Recognising Textual Entailment with Logical Inference

Johan Bos
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh, EH8 9LW
jbos@inf.ed.ac.uk

Katja Markert
School of Computing
University of Leeds
Woodhouse Lane
Leeds, LS2 9JT
markert@comp.leeds.ac.uk

Abstract

We use logical inference techniques for recognising textual entailment. As the performance of theorem proving turns out to be highly dependent on not readily available background knowledge, we incorporate model building, a technique borrowed from automated reasoning, and show that it is a useful robust method to approximate entailment. Finally, we use machine learning to combine these deep semantic analysis techniques with simple shallow word overlap; the resulting hybrid model achieves high accuracy on the RTE testset, given the state of the art. Our results also show that the different techniques that we employ perform very differently on some of the subsets of the RTE corpus and as a result, it is useful to use the nature of the dataset as a feature.

1 Introduction

Recognising textual entailment (RTE) is the task to find out whether some text T entails a hypothesis H. This task has recently been the focus of a challenge organised by the PASCAL network in 2004/5.[1] In Example 1550 H follows from T whereas this is not the case in Example 731.

[1]All examples are from the corpus released as part of the RTE challenge. It is downloadable from http://www.pascal-network.org/Challenges/RTE/. The example numbers have also been kept. Each example is marked for entailment as TRUE if H follows from T and FALSE otherwise. The dataset is described in Section 4.1.

Example: 1550 (TRUE)
T: In 1998, the General Assembly of the Nippon Sei Ko Kai (Anglican Church in Japan) voted to accept female priests.
H: The Anglican church in Japan approved the ordination of women.

Example: 731 (FALSE)
T: The city Tenochtitlan grew rapidly and was the center of the Aztec's great empire.
H: Tenochtitlan quickly spread over the island, marshes, and swamps.

The recognition of textual entailment is without doubt one of the ultimate challenges for any NLP system: if it is able to do so with reasonable accuracy, it is clearly an indication that it has some thorough understanding of how language works. Indeed, recognising entailment bears similarities to Turing's famous test to assess whether machines can think, as access to different sources of knowledge and the ability to draw inferences seem to be among the primary ingredients for an intelligent system. Moreover, many NLP tasks have strong links to entailment: in summarisation, a summary should be entailed by the text; paraphrases can be seen as mutual entailment between T and H; in IE, the extracted information should also be entailed by the text.

In this paper, we discuss two methods for recognising textual entailment: a shallow method relying mainly on word overlap (Section 2), and deep semantic analysis, using state-of-the-art off-the-shelf inference tools, namely a theorem prover and a model builder (Section 3). These tools rely on Discourse Representation Structures for T and H as well as lexical and world knowledge. To our knowledge, few approaches to entailment currently use theorem provers and none incorporate model building (see

Section 5 for a discussion of related work).

Both methods are domain-independent to increase transferrability and have not been tailored to any particular test suite. In Section 4 we test their accuracy and robustness on the RTE datasets as one of the few currently available datasets for textual inference. We also combine the two methods in a hybrid approach using machine learning. We discuss particularly the following questions:

- Can the methods presented improve significantly over the baseline and what are the performance differences between them? Does the hybrid system using both shallow and deep semantic analysis improve over the individual use of these methods?
- How far does deep semantic analysis suffer from a lack of lexical and world knowledge and how can we perform logical inference in the face of potentially large knowledge gaps?
- How does the design of the test suite affect performance? Are there subsets of the test suite that are more suited to any particular textual entailment recognition method?

2 Shallow Semantic Features

We use several shallow surface features to model the text, hypothesis and their relation to each other.

Most importantly, we expect some dependency between surface string similarity of text and hypothesis and the existence of entailment. Our string similarity measure uses only a form of extended word overlap between text and hypothesis, taking into account equality of words, synonymy and morphological derivations. WordNet (Fellbaum, 1998) is used as the knowledge source for synonymy and derivations. The exact procedure is as follows:

Both text and hypothesis are tokenised and lemmatised. A lemma l_1 in the hypothesis is said to be *related* to a lemma l_2 in the text iff l_1 and l_2 are equal, belong to the same WordNet synset (e.g., "murder" and "slay"), are related via WordNet derivations (e.g. "murder" and "murderer") or are related via a combination of synonymy and derivations (e.g. "murder" via "murderer" to "liquidator"). No word sense disambiguation is performed and *all* synsets for a particular lemma are considered.

