Overview of the Task and What It Involves

Research Memo #2
Iddo Lev
December 31, 2005

1 The Goals

1.1 Understanding

In my opinion, the ultimate goal of NLP is to create a computer that really *understands the meaning and information* conveyed by a Natural Language (NL) text or speech. From a practical point of view, we would like computers to read and understand information in texts so that they could give correct answers to questions about that information, learn from this information, and manipulate it in other ways.[1] From a scientific perspective, if a computer is able to understand a class of texts correctly, knowing how it does it might shed some interesting light on how humans do it.

We need to define what is meant by "understanding the meaning and information in a text." We take the operationalized definition: the ability to draw conclusions from the text and answer questions about it in the same way that humans would given the same input. Since humans do not always agree about the inferences and answers to questions from a given NL text, we concentrate our research here on those cases where most people do agree, cases that are quite clear cut.

1.2 Task Definition

We would like to test the computer's level of understanding of a text, as defined above, by giving it a NL text T and a NL sentence S and asking it whether S is a conclusion from T. More generally, instead of S we may give a NL question Q, and ask the computer to produce as output a NL answer A. We would then check whether A is a correct answer to the question Q in light of the information given in T (we call the pair (T, Q) a *query*). As said above, we restrict ourselves to cases where all (or almost all) humans would agree what the correct answer is.

If Q is a yes/no question, the answer may be *Yes* or *No*, and if Q is a wh-question, the answer may be a value. But other possible answers are:

- *Unknown*: the computer *knows* that the answer cannot be determined from the text. For example, when T is "John likes Mary" and Q is "Does Mary like John?"

- *AssumptionFailure*: the computer *knows* that the question relies on a false assumption. For example, a person asking "Does John realize that Mary likes him?" presupposes that Mary likes John. If this question is given to the computer together with a text that does not support this presupposition, such as

[1] For simplicity, I talk here about NL texts, but the ideas carry over (perhaps with some required changes) to other forms of NL input.

the text "John likes Mary", the proper answer is "AssumptionFailure: the fact 'Mary likes John' is not supported by the text". An answer *No* might be misleading as it might lead the user to conclude that the sentence "John does not realize that Mary likes him" is supported by the text, which is also not the case.

- *Ambiguous*: the computer *knows* that there is more than one correct answer to the query due to some ambiguity in the text or the question. For example, if the text includes "John saw her duck" and the question is "Did John see a duck?", the answer may be *Yes* or *Unknown*, depending on the syntactic analysis of the text (i.e. whether *her* is analyzed as a pronoun or a possessive pronoun). If the intended analysis is not clear from the text, the answer should be *Ambiguous*. In addition, it would be useful if the computer pointed out the ambiguities and even gave all possible answers. (See more in section 1.4)

- DontUnderstand: the computer *knows* that understanding the text or the question is beyond the scope of its abilities. An example is when T is "John considers Mary a nice woman" and the computer's knowledge of language does not include knowledge of how to understand reduced clauses. In addition, it would be useful if the computer specified what parts of the input it did and did not understand.

- Fail: the computer could not reach any of the definite answers mentioned above because of lack of computational resources or some other problems.

1.3 Precision and Certainty

Our goal in this research is to have the computer produce an answer with certainty and not by guessing. As the list of possible answers above show, we want the computer to *know* when it cannot answer a query as a result of it not having the necessary knowledge to do so. In short, we want precise understanding.

Other works on question answering do not follow this goal, and instead make guesses about the answer. For example, Pasca and Harabagiu (2001) use a statistical parser for assigning a syntactic analysis to the question and text. As is always the case with such parsers, the parser sometimes selects the wrong syntactic analysis, and sometimes produces an output which is not a legal syntactic analysis at all (this happens especially when the sentence includes a syntactic phenomenon which the parser did not encounter at all in its training corpus). Moreover, there is no attempt to represent the meaning and information that are conveyed by the text and the question. Instead, some heuristic graph matching procedure is applied on the guessed syntactic structures of the question and the text, including alterations at the word level based on synonyms from WordNet. The benefit of such a method is that it is "robust" in the sense that it always produces some answer, and more often than not, the answer happens to be the correct one.

Almost all work on question answering today follows this direction because of the desire to deal in some way with arbitrary NL texts on arbitrary topics, so we need to explain why we do not also follow this direction. There are several reasons. First,

since most people are already working on that direction, there would not be much added value if we also worked on that direction. In contrast, since the direction of precise understanding of texts is largely neglected, there is a far greater need for people to work on that.

