Computational Semantics and Pragmatics

Autumn 2011

Raquel Fernández

Institute for Logic, Language & Computation
University of Amsterdam

Plan for Today

Discussion of Bos & Markert (2005). Along the way we'll discuss:

- the use of automatic reasoning tools for Textual Entailment
  - Theorem provers
  - model building
- WordNet

[N.B.: these slides are only a guideline; most of the discussion took place in class and is not recorded here.]

Bos & Markert's Approach to TE

- Shallow semantic analysis
- Deep semantic analysis
- Comparison of approaches
- Hybrid approach

Bos & Markert's Shallow Semantic Analysis

Surface string similarity based on three shallow features: `wnoverlap`, length of T, length of H, relative length H/T.

- Tokenization: separating out (*tokenizing*) words from running text. E.g. 'New York' and 'rock'n'roll' are individual words, while 'I'm' and 'don't' are not and should be separated into two words.
- Lemmatization: mapping from related word forms that may differ in their surface realisation to their common root or *lemma*. E.g. 'sang', 'sung', 'sings' $\rightarrow$ 'sing'.
- Related lemmas via WordNet.
- The *inverse document frequency* of a lemma l is calculated as follows, where N is the total number of documents in the corpus and df_l (document frequency) is the number of documents in the corpus that contain term l.

$$\text{idf}_l = \log \frac{N}{\text{df}_l}.$$

For more detailed definitions of these notions you can check Jurafsky & Martin (2009) *Speech and Language Processing*.

Bos & Markert's Deep Semantic Analysis

Recall from the previous class...

We may want to think of TE in terms of logical consequence:

> Let the logical meaning representations of T and H be ϕ_T and ϕ_H, and B be a conjunction of axioms or knowledge base.
> If $(\phi_T \wedge B) \models \phi_H$, then $\langle T, H \rangle$ is a correct textual entailment pair.

Obvious challenges:

- assigning ϕ_T and ϕ_H to natural language expressions T and H
- defining B
- checking whether $(\phi_T \wedge B) \models \phi_H$ holds

What do these challenges involve and how can we address them with computational tools?

Validity

Assuming we have been able to assign ϕ_T and ϕ_H and define B, how do we check whether $(\phi_T \wedge B) \models \phi_H$ is a valid argument?

- ϕ_H is a logical consequence of $(\phi_T \wedge B)$ if and only if whenever the premises ϕ_T, B are satisfied in some model using some variable assignment (i.e. are true), the conclusion ϕ_H is also satisfied in the same model and using the same variable assignment (i.e. is also true).

Valid arguments are closely related to valid formulas:

$$\varphi_1, \ldots, \varphi_n \models \psi \text{ if and only if } \models (\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow \psi$$

- A formula φ is *valid* if it is satisfied in all models under any variable assignment: if it is impossible to find a situation and a context where it is not satisfied, e.g.: $\textsc{robber}(x) \vee \neg\textsc{robber}(x)$.
- A formula φ that fails to be satisfied in at least one model is *invalid*: e.g., $\textsc{robber}(x)$ is invalid.

Undecidability

So – we can conclude that $\langle T, H \rangle$ is a true textual entailment pair if we can prove that $(\phi_T \wedge B) \models \phi_H$ is a valid argument or equivalently that $(\phi_T \wedge B) \rightarrow \phi_H$ (or $\phi_T \wedge B \wedge \phi_H$) is a valid formula.

First Order Logic is undecidable: checking validity requires us to check all possible models – that's an infinite number of models!

$\Rightarrow$ there is no algorithm that can correctly decide *in finite time* whether an *arbitrary first-order* formula is valid or not.

Theorem Provers

A theorem prover is a system that provides us with a systematic method for checking whether or not it is possible to build a model in which some given formula is true or false.

Methods for theorem proving are investigated within a branch of computer science called *automated reasoning*. The crucial feature of such proof methods is that they concentrate on the syntactic structure of formulas.

Such a systematic method gives us a way to test a formula for validity:

- a formula φ would be valid if and only if the method told us that there is no way to build a model that falsifies φ (that is, a model for $\neg\varphi$).

Typically, theorem provers use *refutation proof methods*: to show that φ is valid, we show that $\neg\varphi$ leads to a contradiction.

Theorem Provers and Undecidability

Due to the undecidability of FOL, if a theorem prover fails to prove a formula this can be due to two rather different reasons:

- the formula may be not valid, hence no proof exists, and so the prover cannot find a proof;
- the formula may be valid but very difficult to prove, so the prover may use up all its resources without finding a proof (even if it exists).

We can never be sure! A theorem prover thus cannot prove non-validity. It can only prove validity (of formulas that are not extremely complex nor tricky to prove).

Theorem Provers and TE

How can we use a theorem prover to decide whether $\langle T, H \rangle$ is a true or a false textual entailment pair?

- if we give $(\phi_T \wedge B) \rightarrow \phi_H$ to a theorem prover, it will try to show that $\neg[(\phi_T \wedge B) \rightarrow \phi_H]$ leads to a contradiction. [B&M 1. in §3.2]
  - if it succeeds, we know that $(\phi_T \wedge B) \rightarrow \phi_H$ is valid and hence that $\langle T, H \rangle$ is TRUE.
- if we give it $\neg[(\phi_T \wedge B) \rightarrow \phi_H]$, it will try to show that $(\phi_T \wedge B) \rightarrow \phi_H$ leads to a contradiction. [B&M 2. in §3.2]
  - if it succeeds, we know that $\neg[(\phi_T \wedge B) \rightarrow \phi_H]$ is valid and hence that $\langle T, H \rangle$ is FALSE.

Bonus Exercise

As a "bonus" exercise, I encourage you to try an automatic theorem prover to prove e.g. one of the $\langle T, H \rangle$ pairs from HW#1.

