EMNLP
Meaning Representation and Semantic Analysis

Yansong Feng
fengyansong@pku.edu.cn

Institute of Computer Science and Technology
Peking University

June 12, 2018

Recap: Semantics

- Distributional semantics
- Shallow semantics

Recap: Semantics

- Distributional semantics
- Shallow semantics
- Less clear how to deal with compositionality
- Still haven't discussed how to do inference

Examples

- Question
  Did Poland reduce its carbon emissions since 1989?
- Resources available
  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 emmissions. Poland is a country in Central Europe.

Examples

- Question
  Did Poland reduce its carbon emissions since 1989?
- Resources available
  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 emmissions. Poland is a country in Central Europe.
- What is hard?

Examples

- Question
  Did Poland reduce its carbon emissions since 1989?
- Resources available
  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 emmissions. Poland is a country in Central Europe.
- What is hard?
  - need to do inference
  - a problem for sentential, not lexical, semantics

Meaning Representations

- Vector Space Model is one kind of meaning representation (MR)
- For compositionality and inference, we need meaning representations that are **symbolic** and **structured**.
- More difficult: how about the MRs of sentences (using syntax to help).

Meaning Representations

- Vector Space Model is one kind of meaning representation (MR)
- For compositionality and inference, we need meaning representations that are **symbolic** and **structured**.
- More difficult: how about the MRs of sentences (using syntax to help).

Firstly, define the semantics we aim at, i.e., a meaning representation language (MRL).

Basic Assumptions

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.
  - a tiny world model containing 3 entities, and an exhaustive table of ‘who loves whom’ relations.
  - GeoQuery dataset, containing 800 facts about US geography.
  - Freebase, “A community-curated database of well- known people, places, and things” with over 2.6 billion facts.

What do we expect?

- **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.
- **Unambiguous**: an MR should have exactly one interpretation. So, an ambiguous sentence should have a different MR for each sense.
- **Canonical form**: Sentences with the same (literal) meaning should have the same MR.
- **Inference**: Be able to verify sentences not only directly, but also by drawing conclusions based on the input MR and facts in the KB.
- **Expressivity**: Allow us to handle a wide range of meanings and express appropriate relationships between the words in a sentence.

What do we expect?

- **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.
- **Unambiguous**: an MR should have exactly one interpretation. So, an ambiguous sentence should have a different MR for each sense.
  - each interpretation of time flies like an arrow should have a distinct MR
- **Canonical form**: Sentences with the same (literal) meaning should have the same MR.
- **Inference**: Be able to verify sentences not only directly, but also by drawing conclusions based on the input MR and facts in the KB.
- **Expressivity**: Allow us to handle a wide range of meanings and express appropriate relationships between the words in a sentence.

What do we expect?

- **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.
- **Unambiguous**: an MR should have exactly one interpretation. So, an ambiguous sentence should have a different MR for each sense.
- **Canonical form**: Sentences with the same (literal) meaning should have the same MR.
  - Tanjore serves vegetarian food and Vegetarian dishes are served by Tanjore should have the same canonical form
- **Inference**: Be able to verify sentences not only directly, but also by drawing conclusions based on the input MR and facts in the KB.
- **Expressivity**: Allow us to handle a wide range of meanings and express appropriate relationships between the words in a sentence.

What do we expect?

- **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.
- **Unambiguous**: an MR should have exactly one interpretation. So, an ambiguous sentence should have a different MR for each sense.
- **Canonical form**: Sentences with the same (literal) meaning should have the same MR.
- **Inference**: Be able to verify sentences not only directly, but also by drawing conclusions based on the input MR and facts in the KB.
  - query: Did Poland reduce its carbon emissions?
  - facts: Carbon emmissions have fallen for all countries in Central Europe. and Poland is a country in Central Europe.
  - answer: yes
- **Expressivity**: Allow us to handle a wide range of meanings and express appropriate relationships between the words in a sentence.

