Extracting Logical Forms and Recognising Textual Entailment

2007-01-05

Cenny Wenner
Lund University, Sweden
cwenner@gmail.com, cennywenner.com
Ulrikedalsvägen 2U Lg. 216, Lund

Abstract

We present concise high-accuracy rules for Dependency-based Logical Forms[1] extraction and utilization for Recognising Textual Entailment. We will enumerate the most essential rules and supply a comparison of their respective impacts on the 2004 Senseval 3[2] Identifying Logical Forms task. We will also present the rudimentary of a system for Recognising Textual Entailment benchmarked on the data of the first PASCAL[3] Recognising Textual Entailment challenge[4].

1 Introduction

Recognising Textual Entailment is believed to be a central subproblem of many NLP tasks involving understanding such as Question Answering, Machine Translation, Paraphrasing and Information Extraction. Neither Dependency-Graph based Logical Forms Extraction or a Logical Inference approach to Recognising Textual Entailment is avant-garde. (Ahn et al., 2004) and (Anthony and Patrick, 2004) employed the Dependency-Graph technique at Senseval 3 and Logical Inference has been used by for instance (Ferrandez et al., 2006), (Bos and Markert, 2006) and (Tatu et al., 2006). We present a more thorough exposition of the systems and an integration that covers all parts from plain text to inference on it's meaning. We list the few simple rules we have used to achieve good results. In section 3, we will present the Logical Forms aspect of the system and in section 4 the Textual Entailment aspect. This will be followed by conclusions and a discussion in sections 5 and 6. Section 3.1 introduces the Logical Forms aspect, section 3.2 presents the processing done by external modules to produce Dependency Graphs from English text and section 3.3 our implementation for further processing and the associated rules. In section 3.4, we will present the importance of most of these rules.

[1]The Logical Forms here referred are advocated by Vasile Rus (Rus, 2002) and the Senseval 3 task.

[2]http://www.senseval.org/senseval3.

[3]Pattern Analysis, Statistical Modelling and Computational Learning, http://www.pascal-network.org .

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

2 Previous Work

Previous work advocated up to a hundred rules and only achieved only slightly better than our system (see for instance (Bayer et al., 2004) and (Ahn et al., 2004)). Unfortunately, little is said about the actual rules those systems employed and which ones are the most essential. We here aim to give a concise overview that will allow the readers to swiftly implement their own systems. As mentioned in the introduction, (Ahn et al., 2004) and (Anthony and Patrick, 2004) have employed Dependency-Based logical forms identification. Aside from the Logical Inference approach to Textual Entailment, there are many other, most which rely on statistical models or sentence/parse-tree similarity. See (Dagan et al., 2005a) or (Bar-Haim et al., 2006) for an thorough overview.

3 Extracting logical forms

3.1 System overview

Our system features five modules for extracting logical forms: Sentence splitter, Tokenizer, PoS-Tagger, Dependency Graph Parser and Morphological Parser. For these challenges, the first module isn't necessary and the second may be handled by the Dependency Graph Parser alone. Our system use naive hard-coded rules for the first two modules. The Morphological Parser is required to identify base forms as required by the Rus' Logical Forms.

3.2 PoS-tagging and Dependency Graph Parsing

The first step in the Logical Forms Identification is to tag each token in each sentence with a Part-of-Speech. To do this, you may need to split a text into sentences and tokenize each. To tokenize, one does well only by identifying each sequence of word-characters with the exception of those with an "n't" or "'*" suffix. As PoS-tagger we have tried the Stanford Log-linear PoS tagger[5] (Toutanova and Manning, 2000) (Toutanova et al.,

[5]http://nlp.stanford.edu/software/tagger.shtml

2003), MXPOST[6] (Ratnaparkhi, 1996) and TreeTagger[7], in order of decreasing accuracy and delay. For our purposes, MXPOST yielded the best trade-off between accuracy and speed during the development phase.

