Meaning representations

Sharon Goldwater
(based on slides by Frank Keller, Bonnie Webber, Mirella Lapata, and others)

14 November 2017
Recap: distributional semantics

• A useful way to represent meanings of individual words

• Can deal with notions of similarity

• But less clear how to deal with compositionality

• Also, we still haven’t discussed how to do inference
Example Question (6)

• Question
  Did Poland reduce its carbon emissions since 1989?

• Text available to the machine
  Due to the collapse of the industrial sector after the end of communism in 1989, all countries in Central Europe saw a fall in carbon emissions.

  Poland is a country in Central Europe.

• What is hard?
  – we need to do inference
  – a problem for sentential, not lexical, semantics
Meaning representations

• Vector space is one kind of meaning representation

• But to deal with compositionality and inference, we need meaning representations that are **symbolic** and **structured**.

• Next lecture, **semantic analysis**: how to get from sentences to their meaning representations (using syntax to help).

• But first we need to define the semantics we’re aiming at, i.e., a **meaning representation language** (MRL).
Basic assumption

The symbols in our meaning representations correspond to objects, properties, and relations in the world.

- The world may be the real world, or (usually) a formalized and well-specified world: a model or knowledge base of known facts.
  - Ex 1: a tiny world model containing 3 entities, and an exhaustive table of ‘who loves whom’ relations.
  - Ex 2: GeoQuery database [1], containing ~800 facts about US geography.
  - Ex 3: Freebase [2], “A community-curated database of well-known people, places, and things” with over 2.6 billion facts.

What do we want from an MRL?

**Compositional**: The meaning of a complex expression is a function of the meaning of its parts and of the rules by which they are combined.
What do we want from an MRL?

**Compositional**: The meaning of a complex expression is a function of the meaning of its parts and of the rules by which they are combined.

**Verifiable**: Can use the MR of a sentence to determine whether the sentence is true with respect to some given model of the world.

- In Ex 1 above, can establish the truth value of everybody loves Mary by checking it against the model.
What do we want from an MRL?

**Unambiguous:** an MR should have exactly one interpretation. So, an ambiguous sentence should have a different MR for each sense.

- Ex: each interpretation of *I made her duck* or *time flies like an arrow* should have a distinct MR.

- The job of producing all possible MRs for a given sentence will go to the semantic analyzer.

- We also defer the question of choosing which interpretation is correct.
What do we want from an MRL?

**Canonical form:** sentences with the same (literal) meaning should have the same MR.

- Ex: *I filled the room with balloons* should have the same canonical form as *I put enough balloons in the room to fill it from floor to ceiling.*

- Ex: Similarly, *Tanjore serves vegetarian food* and *Vegetarian dishes are served by Tanjore.*

- Simplifies inference and reduces storage needs; but also makes semantic analysis harder.
What do we want from an MRL?

**Inference:** we should be able to verify sentences not only directly, but also by drawing conclusions based on the input MR and facts in the knowledge base.

- Ex: from the MR for a query

  Did Poland reduce its carbon emissions?

- and the MRs for facts

  Carbon emissions have fallen for all countries in Central Europe.

  Poland is a country in Central Europe.

- we should be able to infer the answer: **YES.**
What do we want from an MRL?

Expressivity: the MRL should allow us to handle a wide range of meanings and express appropriate relationships between the words in a sentence.

• Ideally, we could express the meaning of any natural language sentence.

• In practice, we may use simpler MRLs that cover a lot of what we want.

• For example...
FOL: First-order Logic (Predicate Logic)

- A pretty good fit to what we’d like.

- Example FOL expressions:
  - tall(Kim) ∨ tall(Pierre)
  - likes(Sam, owner-of(Tanjore))
  - ∃x. cat(x) ∧ owns(Marie,x)
  - ∃x. movie(x) ∧ ∀y. person(y) ⇒ loves(y,x)
FOL: First-order Logic (Predicate Logic)

- Expressions are constructed from **terms**:
  - **constant and variable symbols** that represent entities
  - **function symbols** that allow us to indirectly specify entities
  - **predicate symbols** that represent properties of entities and relations between entities
FOL: First-order Logic (Predicate Logic)

- Expressions are constructed from **terms**:
  - **constant and variable symbols** that represent entities
  - **function symbols** that allow us to indirectly specify entities
  - **predicate symbols** that represent properties of entities and relations between entities

- Terms can be combined into **predicate-argument structures**, which in turn are combined into complex expressions using:
  - **Logical connectives**: ∨, ∧, ¬, ⇒
  - **Quantifiers**: ∀ (universal quantifier, i.e., “for all”), ∃ (existential quantifier, i.e. “exists”)
Constants in FOL

