linguistic rewriting rules that help break down complex logic structures and express syntactic equivalence. These axioms are automatically generated by the system through logic form and parse tree analysis. Axioms are generated to break down complex nominals and coordinating conjunctions into their components so that other axioms can be applied to the components individually to generate a larger set of inferences. Other axioms help us: (1) establish equivalence between prepositions, (2) establish equivalence between different parts of speech, (3) equate words that have multiple noun forms, and (4) equate substantives within appositions.

2.2.3 WordNet Lexical Chains

WordNet provides links between synsets. Each synset has a set of corresponding predicates for each word in the synonym set. The name of a predicate is formed by synonym word form, its part of speech, and WordNet sense. A predicate can have one or more arguments. The predicates corresponding to noun synsets usually have a single argument and the predicates corresponding to verb synsets have three arguments: event, subject, and object arguments.

A lexical chain is a chain of relations between two synsets. For each relation in the chain, the system generates an axiom using the predicates corresponding to the synsets in the relation. The axiom states

that the predicate from the first synset implies the predicate from the second. For example, there is an ENTAILMENT relation between the verbs buy and pay. The system generates the following axiom for this relation:

$$\text{buy_VB_1}(e1,x1,x2) \rightarrow \text{pay_VB_1}(e1,x1,x3)$$

These axioms help the logic prover infer target concepts from starting concepts when lexical chains are found between the two. Not all WordNet relations are used for generating axioms. The following three classes of relations are used: pure WordNet relations, relations created from WordNet derivational morphology, and relations extracted from WordNet glosses. A detailed description of the system as a whole can be found in (Novischi, 2005) and (Moldovan and Novischi, 2002).

2.3 Logic Prover

Once the set of support and usable lists are complete, the logic prover can begin searching for proofs. The clauses in the set of support list are weighted in the order in which they should be chosen to participate in the search. The negated hypothesis is assigned the largest weight to ensure that it will be the last clause to participate in the search. The logic prover removes the clause with the smallest weight from the set of support, and searches the usable list for new inferences that can be made. Any inferences that are produced are assigned an appropriate weight depending on what axiom they were derived from and appended to the set of support list. The logic prover continues in this fashion until the set of support list is empty. If a refutation is found, then the proof is complete. If a refutation cannot be found, then predicate arguments are relaxed. If argument relaxation fails to produce a refutation, predicates are dropped from the negated hypothesis until a refutation is found. Once a proof by refutation is found, a score for that proof is calculated by starting with an initial perfect score and deducting points for axioms that are utilized in the proof, arguments that are relaxed, and predicates that are dropped.

2.4 Scoring

The score generated by the logic prover is only a measure of the kinds of axioms used in the proof and the significance of the dropped arguments and predicates. T-H pairs with longer sentences can poten-

tially drop more predicates, resulting in lower prover scores. Scores are normalized by first calculating the maximum penalty that can be assessed to a pair by dropping all of the hypothesis’ predicates. The penalty assessed by the logic prover is then divided by the maximum drop penalty to determine the normalized score.

Due to the logic prover’s relaxation techniques, it is always successful in producing a proof. The determination of whether entailment exists is made by examining the penalties assessed by the logic prover in the process of generating the proof. As more axioms are utilized and more predicates are dropped, it becomes much less likely that entailment exists between a pair. All normalized prover scores that fall below a specified threshold are considered false entailment and all scores that are above the threshold are considered true entailment. An appropriate threshold is calculated by examining the scoring output of the development data set to determine what threshold produces the highest accuracy.

The confidence score for a T-H pair in our system is measured as the distance between the normalized score and the threshold. Normalized scores that are further from the threshold will have a higher confidence score than normalized scores that are closer to the threshold. The difference between the normalized score and the threshold itself is normalized such that the resulting confidence score is a value between zero and one.

3 Performance Evaluation

Our results for the challenge are summarized in Table 1. As evidenced by these results, our system performs significantly better on T-H pairs in the comparable documents task. Due to the way T-H pairs are chosen in this task, there is often little to no information in the text of false pairs that could help us logically infer the hypothesis. This inferencing inability causes the logic prover to drop a large number of predicates and return extremely low scores for the false entailment pairs. The large difference between the true and false entailment scores allows us to easily separate the pairs.

The average scores for true and false entailment varied significantly over all of the tasks. This large variance makes it extremely difficult to choose a sin-

Task	Accuracy	CWS	F-measure
test-IR	.478	.386	.472
test-CD	.780	.822	.736
test-RC	.514	.534	.558
test-QA	.485	.434	.481
test-IE	.483	.580	.603
test-MT	.542	.440	.444
test-PP	.450	.450	.585
test-all	.551	.560	.561
dev-all	.630	.639	.619

Table 1: Results for the test and development sets.

gle threshold that can be used to detect entailment for all of the tasks. By selecting thresholds specific to each task, we were able to increase the test set’s accuracy to .562. This accuracy is still considerably lower than the accuracy we received on the development set from which the thresholds were chosen.