The second step is to build a Dependency Graph. Some Dependency Graph Parsers don't require the first step and the PoS-tagging may be omitted. We have used MaltParser 0.4 EngSVM[8].

Processing beyond Dependency Graphs is the focus of this essay and described in the next section.

3.3 Predicate-Argument Extraction

3.3.1 Predicate introduction

The Logical Forms are extracted through three passes over the Dependency Graph. The first pass instantiates predicates and arguments for appropriate PoS-tags. For noun groups, __NN-nodes are also created with an arity of three. PoS-tags are reduced from the Penn-Treebank tags (Santorini, 1991) to the following categories:

Nouns (arity 1) Nouns, singular, mass (NN), plural (NNS), proper singular (NP), plural (NPS) and Noun Groups (here introduced as __NN) (arity 3).

Verbs (arity 3) All V*-tags. There may be optional arguments to verbs besides the mandatory 3, see (Rus, 2002).

Conjunctions (arity 3) Coordinating Conjunctions (CC) and Interjections (UH)

Adjectives (arity 1) Adjectives (JJ), Comparative Adjectives (JJR) and Superlative Adjectives (JJS)

Adverbs (arity 1) Adverbs (RB), Comparative Adverbs (RBR) and Superlative Adverbs (RBS)

Determiners (ignored) Determiners (DT)

Auxiliary verbs (ignored) Modals (MD)

Unary modifiers (arity 1) Possessive Pronoun (PP$)

Binary modifiers (arity 2) To (TO), Prepositions and Subordinating Conjunctions (IN)

Ignored Punctuation (SENT), Wh-pronouns (WH), various symbols including coma.

3.3.2 Argument Identification

The remaining passes are used to identify equalities between arguments and determine whether arguments are events or entities. If not proved otherwise arguments are considered to be unequal entities. If an acceptable node is found, an equality is introduced between the original node's argument that was being processed and the first argument of the found node. The first argument is considered to be an identifier of the event or entity. For noun groups, conjunctions and verbs, a search is done for the second and third argument. For verbs, for instance, nearby nodes are searched to find suitable nodes to use as subject and object. If a noun-group node would be found, the first argument, the argument that identifies the entity, is used. Modifiers, adjectives and adverbs also search for their first argument.

Conjunctions, adjectives and adverbs restrict their attention to either events or entities:

Conjunctions Second and third argument both events or both entities

Adjectives First argument entity

Adverbs First argument event

For this reason, nodes that do not rely on whether arguments are events or not are processed in the second pass and those that do in the third.

During these two passes, every node's arguments may be processed to find up to one other argument of a nearby node to introduce an equality with. The search for arguments that fulfill certain requirements is done in a certain order which is given by the PoS of the node itself, it's parent and, in the case when more than one argument is necessary, the PoS-tag of the first argument. There are four groups of nodes that are searched:

Parent Parent of the node in the Dependency Graph.

Pre-sibblings Sibblings in the Dependency Graph that appear before the node itself in the original text. These are processed in reverse order of appearance.

Post-sibblings Sibblings in the Dependency Graph that appear after the node itself in the original text. These are processed in order of appearance.

Children Children of the node in the Dependency Graph, in order of appearance.

The standard order of processing is *Pre, Child, Post* where the parent is inserted after *Pre* if it isn't a noun and after *Post* otherwise. If the first argument belongs to the pre-sibblings, the second argument is instead searched for in the order *Child, Post, Pre*, with the parent inserted as usual. We will call this *rotated order* here. Certain nodes has special rules as listed below.

PoS-tags	Constraint
__NN	Only nouns
Verbs, 2^{nd} arg.	SUB function preferred*
Verbs, 3^{rd} arg.	OBJ/VC function preferred*
Conjunctions	Both events or both entities**
Adjectives	Only entities
Adverbs	Only events
Unary Mod.	Rotated order***
Binary Mod.	Rotated order for both***

* if an acceptable function of the kind exists, that node is used. If none exist however, ordinary search is done.
** this ended up having no effect on the test set.
*** these rules gave a higher score on the development set but not on the test set.