- Each constant symbol denotes exactly one entity:
  
  Scotland, EU, John, 2014

- Not all entities have a constant that denotes them:
  
  David Cameron’s right knee, this pen

- Several constant symbols may denote the same entity:
  
  The Evening Star $\equiv$ Venus
  Scotland $\equiv$ Alba
Predicates in FOL

• Predicates with one argument represent properties of entities:
  
nation(Scotland), organization(EU), tall(John)

• Predicates with multiple arguments represent relations between entities:
  
member-of(UK, EU), likes(John, Marie), introduced(John, Marie, Sue)

• We write “/N” to indicate that a predicate has arity N (takes N arguments)
  
member-of/2, nation/1, tall/1, introduced/3
The semantics of predicates

• A predicate of arity \( N \) denotes the set of \( N \)-tuples that satisfy it.
  – \( \text{likes}/2 \) is the set of \((x, y)\) pairs for which \( \text{likes}(x, y) \) is true.
  – In the following example world, a set of four pairs:
    \[
    \text{likes}(\text{John, Marie}) \quad \text{likes}(\text{Marie, Kim}) \quad \text{tall}(\text{Kim})
    \]
    \[
    \text{likes}(\text{John, Kim}) \quad \text{eats}(\text{Marie, pizza}) \quad \text{nation}(\text{UK})
    \]
    \[
    \text{likes}(\text{Kim, UK}) \quad \text{lives-in}(\text{Marie, UK}) \quad \text{nation}(\text{USA})
    \]

• If all arguments are instantiated, then the predicate-argument structure has a truth value (determined by comparing it to the set of facts in the world).
  – So, \( \text{likes}(\text{John, Kim}) \) is true, whereas \( \text{likes}(\text{John, UK}) \) is false.
Functions in FOL

• Like constants, are used to specify (denote) unique entities.

• Unlike constants, they refer to entities indirectly, so we don’t need to store as many constants.

  president(EU), father(John), right-knee(Cameron)

• Syntactically, they look like unary predicates, but denote entities, not sets.
Logical connectives

- Given FOL expressions $P$ and $Q$, the meaning of an expression containing $P$ and $Q$ is determined from the meaning of each part and the logical connective.

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$\neg P$</th>
<th>$P \land Q$</th>
<th>$P \lor Q$</th>
<th>$P \Rightarrow Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

Ex: $\text{likes(John, Kim)} \land \text{tall(John)}$ is true iff each predicate is true.
Variables in FOL

- Variable symbols (e.g., $x, y, z$) range over entities.

- An expression consisting only of a predicate with a variable among its arguments is interpreted as a set:

  \[ \text{likes}(x, \text{Kim}) \text{ is the set of entities that like Kim.} \]

- A predicate with a variable among its arguments only has a truth value if it is \textbf{bound} by a quantifier.

  \[ \forall x. \text{likes}(x, \text{Kim}) \text{ has an interpretation as either true or false.} \]
Universal Quantifier (∀)

- Can be used to express general truths:

  Cats are mammals has MR ∀x.cat(x) ⇒ mammal(x)

- This MR is true iff the conjunction of all similar expressions is true, where each of these substitutes a different constant for the variable.

  cat(Sam) ⇒ mammal(Sam) ∧
  cat(Zoot) ⇒ mammal(Zoot) ∧
  cat(Whiskers) ⇒ mammal(Whiskers) ∧
  cat(UK) ⇒ mammal(UK) ∧
  …
Existential Quantifier ($\exists$)

- Used to express that a property/relation is true of some entity, without specifying which one:

  $\text{Marie owns a cat}$ has MR $\exists x. \text{cat}(x) \land \text{owns}(\text{Marie}, x)$

- This MR is true iff the disjunction of all similar expressions is true, where each of these substitutes a different constant for the variable.

  $\text{cat(Sam)} \land \text{owns(Marie, Sam)} \lor$
  $\text{cat(Zoot)} \land \text{owns(Marie, Zoot)} \lor$
  $\text{cat(Whiskers)} \land \text{owns(Marie, Whiskers)} \lor$
  $\text{cat(UK)} \land \text{owns(Marie, UK)} \lor$
  $\ldots$
Existential Quantifier (∃)

- Why use ∧ not ⇒? Notice the difference between these two MRs:
  \[ \exists x. \text{cat}(x) \land \text{own}(\text{Marie}, x) \]  
  vs  
  \[ \exists x. \text{cat}(x) \Rightarrow \text{own}(\text{Marie}, x) \]  

In English:

There is something that is a cat and Marie owns it vs
There is something that if it’s a cat, Marie owns it

- \( P \Rightarrow Q \) is true if the antecedent (left of the \( \Rightarrow \)) is false.