In addition, each lemma in the hypothesis is assigned its inverse document frequency, accessing the Web as corpus via the GoogleAPI, as its weight. This standard procedure allows us to assign more importance to less frequent words.

The overlap measure `wnoverlap` between text and hypothesis is initialised as zero. Should a lemma in the hypothesis be related to a lemma in the text, its weight is added to `wnoverlap`, otherwise it is ignored. In the end `wnoverlap` is normalised by dividing it by the sum of all weights of the lemmas in the hypothesis. This ensures that `wnoverlap` is always a real number between 0 and 1 and also ensures independence of the length of the hypothesis.

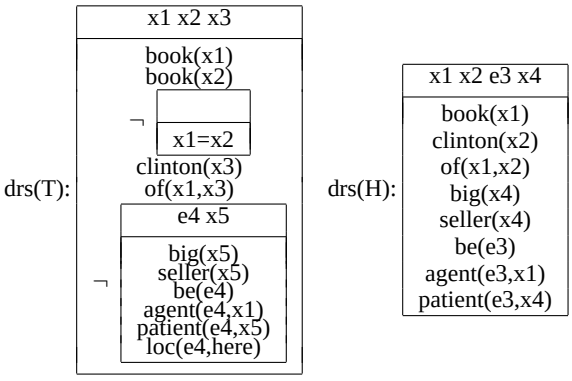
Apart from `wnoverlap` we take into account length (as measured by number of lemmas) of text and hypothesis, because in most of the observed cases for true entailments the hypothesis is shorter than the text as it contains less information. This is covered by three numerical features measuring the length of the text, of the hypothesis and the relative length of hypothesis with regard to the text.

3 Deep Semantic Analysis

3.1 Semantic Interpretation

We use a robust wide-coverage CCG-parser (Bos et al., 2004) to generate fine-grained semantic representations for each T/H-pair. The semantic representation language is a first-order fragment of the DRS-language used in Discourse Representation Theory (Kamp and Reyle, 1993), conveying argument structure with a neo-Davidsonian analysis and including the recursive DRS structure to cover negation, disjunction, and implication. Consider for example:

Example: 78 (FALSE)
T: Clinton's new book is not big seller here.
H: Clinton's book is a big seller.



Proper names and definite descriptions are treated as anaphoric, and bound to previously introduced discourse referents if possible, otherwise accommodated. Some lexical items are specified as presupposition triggers. An example is the adjective 'new' which has a presuppositional reading, as shown by the existence of two different "book" entities in drs(T). Scope is fully specified.

To check whether an entailment holds or not, we use two kinds of automated reasoning tools: Vampire, a theorem prover (Riazanov and Voronkov, 2002), and Paradox, a model builder (Claessen and Sörensson, 2003). Both tools are developed to deal with inference problems stated in first-order logic. We use the standard translation from DRS to first-order logic (Kamp and Reyle, 1993) to map our semantic representation onto the format required by the inference tools.

3.2 Theorem Proving

Given a T/H pair, a theorem prover can be used to find answers to the following conjectures:

1. T implies H (shows entailment)
2. T+H are inconsistent (shows no entailment)

Assume that the function DRS denotes the DRS corresponding to T or H, and FOL the function that translates a DRS into first-order logic. Then, if the theorem prover manages to find a proof for

$$\text{FOL}(\text{DRS}(\text{T}))\rightarrow\text{FOL}(\text{DRS}(\text{H})) \qquad \text{(A)}$$

we know that we are dealing with a true entailment. In addition, to use a theorem prover to detect inconsistencies in a T/H pair, we give it:

$$\neg\text{FOL}(\text{DRS}(\text{T});\text{DRS}(\text{H})) \qquad \text{(B)}$$

If the theorem prover returns a proof for (B), we know that T and H are inconsistent and T definitely doesn't entail H (assuming that T and H are themselves consistent).

Examples The theorem prover will find that T implies H for the following examples:

Example: 1005 (TRUE)
T: Jessica Litman, a law professor at Michigan's Wayne State University, has specialized in copyright law and Internet law for more than 20 years.
H: Jessica Litman is a law professor.

