Inference for Natural Language

Amal Alshahrani
School of Computer Science
Universit of Manchester
Manchester M13 9PL, UK
amal.alshahrani@postgrad.manchester.ac.uk

Allan Ramsay
School of Computer Science
Universit of Manchester
Manchester M13 9PL, UK
Allan.Ramsay@manchester.ac.uk

Abstract

The main aim of this study is to develop a natural language inference (NLI) engine that is more robust than typical systems that are based on post-Montague approaches to semantics and more accurate than the kinds of shallow approaches usually used for textual entailment, The term robustness is concerned with processing as many inputs as possible successfully, and the term accuracy is concerned with producing correct result. In recent years, several approaches have been proposed for NLI. These approaches range from shallow approaches to deep approaches. However, each approach has a number of limitations, which we discuss in this paper. We argue that all approaches to NLI share a common architecture, and that it may be possible to overcome the limitations inherent in the existing approaches by combining elements of both kinds of strategy.

1 Introduction

In order to understand natural language, we need to know a lot about the world and be able to draw inference (Ovchinnikova, 2012). For instance, to answer the query "Was Shakespeare the author of Romeo and Juliet?" from the following text: *"Romeo and Juliet* is one of Shakespeare's early *tragedies.* The *play* has been highly praised by critics for its language and dramatic effect" we need background knowledge such as: (i) Tragedies are plays. (ii) Shakespeare is a playwright; playwrights write plays. (iii) Plays are written in some language and have dramatic effect.

Hence without background knowledge, answering the query would be impossible.

Tackling this task will open the door to applications of these ideas in various areas of Natural Language Processing (NLP) (Dale, Moisl and Somers, 2000) such as question answering (QA), information extraction (IE), summarisation, and semantic search.

Many approaches have been suggested in the literature to achieve this goal. These approaches can be divided into two groups:

Shallow approaches, which are based on lexical overlap, pattern matching, distributional similarity and others (Dagan and Glickman, 2004).

These approaches have a number of limitations and difficulties. In particular,

- They may not take semantic representation into account.
- They may not be sound.
- They cannot easily make use of complex background knowledge.

Deep approaches, which are based on semantic analysis, lexical and world knowledge, logical inference and others (Blackburn et al., 2001).

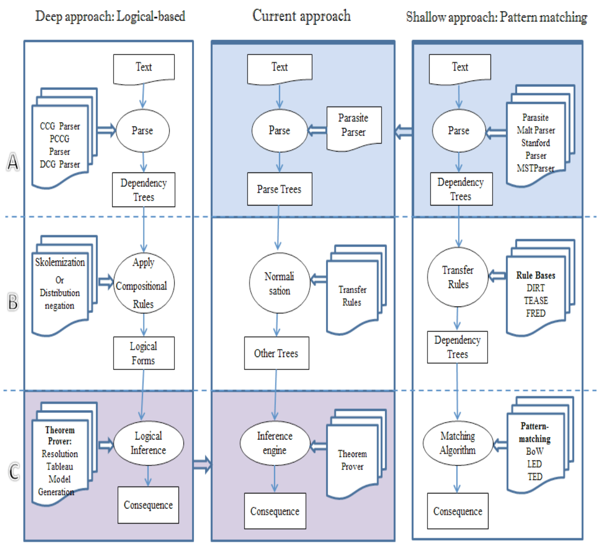
These approaches have a number of limitations and difficulties For instance,

- Compositional translation to logical form requires syntactic analysis which conforms to a grammar expressed as a set of rules. Such analyses are very hard to obtain for freely occurring texts.
- For complex sentences, logical forms often turn out to be extremely verbose, and hence are difficult for standard theorems provers to handle.
- Vast amounts of additional knowledge are required.

This kind of deep approach can succeed in restricted domains, but it fails badly on open domain problems.

2 Proposed system

It is widely assumed that shallow and deep approaches of NLI have completely different structure (MacCartney, 2009). However, if you look at the left and right-sides of Figure (1), you can see that at a very gross level of abstraction they can be decomposed into the same three major steps. They start with a pre-processing stage (stage A) which analyses the syntactic structure of input as some kind of parse tree. Then the second step (stage B) is responsible for normalising these trees to some format that is suitable for the intended inference engine. Finally, the inference engone (stage C) is responsible for comparing the representations obtained by stage B to see what follows from what was said.