[6] ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz
[7] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger
[8] http://w3.msi.vxu.se/ nivre/research/MaltParser.html

3.4 Rule impacts

The impact on the argument F-Score for alterations to the described system. The changes are absolute, not relative.

No verb functions	-6.5%
No conj. constraint	+-0.0%
No mods. rotation	+0.1%
No rotations	-3.4%
No pre first case	-0.2%
Always pre first case	-0.1%
No noun parents	-1.1%

3.5 Results

Accuracy at the Logical Form Identification of Senseval 3 is primarily measured by the F-Score on Predicate Level and F-Score on Argument Level. The systems at Senseval 3 achieved an Argument F-Score between 0.694 and 0.776 and a Predicate F-Score between 0.801 and 0.892. We report an Argument F-Score of 0.642 and a Predicate F-Score of 0.841 on the genuine test set. We therefor did slightly worse than the systems at Senseval 3 on the Argument level but score high on the Predicate Level.

4 Recognising Textual Entailment

4.1 Overview

A common strategy for RTE, especially in the original PASCAL's RTE 1 challenge is Logical Inference. Either a proof of the hypothesis should be found given the text or alternatively the absence of a proof of the negation of the hypothesis. Our logical inference system takes the previously described Logical Forms as input and attempts to prove the hypothesis given the text using optional minor linguistic rules and background knowledge. The proofs are found through First Order Logic Backward Chaining (see (Russell and Norvig, 2003) p. 287-295).

To handle ambiguity, each predicate is assumed to be either of the senses (synsets) listed in WordNet[9] (Fellbaum, 1998). For tokens without WordNet entries, a new predicate unique to the token text is created. In practice, this means assuming that all senses of every predicate in the antecedent is true and the goal is to prove that there is an assignment of senses to the hypothesis predicates so that it is inferable.

4.1.1 A simple proof knowledgebase

Foundational linguistic rules

$$\forall_{p,a,v,s} text(p,a,v) \wedge s \in senses(p) \Rightarrow kb(s,a,v) \quad (1)$$

$$\forall_{p,a,v,s} kb(s,a,v) \wedge s \in senses(p) \Rightarrow hypo(p,a,v) \quad (2)$$

WordNet senses and hypernym relations

$$senses(cat) = \{n3414, n10085, \ldots\} \quad (3)$$

$$senses(animal) = \{n5934, n6136, \ldots\} \quad (4)$$

$$\forall_{a_1,a_2} kb(n3414, a_1, T) \Rightarrow kb(n5934, a_2, T) \quad (5)$$

[9] http://wordnet.princeton.edu/

Antecedent and Hypothesis

$$text(cat, [x1], T) \quad (6)$$

$$hypo(animal, [x1], T) \quad (7)$$

4.2 Knowledgebase

Besides WordNet relations, so called semantic relationships, Linguistic/NLP rules and other Real-World Background Knowledge may be necessary to find more complicated proofs. Linguistic rules are used to intepret the structure of the Logical Forms. "Dogs and cats fight" would for instance yield the conjunction of the two nouns as the subject of the event and linguistic rules would be needed to draw the separate conclusion that "dogs fight". Background Knowledge would include information about how the world is. Something that includes the so called common sense. An example would be that when a man buys a boat, the man gets in his possession the very same boat.

4.3 Results

We only achieved an accuracy of 52.0% on the RTE 1 development set. This is very poor in comparison with the high-end full-coverage systems which achieved up to 58.6% (Bayer, MITRE and Glickman, Bar Ilan) (Dagan et al., 2005b). The precision of this accuracy was 85.7% while the recall was 4.2%.

5 Conclusions

It appears as though we can conclude that transforming Dependency Graphs to Rus' Logical Forms with high accuracy is possible through a few simple rules. Previous work in the area (Ahn et al., 2004) confirms these results, claiming the transformation to be "straight-forward". We would like to complement that implementing such a system is anything but complex or time-consuming, given modules mentioned above. On Recognising Logicsal Inference we find the problem more complex. The initial inference engines were too slow to discover any but the simplest proofs. With an optimized low-complexity engine, considerably better results are achieved but they are still not comparable to the high-end systems of RTE 1 which achieve up to four times as good improvements compared to us, on top of the Majority Model accuracy (Dagan et al., 2005a).