Second, "robust" question answering is suitable at most for questions whose shape is a minor variation on the shape of a one-sentence factoid answer that appears explicitly in the text. In contrast, if one is interested in really understanding the information and meaning conveyed by the text for the purpose of combining pieces of information from different parts of the text and drawing new conclusions from it, methods with a much higher quality and precision are necessary. Most interesting NLP applications, both today and in the long term, would need such an ability.

Third, work on precise understanding, a long-term basic research initiative, aims to develop high-quality domain-independent linguistic knowledge. Such knowledge could be used in a large variety of applications, including guiding the research on "robust" understanding by improving its precision and quality, as well as aiding in knowledge acquisition. For more on these points, see Research Memo #1.

1.4 Ambiguity

It is often claimed that NL input is riddled with lots of ambiguities. This is true in the technical sense that if an NL unit, such as a sentence, is taken out of context, it could have multiple meanings. Nonetheless, if the context of utterance of an NL input is taken into account, including the purpose of the utterance by the speaker and the common knowledge that is assumed, the number of truly ambiguous NL inputs is quite small (otherwise, people would constantly be baffled by what others are saying).

We are aiming here to take into account as much of the context and background knowledge as is necessary in order to reduce technical ambiguities. In the few remaining cases where a text or a question is truly ambiguous in context, the computer should notify the user about this by producing the output *Ambiguous* (see section 1.2). In other words, the computer should answer in the same way as humans would.

1.5 Assumed Knowledge

High-quality understanding of NL requires a lot of knowledge about what is assumed in the text. In the research here, we focus mainly on the knowledge of language, and we rely on other people's work (existing or in the future) on world knowledge representation and reasoning. We do so merely because working on the knowledge of language is a huge task in itself, and one needs to cut out a reasonable chunk to work on. For a justification of this direction, see Research Memo #1.

2 What Is Involved

To achieve the goals above, we need to address the following issues:

2.1 Semantic Representations

1. What *kinds of entities* do we want to talk about? Natural Language talks about single entities, plural entities, kinds, relations, events, etc., and we need to decide which of them we are currently going to handle.

2. What *semantic representations* should we use to capture and express the meaning of NL texts? A semantic representation should be well-defined, i.e. there should be independent criteria of what the representations means which allow us to understand them and predict what the computer is going to do with them.[2] Such criteria could be spelled out by providing precise rules of (model-theoretic) interpretation for the representation language, i.e. what each expression means in terms of the entities we decided to talk about in question 1 above.

3. What does each NL sentence of interest mean? (There could be one or more readings.) We would use the representation language to help us write this meaning down. This is sometimes harder than it may seem because it may not be clear out of context what a sentence could mean (e.g. how many readings does the sentence "four boys lifted five pianos" have?). To help with answering this question, the sentence should be checked in different natural contexts and texts.

4. Once this is done, we can calculate predictions of entailments based on the interpretations of the semantic representations that are assigned to the sentences, and verify these predictions against human judgments of entailments between NL sentences. Similarly with equivalences and contradictions.

2.2 Calculating the Representations

We need to explain how to calculate the semantic representations given the NL text. This involves providing the computer with the following:

1. Knowledge of NL morphology and how to calculate it for a particular NL input.

2. Similarly with NL syntax.

3. A precise and detailed specification of the syntax-semantics interface which explains how the semantic representations are calculated from the syntactic analysis of the input NL sentence. The interface should be compositional (in an extended sense), and defined by declarative constraints and not by (destructive, one-directional) transformation rules (see, e.g., (Lev, 2005)). This includes a specification of lexical properties of base form of words (e.g. subcategorization frames, initial or default aspect of base form of verbs, etc.).

In general, understanding texts also requires uncovering the hierarchical structure of the entire text itself, including such things as rhetorical relations between sentences.

[2] Unfortunately, sometimes representations are proposed without such criteria, and the only way to check whether they are correct or useful is to run the computer program and see what happens. See (McDermott, 1978) for more on this point.

In this research we mostly focus on understanding precisely the literal, direct meaning and information in texts, so we will not pay much attention to such issues here.