Deep approach: Logical-based
Current approach
Shallow approach: Pattern matching
A
B
C
Text
CCG Parser PCCG Parser DCG Parser
Parse
Dependency Trees
Skolemization Or Distribution negation
Apply Compositional Rules
Logical Forms
Theorem Prover: Resolution Tableau Model Generation
Logical Inference
Consequence
Text
Parse
Parasite Parser
Parse Trees
Normali sation
Transfer Rules
Other Trees
Inference engine
Theorem Prover
Consequence
Text
Parse
Parasite Malt Parser Stanford Parser MSTParser
Dependency Trees
Transfer Rules
Rule Bases DIRT TEASE FRED
Dependency Trees
Matching Algorithm
Pattern-matching BoW LED TED
Consequence

Figure 1: System Architecture.

The differences between the left- and right-hand sides of Figure 1 are that the stage C of the deep approach utilises a standard theorem prover for first-order logic (or some extension thereof), and hence requires stage B to produce formulae of the relevant logic on the basis of the trees produced by stage A. It is, however, extremely difficult to produce such formulae from freely occurring texts, since most parsers that are robust enough to handle texts such as newspaper articles or Wikipedia pages rely on implicit rules that have been extracted from corpora, and it is somewhere between difficult and impossible to attach compositional rules to such inferred parsing rules. Shallow approach are less ambitious about the degree of normalisation that can be achieved, but as a consequence the inference engines that they depend on are less powerful. The goal of the current proposal is to use an adaptation of a standard theorem prover, but to apply it directly, or almost directly, to the dependency trees obtained by the parser.

2.1 Stage A: Structural Analysis

This stage represents the pre-processing of the current system. It is responsible of converting input sentences from natural language expressions into dependency trees. To achieve this goal, we use the PARASITE parser (Ramsay, 1999; Seville and Ramsay, 2001). The advantage of using an in-house parser is that it allows some measure of control over the shape of then output trees—that if, for instance, we believe that it is better for the auxiliaries in a verb chain to be the head of the chain then we can arrange it so that our trees have this shape; and if we decide that the contrary is the case, then we can easily make the change. Controlling the underlying structure of the grammar obviates the need for subsequent transformations during the second stage of the process—to take another example, making the determiner the head of an NP might make sense from the point of view of the inference engine, so if we have control over that decision during the parsing process then we will not have to do anything about it during normalisation.

2.2 Stage B: Normalisation

In any NLI system, the output of the initial structural analysis is likely to produce structures that are not well-matched to the intended inference engine. This is clear for deep approaches, where a considerable amount of machinery is required for transforming parse trees into logical forms, but it is also true for shallow approaches: Alabbas & Ramsay (2012), for instance, showed that induced dependency parsers work better if the head of the first element of a coordinated expression is taken to be the head of the whole coordinated expression, but almost all approaches to inference require the head of such an expression to be the conjunction itself. It is therefore nearly always necessay to carry out some post-processing of the trees produced by the parser before carrying out the third stage of the overall task. In the following sections we describe three such normalisation techniques.

Shallow normalisation

Normalisation in shallow approaches is typically involves producing abstract 'entailment templates' from sets of sentence pairs, where common element of the two sentences in a pair are replaced by variables (Kouylekov and Magnini, 2005).

Numerous systems have been suggested for automatic acquisition of rules, ranging from distributional similarity to finding shared contexts such as *DIRT*[1] (Lin and Pantel, 2001), *TEASE*[2] (Szpektor et al., 2004), and *MSR Paraphrase Corpus (Dolan et al., 2004)*. For example, the normalisation for the sentence (‘*X solves Y*’ implies ‘*X finds a solution to Y*’), which is (Templates with variables) is illustrated in Figure 4.

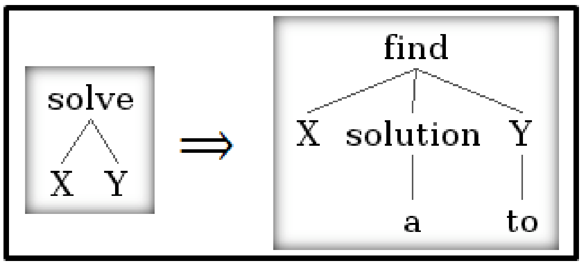


Figure 4: Normalise the sentences ‘*X solves Y ⟹ X finds a solution to Y*’.

Deep normalisation

Normalisation in deep approaches is defined as translation of natural language expressions into formal meaning representations (logical form) (Blackburn et al., 2001). There are a lot of systems available such as conversion to clausal form (Lukasova et al., 2012), Skolemisation (Degtyarev, Lyaletski, and Morokhovets, 1999), distribution of negation and others. For instance, *(John solves the problem → John finds a solution to the problem)*.