Example: 1977 (TRUE)
T: His family has steadfastly denied the charges.
H: The charges were denied by his family.

Example: 898 (TRUE)
T: After the war the city was briefly occupied by the Allies and then was returned to the Dutch.
H: After the war, the city was returned to the Dutch.

Example: 1952 (TRUE)
T: Crude oil prices soared to record levels.
H: Crude oil prices rise.

These examples show how deep semantic analysis deals effectively with apposition, active-passive alternation, coordination, and can integrate lexical knowledge.

The RTE dataset only contains a few inconsistent T/H pairs. Even although Example 78 might look like a case in point, it is not inconsistent: It would be if the T in the example would have been *Clinton's new book is not a big seller*. The addition of the adverb *here* makes T+H consistent.

3.3 Background Knowledge

The theorem prover needs background knowledge to support its proofs. Finding a proof for Example 1952 above is only possible if the theorem prover knows that soaring is a way of rising.

How does it know this? Because in addition to the information from T and H alone, we also supply relevant background knowledge in the form of first-order axioms. Instead of giving just $\text{FOL}(\text{DRS}(\text{T});\text{DRS}(\text{H}))$ to the theorem prover, we supply it with $(\text{BK} \wedge \text{FOL}(\text{DRS}(\text{T});\text{DRS}(\text{H})))$ where BK is short for the relevant background knowledge.

We generate background knowledge using three kinds of sources: generic knowledge, lexical knowledge, and geographical knowledge. Axioms for generic knowledge cover the semantics of possessives, active-passive alternation, and spatial knowledge. There are about 20 different axioms in the current system and these are the only manually generated axioms. An example is

$$\forall e \forall x \forall y(\text{event}(e) \wedge \text{agent}(e,x) \wedge \text{in}(e,y) \rightarrow \text{in}(x,y))$$

which states that if an event is located in y, then so is the agent of that event.

Lexical knowledge is created automatically from WordNet. A hyponymy relation between two

synsets A and B is converted into $\forall$x(A(x)$\rightarrow$B(x)). Two synset sisters A and B are translated into $\forall$x(A(x)$\rightarrow$ $\neg$B(x)). Here the predicate symbols from the DRS are mapped to WordNet synsets using a variant of Lesk's WSD algorithm (Manning and Schuetze, 1999). Examples 78 and 1952 would be supported by knowledge similar to:

$\forall$x(clinton(x)$\rightarrow$person(x)) $\forall$x(book(x)$\rightarrow$artifact(x))
$\forall$x(artifact(x)$\rightarrow$ $\neg$person(x)) $\forall$x(soar(x)$\rightarrow$rise(x))

Finally, axioms covering geographical knowledge about capitals, countries and US states are extracted automatically from the CIA factbook. An example:

$\forall$x$\forall$y(paris(x)$\wedge$france(y)$\rightarrow$in(x,y))

3.4 Model Building

While theorem provers are designed to prove that a formula *is* a theorem (i.e., that the formula is true in any model), they are generally not good at deciding that a formula is *not* a theorem. Model builders are designed to show that a formula is true in at least one model. To exploit these complementary approaches to inference, we use both a theorem prover and a model builder for any inference problem: the theorem prover attempts to prove the input whereas the model builder simultaneously tries to find a model for the negation of the input. If the model builder finds a model for

$$\neg\text{FOL}(\text{DRS}(\text{T}))\rightarrow\text{FOL}(\text{DRS}(\text{H})) \quad (= \neg\text{A})$$

we know that there can't be a proof for its negation (hence no entailment). And if the model builder is able to generate a model for

$$\text{FOL}(\text{DRS}(\text{T});\text{DRS}(\text{H})) \quad (= \neg\text{B})$$

we know that T and H are consistent (maybe entailment). (In practice, this is also a good way to terminate the search for proofs or models: if the theorem prover finds a proof for $\neg\phi$, we can halt the model builder to try and find a model for ϕ (because there won't be one), and vice versa.)