You can download Prover9 and the model builder Mace4 and install them in your personal computer (Linux, Mac, Windows). They are also available in all the Linux machines in the Faculty of Science. To activate them, you need to do the following:

- Log in with your student account.
- In your home directory you'll find the file `.pkgrc`. This is a hidden file; if you can't see it, tick the option "Show Hidden Files" under "View".
- Open `.pkgrc` with a text editor, add `ladr` at the bottom of the file, and save it.
- Open a terminal, run the command `eval 'softpkg'` and kill the terminal.

Instructions on how to install and run the programs, create input files, the syntax of formulas, etc. can be found at: `http://www.cs.unm.edu/~mccune/prover9/` In class we tried the command line interface, but there is also a GUI version.

Bonus Exercise: Test

- As a test, we considered the following $\langle T, H \rangle$:

T: Ener bought Nival. H: Nival was sold to Ener.

- Recall that we want to prove the validity of $(\phi_T \wedge B) \rightarrow \phi_H$
- Using the syntax of Prover9 for first-order logic, we came up with the following formulas:

ϕ_T: `exists x exists y (ener = x & nival = y & buy(x,y)).`
ϕ_H: `exists x exists y (ener = x & nival = y & sell(y,x)).`
B: `all x all y (buy(x,y) <-> sell(y,x)).`

- Using a text editor (not a word processor), we created an input file for the prover (`test.in`) that included the following lines:

```
formulas(sos).
exists x exists y (ener = x & nival = y & buy(x,y)).
all x all y (buy(x,y) <-> sell(y,x)).
end_of_list.

formulas(goals).
exists x exists y (ener = x & nival = y & sell(y,x)).
end_of_list.
```

Bonus Exercise: Test

- Using the command line interface (a terminal) from the directory where our input file `test.in` had been saved, we run the prover with the command "`prover9 -f test.in > test.out`"

```
~:  prover9 -f test.in > test.out
---- Proof 1 ----
THEOREM PROVED
--- process 31610 exit (max_proofs) ---
```

- The freshly created file `test.out` contains the proof.
- If the background knowledge B is not included in the premises:

```
~:  prover9 -f test.in > test.out
SEARCH FAILED
--- process 31730 exit (sos_empty) ---
```

You may want to use the Graphical User Interface instead of the command line (especially convenient for Windows users I guess):
http://www.cs.unm.edu/~mccune/prover9/gui/v05.html

Theorem Provers and TE

How can we use a theorem prover to decide whether $\langle T, H \rangle$ is a true or a false textual entailment pair?

- if we give $(\phi_T \wedge B) \rightarrow \phi_H$ to a theorem prover, it will try to show that $\neg[(\phi_T \wedge B) \rightarrow \phi_H]$ leads to a contradiction. [B&M 1. in §3.2]
  - if it succeeds, we know that $(\phi_T \wedge B) \rightarrow \phi_H$ is valid and hence that $\langle T, H \rangle$ is TRUE.
- if we give it $\neg[(\phi_T \wedge B) \rightarrow \phi_H]$, it will try to show that $(\phi_T \wedge B) \rightarrow \phi_H$ leads to a contradiction. [B&M 2. in §3.2]
  - if it succeeds, we know that $\neg[(\phi_T \wedge B) \rightarrow \phi_H]$ is valid and hence that $\langle T, H \rangle$ is FALSE.

Problems:

- the theorem prover may not return a proof due to undecidability.
- it may return the wrong answer due to lack of background knowledge.
- many FALSE pairs are not contradictory: we can't prove non-validity without contradiction.

Model Building

But we can prove non-validity by other means:

- if we can find a model for $\neg[(\phi_T \wedge B) \rightarrow \phi_H]$, then we know that $(\phi_T \wedge B) \rightarrow \phi_H$ in not valid ...
- ... and hence that $\langle T, H \rangle$ is (perhaps) FALSE.

Model builders are relatively new automated reasoning tools that take a formula and try to build a model that satisfies it.

- They do not build arbitrary infinite models, so even when some possible models exist they may not be able to always build them.
- Model builders are only capable to build relatively small finite models.
- Often the domain size can be specified (e.g. up to 20 elements).

Bos & Markert's Deep Semantic Analysis

The theorem prover and the model builder require background knowledge.

- Features generated by the theorem prover:
  - entailed
  - inconsistent
- Features generated by the model builder:
  - domain size
  - model size
  - domain size absolute difference
  - domain size relative difference
  - model size absolute difference
  - model size relative difference

Bos & Markert's Evaluation Methods

Notions used in the evaluation of their results:

- Accuracy: percentage of correct judgements
- Confidence-weighted average score

Measures computed per class (TRUE / FALSE)

- Precision: proportion of correct hypotheses for class X given the total of hypothesis for that class.
- Recall: proportion of correct hypotheses for class X given the total number of items labeled with class X.
- F-measure: combination of precision (P) and recall (R) as follows: $F_\beta = (\beta^2 + 1)PR/\beta^2 P + R$. Normally, the parameter $\beta = 1$, therefore $F1 = 2PR/P + R$. If $\beta > 1$, recall is favoured, while $\beta < 1$ would favour precision.

Other important aspects:

- Baseline
- z-test: when are results significantly different?

Next Week

We'll start to look into Vector Space Models of word meaning.

Readings (see the overview bibliography on the website):

- Adam Kilgarriff (1997) I don't believe in word senses, *Computers and the Humanities*, 31:91-113.
- A. Lenci (2008) Distributional Semantics in Linguistic and Cognitive Research, in Lenci (ed.), *From context to meaning: Distributional models of the lexicon in linguistics and cognitive science*, special issue of the *Italian Journal of Linguistics*, 20(1):1-30.