The normalization for the previous sentences is:

$\forall x,y(solve(x,y) \Rightarrow \exists z(find(x,z) \land solution(z) \land to(z,y))$

Our normalization

In our normalization we translate a form of a natural language into a restricted subset of the same natural language,

In our case the first form is a dependency tree, obtained from the parser in stage (A). Such a tree may not be ideal for using with the theorem prover in stage (C). We therefore have to normalise such trees in order to adapt them for use with our chosen theorem proving strategy. Exactly what normalisation is required depends on the nature of the theorem prover. For example in Figure 5 we use the dependency tree in figure 5(b’) to obtain a subset of dependency tree by converting into the form (LHS → RHS) as in Figure 5(c’) and Figure 5(d’), using the rules in Figure 6(b, c, and d). Then in Figure 7&8 we simplified the sub-tree (c) and (d) to obtain the last version of sub-tree (e) & (f) as required for using with our theorem prover.

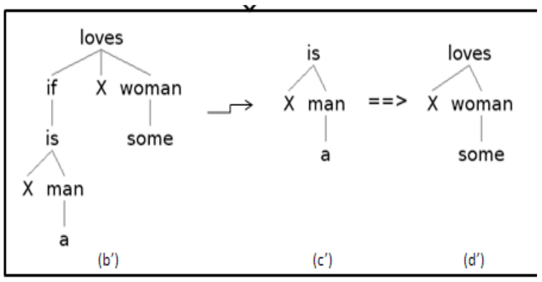


Figure 5: Convert the sentence into the form (LHS (c’) → RHS (d’)).

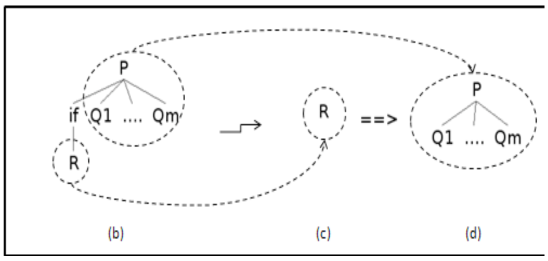


Figure 6: Rule for converting the sentence into the form (antecedent (c) → consequent (d)).

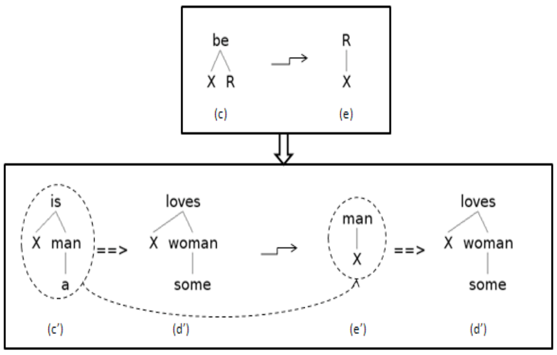
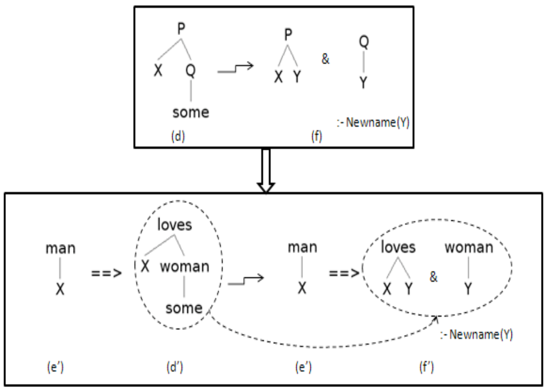


Figure 7:Simplifed subtree(c’) to subtree (e’) by applying the rule in (c) & (e).



[1] “Discovery of Inference Rules from Text”.
[2] “Textual Entailment Anchor Set Extraction”.

Figure 8:Simplified subtree(d') into sub-tree(f') by applying the rule in (d) & (f).

2.3 Stage C: Inference Engine

The key to the current proposal is the observation that the central step in almost all current theorem provers, namely that given two sequents/clauses A_1 & ... & A_{n1} ==> C_1 *or* ... *or* C_{m1} *or* X and X' & A'_1 & ... & A'_{n2} ==> C'_1 *or* ... *or* C'_{m2} where X and X' can be unified by some unifier σ, you can 'cut' X to obtain σ (A_1 & ... & A_{n1} & A'_1 & ... & A'_{n2} ==> C_1 *or* ... *or* C_{m1} *or* C'_1 *or* ... *or* C'_{m2}).

There are numerous ways of invoking this rule: the key is that for the vast majority of theorem provers this rule is the core of the process.

It is worth noting that the elements of a rule need not be expressions of some formal logic. They usually are, but there is no a priori reason why they should be. They could, for instance, be the rules of a game: a program for playing chess might exploit rules which describe legal board transformations, a program for finding routes might exploit rules which describe links between places, … In particular, they might be dependency trees.