Another attractive property of a model builder is that it outputs a model for its input formula (only of course if the input is satisfiable). A model is here the logical notion of a model, describing a situation in which the input formula is true. Formally, a model is a pair $\langle D, F\rangle$ where D is the set of entities in the domain, and F a function mapping predicate symbols to sets of domain members. For instance, the model returned for fol(drs(T)) in Example 78 is one where the domain consists of three entities (domain size = 3):

```
D = {d1,d2,d3}      F(loc) = {}
F(book) = {d1,d2}   F(seller) = {}
F(clinton) = {d3}   F(be) = {}
F(of) = {(d1,d3)}   F(agent) = {}
F(big) = {}         F(patient) = {}
```

Model builders like Paradox generate finite models by iteration. They attempt to create a model for domain size 1. If they fail, they increase the domain size and try again, until either they find a model or their resources run out. Thus, although there are infinitely many models satisfying fol(drs(T)), model builders generally build a model with a minimal domain size. (For more information on model building consult (Blackburn and Bos, 2005)).

3.5 Approximating Entailment

In an ideal world we calculate all the required background knowledge and by either finding a proof or a countermodel, decide how T and H relate with respect to entailment. However, it is extremely hard to acquire all the required background knowledge. This is partly due to the limitations of word sense disambiguation, the lack of resources like WordNet, and the lack of general knowledge in a form suitable for automatic inference tasks.

To introduce an element of robustness into our approach, we use the models as produced by the model builders to measure the "distance" from an entailment. The intuition behind it is as follows. If H is entailed by T, the model for T+H is not informative compared to the one for T, and hence does not introduce new entities. Put differently, the domain size for T+H would equal the domain size of T. In contrast, if T does not entail H, H normally introduce some new information (except when it contains negated information), and this will be reflected in the domain size of T+H, which then is larger than the domain size of T. It turns out that this difference between the domain sizes is a useful way of measuring the likelihood of entailment. Large differences are mostly not entailments, small differences mostly are. Consider the following example:

Example: 1049 (TRUE)

T: Four Venezuelan firefighters who were traveling to a training course in Texas were killed when their sport utility vehicle drifted onto the shoulder of a highway and struck a parked truck.

H: Four firefighters were killed in a car accident.

Although this example is judged as a true entailment, Vampire doesn't find a proof because it lacks the background knowledge that one way of causing a car accident is to drift onto the shoulder of the highway and strike something. It generates a model with domain size 11 for fol(drs(T)), and a model with domain size 12 for fol((drs(T);drs(H))). The absolute difference in domain sizes is small, and therefore likely to indicate an entailment. Apart from the absolute difference we also compute the difference relative to the domain size. For the example above the relative domain size yields $1/12 = 0.083$.

The domain size only tells us something about the number of entities used in a model—not about the number of established relations between the model's entities. Therefore, we also introduce the notion of model size. The model size is defined here by counting the number of all instances of two-place relations (and three-place relations, if there are any) in the model, and multiplying this with the domain size. For instance, the following model

```
D = {d1,d2,d3}
F(cat) = {d1,d2}
F(john) = {d3}
F(of) = {(d1,d3)}
F(like) = {(d3,d1),(d3,d2)}
```

has a domain size of 3 and 3 instantiated two-place relations, yielding a model size of $3 * 3 = 9$.

3.6 Deep Semantic Features

Given our approach to deep semantic analysis, we identified eight features relevant for recognising textual entailment. The theorem prover provides us with two features: `entailed` determining whether T implies H, and `inconsistent` determining whether T together with H is inconsistent. The model builder gives us six features: `domainsize` and `modelsize` for T+H as well as the absolute and relative difference between the sizes of T and T+H, both for the size of the domains (`domainsizeabsdif`, `domainsizereldif`) and the size of the models (`modelsizeabsdif`, `modelsizereldif`).

4 Experiments

There are not many test suites available for textual inference. We use throughout this section the dataset made available as part of the RTE challenge.