The strength of Logical Inference is it's high precision but the mean accuracy remains low, something that could also be observed in RTE 2 (Bar-Haim et al., 2006). Some of the best systems sacrifice some of the precision to improve the recall, performing various approximations.

6 Discussion

Although the logical forms accuracy is rather high, much improvement is necessary for higher modules due to the propagating errors. The greatest problem for us for the Recognising Textual Entailment problem, that we are aware of, is the exponential growth of possible proofs

in the depth of the proof (whether forward-chaining, backward-chaining or resolution is applied). Due to the high branching factor produced by possible senses or linguistic rules, no greater than intermediate length hypotheses can be solved by our engine. We early noticed how the highest accuracy was achieved with the simplest set of inference rules. This is explained by how, even with a heuristically guided search, not even the simple proofs might be found with a too great branching factor and that the set of newly solvable pairs small in comparison.

7 Future Work

We have noticed the impact of the accuracy of low-level modules, such as PoS-tagging. Enhancement of these modules should and probably essential for accuracies beyond 97% on the Logical Forms Extraction. It appears fruitful for the Textual Entailment task as well as we frequently notice that proofs are disrupted by single errors. Collocations aren't handled and would probably yield another % both on predicates and arguments. Determiners and auxiliary verbs are ignored by both the logical forms and the logical inference. Natural Language appears especially ambiguous at this point but linguistic rules for handling these cases should bring up the logical inference accuracy slightly. A sense disambiguation system could be included to reduce the search space significantly and thereby lend more resources to feasible domains. Machine Learning techniques appears to work well for both tasks and should complement the high-precision, low-recall logical inference well (although previous work (Bos and Markert, 2006) have so far achieved opposite results where for logical inference, machine learning techniques outperform logical inference rather than complementing it). A more bold idea would be approximative inference for merging similar proof branches. This might be able to reduce the branching factor significantly.

References

David Ahn, Sisay Fissaha, Valentin Jijkoun, and Maarten De Rijke. 2004. The university of amsterdam at senseval-3: Semantic roles and logic forms. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 49–53, Barcelona, Spain, July. Association for Computational Linguistics.

Stephen Anthony and Jon Patrick. 2004. Dependency based logical form transformations. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 54–57, Barcelona, Spain, July. Association for Computational Linguistics.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Samuel Bayer, John Burger, John Greiff, and Ben Wellner. 2004. The mitre logical form generation system. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 69–72, Barcelona, Spain, July. Association for Computational Linguistics.

Johan Bos and Katja Markert. 2006. When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005a. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Ido Dagan, Bernardo Magnini, and Oren Glickman. 2005b. The pascal recognising textual entailment challenge. In *Proceedings of Pascal Challenge Workshop on Recognizing Textual Entailment*.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. pub-MIT.

O. Ferrandez, R.M. Terol, R. Munoz, P. Martinez-Barco, and M. Palomar. 2006. An approach based on logic forms and wordnet relationships to textual entailment performance. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, Venice, Italy, April.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics, Somerset, New Jersey.

Vasile Rus. 2002. Logic forms for wordnet glosses.

St. Russell and P. Norvig. 2003. *Artificial Intelligence: A Modern Approach, Second Edition, International Edition*. Prentice Hall International Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, NJ. RUS st 03:1 1.Ex.

Beatrice Santorini. 1991. Part-of-speech tagging guidelines for the penn treebank project.

Marta Tatu, Brandon Iles, John Slavick, Adrian Novischi, and Dan Moldovan. 2006. Cogex at the second recognizing textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Kristina Toutanova and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, Hong Kong.

Kristina Toutanova, Dan Klein, and Christopher D. Manning. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 03*.