This is clear enough for simple rules: if we allow natural language utterances to contain variables, then we can easily write rules like

X and Y used to be married if X and Y have got divorced

Rules like this can easily be applied using the standard rule of cut mentioned above. More interestingly, we can also us it to apply essentially higher-order rules such as

P are not Q if P used to be Q

We have previously shown how to extend SATCHMO (Manthey & Bry 1988) to cover intensionality (Ramsay 2001). The same machinery can be exploited to handle higher rules of the kind shown, which is crucial for handling natural language, where intensionality, type-shifting and other higher-order notions are rife.

We also intend to generalise the conditions under which cut applies. The standard rule requires X and X' to unify. Within the current framework, X and X' are trees. As such, we can use approximate matching, e.g. allowing X' to be subset of X to allow for the deletion of modifiers, or by allowing the terms that appear in X' to denote subsets of the corresponding terms in X.

These two moves will allow us to work directly with dependency trees, without making any assumptions about where these trees came from. We can thus avoid the need to translate into some target formal language: if some element of the antecedent of one rule matches an element of the consequent of another, subject to whatever constraints we put on the matching process, then we can use the rule.

3 Conclusion

We have proposed a strategy for carrying out inference over natural language sentences by applying standard theorem proving technology directly to dependency trees. This circumvents the need to translate from parse trees, of whatever kind, to formal logic, which has proved challenging for over forty years. It does introduce two risks: that the inference chaijns will become unsound, and that inference will become very slow. The first of these can be moderated by varying the conditions under which a partial match is allowed: if only exact matches are allowed, then there will be no risk, but rules which are potentially relevant may be missed, and as more flexibility in the matching process is permitted there will be more chance of mistakes but wider coverage. Similarly, if subtrees are only matched if they are term-unifiable then there should be no loss of speed, and as the conditions for matching are relaxed the process will become slower but more flexible.

Acknowledgements

Allan Ramsay's contribution to this work was supported by Qatar National Research Foundation grant NPRP 09-046-6-001. Amal Alshahran is supported by a grant from the government of the Kingdom of Saudi Arabia.

References

Alabbas, M & Ramsay A M, Arabic Treebank: from Phrase-Structure Trees to Dependency Trees, META-RESEARCH Workshop on Advanced Treebanking, LREC 2012, Istanbul, 61--68, 2012

Blackburn, P., Bos, J., Kohlhase, M., & De Nivelle, H. (2001). Inference and computational semantics. *In Computing Meaning* (pp. 11-28). Springer Netherlands.

Dagan, I. and Glickman, O. (2004). *Probabilistic textual entailment: generic applied modeling of language variability*. In Proceedings of the PA CAL Workshop on Learning Methods for Text Understanding and Mining, pp. 26–29, Grenoble, France.

Dale, R., Moisl, H. L., & Somers, H. L. (Eds.). (2000*). Handbook of natural language pr cessing. CRC Press*.

Degtyarev, A. I., Lyaletski, A. V., & Morokhovets, M. K. (1999, January). *Evidence algorithm and sequent logical inference search*. In Logic for Programming and Automated Reasoning (pp. 44-61). Springer Berlin Heidelberg.

Dolan, W. B., Quirk, C., and Brockett, C.. 2004. *Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources*. In Proceedings of COLING 2004.

Kouylekov, M., & Magnini, B. (2005, April). *Recognizing textual entailment with tree edit distance algorithms*. In Proceedings of the First Challenge Workshop Recognising Textual Entailment (pp. 17-20).

Lin, D. and Pantel, P. (2001). *DIRT-discovery of inference rules from text.* In Proceedings of the 7th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 323–328, San Francisco, California, USA. doi:10.1145/502512.502559.

Lukasova, A., Zacek, M., Vajgl, M., & Kotyrba, M. (2012). *Resolution reasoning by RDF Clausal Form Logic*. International Journal of Computer Science, 9.

MacCartney, B. (2009). *Natural language inference* (Doctoral dissertation, Stanford University).
Manthey, R., & Bry, F. (1988, January). *SATCHMO: a theorem prover implemented in Prolog*. In 9th International Conference on Automated Deduction (pp. 415-434). Springer Berlin Heidelberg.

Ovchinnikova, E. (2012). *Integration of world knowledge for natural language understanding* (Vol. 3). Springer.

Ramsay, A.M., Theorem proving for untyped constructive lambda-calculus: implementation and application, *Logic Journal of the Interest Group in Pure and Applied Logics,* 9(1), 89-106, 2001

Ramsay, A.M., (1999). *Parsing with discontinuous phrases.* Natural Language Engineering, 5(3):271–300,doi:10.1017/S1351324900002242.

Seville, H. and Ramsay, A. (2001). *Capturing sense in intensional contexts*. In Proceedings of the 4th International Workshop on Computational Semantics, pp. 319–334, Tilburg, The Netherlands.