4.1 Dataset Design and Evaluation Measures

The organisers released a development set of 567 sentence pairs and a test set of 800 sentence pairs. In both sets, 50% of the sentence pairs were annotated as TRUE and 50% as FALSE, leading to a 50% most frequent class baseline for automatic systems. The examples are further distinguished according to the way they were designed via a so-called *Task* variable. For examples marked CD (Comparable Documents), sentences with high lexical overlap in comparable news articles were selected, whereas the hypotheses of examples marked QA (Question Answering) were formed by translating questions from e.g., TREC into statements. The other subsets are IE (Information extraction), MT (Machine Translation) RC (Reading Comprehension), PP (Paraphrase Acquisition) and IR (Information Retrieval). The different examples and subsets cover a wide variety of different aspects of entailment, from incorporation of background knowledge to lexical to syntactic entailment and combinations of all these. For a more exhaustive description of dataset design we refer the reader to (Dagan et al., 2005).

4.2 Experiment 1: Human Upper bound

To establish a human upper bound as well as investigate the validity of the datasets issued, one of the authors annotated all 800 examples of the test set for entailment, using the short RTE annotation rules. The annotation was performed before the release of the gold standard annotation for the test set and was therefore independent of the organisers' annotation. The organisers' and the author's annotation yielded a high percentage agreement of 95.25%. However, 33% of the originally created examples were already filtered out of the corpus before release by the organisers because of agreement-related problems. Therefore we expect that human agreement on textual entailment in general is rather lower.

4.3 Decision trees for entailment recognition

We expressed each example pair as a feature vector, using different subsets of the features described in Section 2 and Section 3 for each experiment. We then trained a decision tree for classification into TRUE and FALSE entailment on the development set, using the Weka machine learning tool (Witten and Frank, 2000), and tested on the test set. Apart from a classification, Weka also computes a confidence value for each decision, dependent on the leaf in the tree that the classified example falls into: if the leaf covers x examples in the training set, of which y examples are classified wrongly, then the error rate is y/x and the confidence value is $1 - y/x$.

Our evaluation measures are accuracy (acc) as the percentage of correct judgements as well as confidence-weighted average score (cws), which rewards the system's ability to assign a higher confidence score to correct judgements than wrong ones (Dagan et al., 2005): after the n judgements are sorted in decreasing order by their confidence value, the following measure is computed:

$$cws = \frac{1}{n} \sum_{i=1}^{n} \frac{\#\text{correct-up-rank-}i}{i}$$

All evaluation measures are computed over the whole test set as well as on the 7 different subsets (CD, IE, etc.). The results are summarised in Table 1. We also computed precision, recall and F-measure for both classes TRUE and FALSE and will discuss the results in the text whenever of interest.

Experiment 2: Shallow Features In this experiment only the shallow features (see Section 2) were used. The overall accuracy of 56.9% is significantly higher than the baseline.[2]

Column 2 in Table 1 shows that this decent performance is entirely due to excellent performance on the CD subset. (Recall that the CD set was designed explicitly with examples with high lexical overlap in mind.) In addition, the method overestimates the number of true entailments, achieving a Recall of 0.926 for the class TRUE, but a precision of only 0.547 on the same class. In contrast, it has good precision (0.761) but low recall (0.236) for the FALSE class. Thus, there is a correspondence between low word overlap and FALSE examples (see Example 731 in the Introduction, where important words in the hypothesis like "swamps" or "marshes" are not matched in the text); high overlap, however, is normally necessary but not sufficient for TRUE entailment (see also Example 78 in Section 3).

Experiment 3: Strict entailment To test the potential of entailment as discovered by theorem proving alone, we now use only the `entailment` and `inconsistent` features. As to be expected, the decision tree shows that, if a proof for T implies H has been found, the example should be classified as TRUE, otherwise as FALSE.[3] The precision (0.767) for the class TRUE is reasonably high: if a proof is found, then an entailment is indeed very likely. However, recall is very low (0.058) as only 30 proofs were found on the test set (for some examples see Section 3). This yields an F-measure of only 0.10 for the TRUE class. Due to the low recall, the overall accuracy of the system (0.52, see Table 1) is not significantly higher than the baseline.

Thus, this feature behaves in the opposite way to shallow lexical overlap and overgenerates the FALSE class. Missing lexical and background knowledge is the major cause for missing proofs.