2.3 Analysis of NL Phenomena

To develop the semantic representations and the methods of calculating them from the NL input, we need a careful analysis of various NL phenomena of interest. This requires learning the existing theories about them from the Linguistic literature, distilling useful ideas from these theories, expressing these ideas in a rigorous and precise form, and filling the gaps in the theories (gaps of detail and gaps of coverage). We need to examine as many different *kinds* of cases as possible so that the solutions we develop are as comprehensive as possible. We do so by inspecting cases from real data while taking into account the context of each case to help us determine the meaning of the sentences we examine.

NL is very complex, and it is usually impossible to predict the correct analysis of a new *kind* of use that has not been considered before. For example, there are many different kinds of ellipsis, and a program that is developed based on an analysis of even most of them is not guaranteed to produce any meaningful or useful results for those kinds that were not considered during the development of the program. We do not have any expectations for miracle solutions. Instead, we expect that nothing less than a thorough and meticulous analysis of all the kinds of uses could suffice for precise understanding.

To make this point clear, one can observe that Linguistic theories that are published in papers are often "refuted" in subsequent papers. But in fact, what is really refuted is not necessarily the analysis of the particular cases that were considered in earlier papers, but merely the claim that the analysis *generalizes to all cases*. Additional cases that were not considered during the development of the theory may need a slightly different analysis. Similarly, computer programs that are based on statistical models are often claimed to "generalize well and scale up". Although novel constructions that the system did not see in its training data do receive some "robust" analysis, this analysis is merely a guess which is often incorrect and meaningless.

2.4 Reasoning with the Representations

Once the representations are calculated from the text, the computer needs to reason with them. It is most convenient if existing off-the-shelf automated reasoners, i.e. theorem provers and model builders, could be used for that purpose, but they can only accept formulas in first-order logic (FOL). This formal language may not be the best one for the purpose of expressing the semantics of NL and for calculating the representations from the NL input. But there are ways of using semantic representations that are suitable for NL, and then converting them to a "more rudimentary level" of FOL formulas for the purpose of automated reasoning (much as high-level programming languages are converted to assembly code). An example of this is using Discourse Representation Structures for semantic representation. Although this formal language has the same expressive power as FOL, the formulas have a different shape that is more amenable to compositional computation from NL inputs (see e.g. (Bos, 2004)).

This is not the end of the story, however. First, there is the issue of decidability. FOL is undecidable in general, so in principle, a system that translates NL input to FOL may not be able to terminate on all queries with an answer. A common practice is to use a time limit, and if the automated reasoners do not terminate within this limit, the computer gives up and outputs *Fail*. Although this problem cannot be solved in general since NL is powerful enough to state undecidable questions, it might turn out that most or all queries of interest in a particular domain or application are in fact decidable. For example, it is quite unusual in daily life to encounter texts that require us to reason about infinitely large or even arbitrarily large domains. This area is ripe for investigation.

For the purpose of efficiency (and sometimes even decidability), in addition to off-the-shelf automated reasoners we may also want to use procedures that are specialized for certain kinds of inferences that commonly occur in NL or with our representation language, e.g. reasoning about transitive relations, sets, kinds, taxonomy, monotonicity properties of NL quantifiers and other operators, etc.

Finally, work is needed on interfacing the semantic representation language with the world knowledge component of the system. Partly, the two representations might talk about different kinds of entities. Partly, facts may be expressed in many different ways in NL, and some process of conversion to canonical knowledge representation forms might be required.

2.5 Additional Issues In An Application

When the question-answering component is applied in a specific domain or integrated in an application, there are a few additional issues to consider:

1. Knowledge about domain-specific NL phenomena and meanings that are needed for the application. E.g. certain idioms, particular ways of saying things, domain-specific word meanings.

2. General and domain-specific world knowledge needed for the application.

3. Domain-specific assumptions about the texts. For example, in a logic puzzles application, we have the closed-world and unique-names assumptions about the entities mentioned in the texts, and we also assume that exactly one of the answers to a multiple choice question is correct.

References

Bos, Johan. 2004. Computational semantics in discourse: Underspecification, resolution, and inference. *Journal of Logic, Language and Information* 13:139–157.

Lev, Iddo. 2005. The syntax-semantics interface: A gentle introduction to Glue Semantics (with some new results). Unpublished manuscript, Stanford University. http://www.stanford.edu/∼iddolev/.

McDermott, Drew. 1978. Tarskian Semantics, or No Notation Without Denotation! *Cognitive Science* 2.

Pasca, Marius, and Sanda M. Harabagiu. 2001. High performance question/answering. In *Proc. of SIGIR*, 366–374.