- So the righthand MR is true if there is anything that’s not a cat!
  - If \( \text{cat}(\text{UK}) \) is false, then \( \text{cat}(\text{UK}) \Rightarrow \text{own}(\text{Marie, UK}) \) is true,
    and so is \( \exists x. \text{cat}(x) \Rightarrow \text{own}(\text{Marie, x}) \).
Quantifier scoping

• Consider the following sentence:

   Everyone loves some movie

   – No ambiguity in POS tags, syntactic structure, or word senses.
   – But this sentence is still ambiguous!
Quantifier scoping

• Consider the following sentence:

Everyone loves some movie

– No ambiguity in POS tags, syntactic structure, or word senses.
– But this sentence is still ambiguous!

• Two possible meanings:

(a) There is a single movie that everyone loves
(b) Everyone loves at least one movie, but the movies might be different

• This kind of ambiguity is called quantifier scope ambiguity
Quantifier scope ambiguity

• The two meanings have different MRs:

(a) $\exists x. \text{movie}(x) \land \forall y. \text{person}(y) \Rightarrow \text{loves}(y,x)$
(b) $\forall y. \text{person}(y) \Rightarrow \exists x. \text{movie}(x) \land \text{loves}(y,x)$

• In (a), the ‘$\exists$’ has scope over the ‘$\forall$’; in (b) it’s vice versa.

• Other examples of quantifier scope ambiguity:

  A boy gave flowers to each teacher
  Every cat chased a dog
Are we done?

- MRs in FOL are verifiable, unambiguous, canonical.

- Predicate-argument structure is a good match for natural language
  - Predicate-like elements: verbs, prepositions, adjectives
  - Argument-like elements: nouns, NPs

- Determiners (a, some, every) and coordination (if, and, or) can often be expressed with logical connectives and quantifiers.

- But what about compositionality?
Suppose we have the following words with the following meanings:

<table>
<thead>
<tr>
<th>word</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marie</td>
<td>Marie</td>
</tr>
<tr>
<td>pizza</td>
<td>pizza</td>
</tr>
<tr>
<td>loves</td>
<td>love(x,y)</td>
</tr>
</tbody>
</table>

How do we get from there to the meaning of the sentence *Marie loves pizza*?
Lambda (\(\lambda\)) Expressions

- Extension to FOL, allows us to work with ‘partially constructed’ formulae.

- A \(\lambda\)-expression consists of:
  - the Greek letter \(\lambda\), followed by a variable (formal parameter);
  - a FOL expression that may involve that variable.

\[
\lambda x. \text{sleep}(x)
\]

‘The function that takes an entity \(x\) to the FOL expression \(\text{sleep}(x)\)’

- This lambda is the same one used in Python!
\( \lambda \)-Reduction

- A \( \lambda \)-expression can be **applied** to a **term**

\[
\lambda x. \text{sleep}(x) (\text{Marie})
\]

- This expression can be simplified using **\( \lambda \)-reduction**: replace the formal parameter with the term and remove the \( \lambda \). Result:

\[
\text{sleep(Marie)}
\]
Nested $\lambda$-expressions

- Use one $\lambda$-expression as the body of another.

- Allows predicates with several arguments to accept them one by one.

\[ \lambda y. \lambda x. love(x,y) \]

‘The function that takes $y$ to (the function that takes $x$ to the FOL expression $love(x,y)$)’

\[ \lambda z. \lambda y. \lambda x. give(x,y,z) \]

‘The function that takes $z$ to (the function that takes $y$ to (the function that takes $x$ to the FOL expression $give(x,y,z)$))’
Nested $\lambda$-reduction

- Starting from binary predicate $\lambda y. \lambda x. \text{love}(x,y)$

- Apply to first argument:

  $\lambda y. \lambda x. \text{love}(x,y)$ (pizza) becomes $\lambda x. \text{love}(x, \text{pizza})$

- Apply to second argument:

  $\lambda x. \text{love}(x, \text{pizza})$ (Marie) becomes $\text{love}(\text{Marie}, \text{pizza})$
Summary

- First-order logic can be used as a meaning representation language for natural language.

- λ-expressions can be used to compute meaning representations compositionally.

- Next time, we will see how to use these tools in a syntax-driven approach to semantic analysis.