Experiment 4: Approximating entailment As discussed in Section 3.5 we now try to compensate for missing knowledge and improve recall for TRUE entailments by approximating entailment with the features that are furnished by the model builder. Thus, Experiment 4 uses all eight deep semantic analysis features, including the features capturing differences in domain- and modelsizes. The recall for the TRUE class indeed jumps to 0.735. Although, unavoidably, the FALSE class suffers, the resulting overall accuracy (0.562, see Column 4 in Table 1) is significantly higher than when using the features provided by the theorem prover alone (as in Experiment 3). The confidence weighted score also rises substantially from 0.548 to 0.608. The approximation achieved can be seen in the different treatment of Example 1049 (see Section 3.5) in Experiments 3 and 4. In Experiment 3, this example

[2] We used the z-test for the difference between two proportions to measure whether the difference in accuracy between two algorithms or an algorithm and the baseline is statistically significant at the 5% level.

[3] The `inconsistent` feature was not used by the decision tree as very few examples were covered by that feature.

Table 1: Summary of Results for Experiments 1 to 6

Exp	1: Human		2: Shallow		3: Strict		4: Deep		5: Hybrid		6: Hybrid+Task	
Task	acc	cws	acc	cws	acc	cws	acc	cws	acc	cws	acc	cws
CD	0.967	n/a	**0.827**	**0.881**	0.547	0.617	0.713	0.787	0.700	0.790	**0.827**	0.827
IE	0.975	n/a	0.508	0.503	**0.542**	0.622	0.533	0.616	**0.542**	0.639	**0.542**	**0.627**
MT	0.900	n/a	0.500	0.515	0.500	0.436	**0.592**	**0.596**	0.525	0.512	0.533	0.581
QA	0.961	n/a	0.531	0.557	0.461	0.422	0.515	0.419	0.569	0.520	**0.577**	**0.531**
RC	0.979	n/a	0.507	0.502	**0.557**	0.638	0.457	0.537	0.507	0.587	**0.557**	**0.644**
PP	0.920	n/a	0.480	0.467	0.540	0.581	0.520	0.616	0.560	**0.667**	**0.580**	0.619
IR	0.922	n/a	0.511	0.561	0.489	0.421	0.567	0.503	**0.622**	**0.569**	0.611	0.561
all	0.951	n/a	0.569	0.624	0.520	0.548	0.562	0.608	0.577	0.632	**0.612**	**0.646**

is wrongly classified as FALSE as no proof can be found; in Experiment 4, it is correctly classified as TRUE due to the small difference between domain- and modelsizes for T and T+H.

There is hardly any overall difference in accuracy between the shallow and the deep classifier. However, it seems that the shallow classifier in its current form has very little potential outside of the CD subset whereas the deep classifier shows a more promising performance for several subsets.

Experiment 5: Hybrid classification As shallow and deep classifiers seem to perform differently on differently designed datasets, we hypothesized that a combination of these classifiers should bring further improvement. Experiment 5 therefore used all shallow and deep features together. However, the overall performance of this classifier (see Column 5 in Table 1) is not significantly better than either of the separate classifiers. Closer inspection of the results reveals that, in comparison to the shallow classifier, the hybrid classifier performs better or equally on all subsets but CD. In comparison to the deep classifier in Column 4, the hybrid classifier performs equally well or better on all subsets apart from MT. Overall, this means more robust performance of the hybrid classifier over differently designed datasets and therefore more independence from dataset design.

Experiment 6: Dependency on dataset design As Eperiment 5 shows, simple combination of methods, while maybe more robust, will not necessarily raise overall performance if the system does not know when to apply which method. To test this hypothesis further we integrated the subset indicator as a feature with the values CD, IE, MT, RC, IR, PP, QA into our hybrid system. Indeed, the resulting overall accuracy (0.612) is significantly better than either shallow or deep system alone. Note that using both a combination of methodologies *and* the subset indicator is necessary to improve on individual shallow and deep classifiers for this corpus. We integrated the subset indicator also into the shallow and deep classifier by themselves, yielding classifiers Shallow+Task and Deep+Task, with no or only very small changes in accuracy (these figures are not included in Table 1).

5 Related Work

Our shallow analysis is similar to the IDF models proposed by (Monz and de Rijke, 2003; Saggion et al., 2004). We have expanded their approach by using other shallow features regarding text length.