The numerous disagreements we had amongst ourselves and with the “Gold Standard” annotations leads us to believe to that the only appropriate way to calculate an upper bound for this task is to utilize the Kappa agreement metric. However, without a large set of different human annotations for the data set, it is impossible to calculate this metric.

Before evaluating the T-H pairs in the test set with our system, we manually determined how difficult it is to prove entailment in each of the true entailment T-H pairs. We established five different difficulty levels: easy, moderate, difficult, intractable, and invalid. Proofs are considered easy in cases where the entailment is simply a matter of eliminating information from the first sentence, recognizing an apposition or replacing one or two words with synonyms. Consider the following example:

Text: *A Union Pacific freight train hit five people.*

Hypothesis: *A Union Pacific freight train struck five people.*

Lexical Chain: *hit_VB* → *strike_VB*

Proofs are considered moderate when one or more inference rules are needed to derive the second sentence of the entailment pair from the first one. Consider the following example:

Text: *Satomi Mitarai died of blood loss.*

Hypothesis: *Satomi Mitarai bled to death.*

World Knowledge Axiom: *die_VB(e1,x1,x2) &*

of_IN(e1,x2) & nn_NNC(x2,x3,x4) & blood_NN(x3) & loss_NN(x4) → bleed_VB(e2,x1,x5) & to_TO(e1, x5) & death_NN(x5)

The expectation is that all entailment pairs that have been deemed easy or moderate can be handled by our current system implementation. Difficult proofs are those that cannot be handled by our theorem prover without adding substantial new functionality (coreference resolution, predicate variables in rules, etc.) or without using ad hoc rules (those not applicable beyond the case which motivates them). The following example requires very specific axioms and coreference resolution:

Text: *Israeli Prime Minister Ariel Sharon threatened to dismiss Cabinet ministers who don't support his plan to withdraw from the Gaza Strip.*

Hypothesis: *Israeli Prime Minister Ariel Sharon threatened to fire cabinet opponents of his Gaza withdrawal plan.*

We have labeled T-H pairs as intractable if we believe that entailment could not be correctly detected by an automated system. Invalid is used to indicate that, in our opinion, an entailment pair which was labeled TRUE should have been labeled FALSE. In the following pair the text does not imply that Silvio Berlusconi is Prime Minister of Italy, only that he is a prime minister with a mandate to reform Italy.

Text: *Prime Minister Silvio Berlusconi was elected March 28 with a mandate to reform Italy's business regulations and pull the economy out of recession.*

Hypothesis: *The Italian Prime Minister is Silvio Berlusconi.*

The system's performance on the T-H pairs classified as easy or moderate is significantly better than its performance on other pairs as illustrated in Table 2. Since many of the T-H pairs with the moderate classification require some external world knowledge, we suspect that with a larger knowledge base, the accuracy of the T-H pairs classified as moderate would be significantly higher.

It may be possible to build a classifier to determine the inference difficulty and only return results for pairs it deems to be easy or moderate. The main difficulty with such an approach is that it is hard to classify the difficulty of an inference without knowing whether the inference is true or false. We suspect that a difficulty classifier would have trouble distin-

Difficulty	Pairs	Accuracy	CWS
easy	81	.852	.892
moderate	122	.582	.610
difficult	126	.444	.413
intractable	1	1.000	1.000
invalid	70	.457	.501

Table 2: Results for the true entailment pairs categorized by proof difficulty

guishing difficult true entailments from easy false entailments, and vice versa.

4 Conclusion

We participated in the RTE challenge mainly as a learning experience and a test of our existing logic prover system implemented in a new way. Adding semantic relations to the logic form provided deeper semantic connectivity between concepts. This made it possible to write more abstract (more generally-applicable) world knowledge axioms. WordNet lexical chains helped to connect related concepts that used different words or different forms of the same word. And finally, based on linguistic patterns, the NLP axioms helped to link concepts that would otherwise not be connected in the logic form transformation.

References

- William W. McCune, 1994. *OTTER Reference Manual and Guide*. Argonne National Laboratory, Illinois, USA, 3.0 edition, January.
- Dan I. Moldovan and Adrian Novischi. 2002. Lexical chains for question answering. In *COLING*.
- Dan I. Moldovan and Vasile Rus. 2001. Logic form transformation of wordnet and its applicability to question answering. In *Meeting of the Association for Computational Linguistics*, pages 394–401.
- Dan I. Moldovan, Christine Clark, Sanda M. Harabagiu, and Steven J. Maiorano. 2003. Cogex: A logic prover for question answering. In *HLT-NAACL*.
- Adrian Novischi. 2005. *Semantic Disambiguation of WordNet glosses*. Ph.D. thesis, University of Texas at Dallas.
- L. Wos. 1988. *Automated Reasoning - 33 Basic Research Problems*. Prentice-Hall.