The basic idea of our deep analysis, using a detailed semantic analysis and first-order inference, goes back to (Blackburn and Bos, 2005). It is similar to some of the recent approaches that were proposed in the context of the PASCAL RTE workshop, i.e. using the OTTER theorem prover (Akhmatova, 2005; Fowler et al., 2005), using EPILOG (Bayer et al., 2005), or abduction (Raina et al., 2005).

None of these systems, however, incorporate model building as a central part of the inference mechanism. We have shown that solely relying on theorem proving is normally insufficient due to low recall, and that using model builders is a promising way to approximate entailment.

Results of other approaches to determining textual entailment indicate that it is an extremely hard

task. The aforementioned RTE workshop revealed that participating systems reached accuracy figures ranging between 0.50 and 0.59 and cws scores between 0.50 and 0.69 (Dagan et al., 2005). Comparing this with our own results (accuracy 0.61 and cws 0.65) shows how well our systems performs on the same data set. This is partly due to our hybrid approach which is more robust across different datasets.

6 Conclusions

Relying on theorem proving as a technique for determining textual entailment yielded high precision but low recall due to a general lack of appropriate background knowledge. We used model building as an innovative technique to surmount this problem to a certain extent. Still, it will be unavoidable to incorporate automatic methods for knowledge acquisition to increase the performance of our approach. Future work will be directed to the acquisition of targeted paraphrases that can be converted into background knowledge in the form of axioms.

Our hybrid approach combines shallow analysis with both theorem proving and model building and achieves high accuracy scores on the RTE dataset compared to other systems that we are aware of. The results for this approach also indicate that (a) the choice of entailment recognition methods might have to vary according to the dataset design and/or application and (b) that a method that wants to achieve robust performance across different datasets might need the integration of several different entailment recognition methods as well as an indicator of design methodology or application.

Thus, although test suites establish a controlled way of assessing textual entailment detection systems, the importance of being able to predict textual entailment in NLP might be better justified using task-based evaluation. This can be achieved by incorporating them in QA or summarisation systems.

Acknowledgements We would like to thank Mirella Lapata and Malvina Nissim as well as three anonymous reviewers for their comments on this paper. We are also grateful to Valentin Jijkoun and Bonnie Webber for discussion and Steve Clark and James Curran for help on using the CCG-parser.

References

E. Akhmatova. 2005. Textual entailment resolution via atomic propositions. In *PASCAL. Proc. of the First Challenge Workshop. Recognizing Textual Entailment*.

S. Bayer, J. Burger, L. Ferro, J. Henderson, and A. Yeh. 2005. Mitre's submission to the eu pascal rte challenge. In *PASCAL. Proc. of the First Challenge Workshop. Recognizing Textual Entailment*.

P. Blackburn and J. Bos. 2005. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI.

J. Bos, S. Clark, M. Steedman, J. Curran, and J. Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proc of COLING*.

K. Claessen and N. Sörensson. 2003. New techniques that improve mace-style model finding. In *Model Computationa - Principles, Algorithms, Applications (Cade-19 Workshop)*, Miami, Florida.

I. Dagan, O. Glickman, and B. Magnini. 2005. The pascal recognising textual entailment challenge. In *PASCAL. Proc. of the First Challenge Workshop. Recognizing Textual Entailment*.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

A. Fowler, B. Hauser, D. Hodges, I. Niles, A. Novischi, and J. Stephan. 2005. Applying cogex to recognize textual entailment. In *PASCAL. Proc. of the First Challenge Workshop. Recognizing Textual Entailment*.

H. Kamp and U. Reyle. 1993. *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Dordrecht, Netherlands.

C. Manning and H. Schuetze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

C. Monz and M. de Rijke. 2003. Light-weight entailment checking for computational semantics. In *Proc. of ICOS-3*.

R. Raina, A.Y. Ng, and C. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proc. of AAAI 2005*.

A. Riazanov and A. Voronkov. 2002. The design and implementation of Vampire. *AI Comm.*, 15(2-3).

H. Saggion, R. Gaizauskas, M. Hepple, I. Roberts, and M Greenwood. 2004. Exploring the performance of boolean retrieval strategies for open domain question answering. In *Proc. of the IR4QA Workshop at SIGIR*.

I. H. Witten and E. Frank. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Diego, CA.