FOL: First-order Logic

- Predicate Logic
- A simpler MRL that covers a lot of what we want.
- $tall(Kim) \vee tall(Pierre)$
- $likes(Sam, ownerOf(Tanjore))$
- $\exists x.cat(x) \wedge owns(Marie, x)$
- $\exists x.movie(x) \wedge \forall y.person(y) \Rightarrow loves(y, x)$

FOL: First-order 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

- 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: $\forall$ (universal quantifier, i.e., "for all"), $\exists$ (existential quantifier, i.e. "exists")

Constants

- Each constant symbol denotes exactly one entity
- Not all entities have a constant that denotes them
- Several constant symbols may denote the same entity:

Predicates

- 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 and Functions

Predicates:

- A predicate of arity N denotes the set of N-tuples that satisfy it.
  likes/2 is the set of (x, y) pairs for which likes(x, y) is true
- 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).
  nation(UK) is true, locatedIn(China, Europe) is not true

Functions:

- look like unary predicates
- used to specify (denote) unique entities indirectly
- president(EU), father(John)

Logical Connectives, Variables, Quantifiers

Logical Connectives

- $\neg, \vee, \wedge, \Rightarrow$

Variables

- range over entities
- An expression consisting only of a predicate with a variable among its arguments is interpreted as a set:
  likes(x, Kim) is the set of entities that like Kim
- A predicate with a variable among its arguments only has a truth value if it is bound by a quantifier.
  $\forall x.likes(x, Kim)$ has an interpretation as either true or false.

Quantifiers

- Universal Quantifier $\forall$:
  Cats are mammals has MR $\forall x.cat(x) \Rightarrow mammal(x)$
- Existential Quantifier $\exists$:
  Marie owns a cat has MR $\exists x.cat(x) \wedge owns(Marie, x)$

Advantages

- verifiable, unambiguous, canonical
- Predicate-argument structure
- Determiners and coordination $\rightarrow$ logical connectives and quantifiers

Advantages

- verifiable, unambiguous, canonical
- Predicate-argument structure
- Determiners and coordination $\rightarrow$ logical connectives and quantifiers
- Compositionality??

Lambda (λ) Expressions

- Extension to FOL, allows us to work with ‘partially constructed’ formulae
- A λ-expression consists of:
  - the Greek letter λ, followed by a formal parameter
  - a FOL expression that may involve that variable
    $\lambda x.nation(x)$
- λ-reduction:
  - $\lambda x.sleep(x)(Marie)$ becomes $sleep(Marie)$
  - $\lambda y.\lambda x.love(x, y)(pizza)$ becomes $\lambda x.love(x, pizza)$
  - $\lambda x.love(x, pizza)(Marie)$ becomes $love(Marie, pizza)$

Describing Events

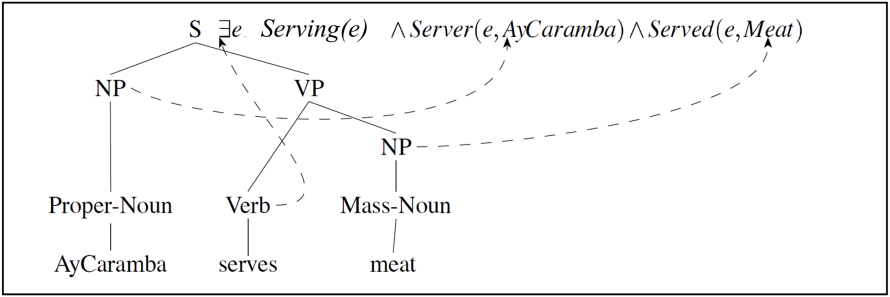
- Reifying events: introduce variables for events, which we can quantify over
- John gave Mary a book
- $\exists e, z. Giving(e) \wedge Giver(e, John) \wedge Givee(e, Mary) \wedge Given(e, z) \wedge Book(z)$
- John gave Mary a book on Tuesday
- $\exists e, z. Giving(e) \wedge Giver(e, John) \wedge Givee(e, Mary) \wedge Given(e, z) \wedge Book(z) \wedge Time(e, Tuesday)$
- relatively complex structures, but clear argument assignments

Semantic Analysis

- Semantic Analysis or Semantic Parsing
- The task of constructing meaning representations from sentences.
- Most popular: Syntax Driven Semantic Analysis

Semantic Analysis

- Semantic Analysis or Semantic Parsing
- The task of constructing meaning representations from sentences.
- Most popular: Syntax Driven Semantic Analysis



[S. G.]

Syntax Driven Semantic Analysis

- Based on the principle of compositionality.
  - meaning of the whole built up from the meaning of the parts
  - more specifically, in a way that is guided by word order and syntactic relations.
- Build up the MR by augmenting CFG rules with semantic composition rules.
- Representation produced is literal meaning: context independent and free of inference

CFG Rules with Semantic Attachments

- To compute the final MR, we add semantic attachments to our CFG rules.
- These specify how to compute the MR of the parent from those of its children.
- Rules will look like: $A \rightarrow \alpha_1 ... \alpha_n \{f(\alpha_j.sem, ..., \alpha_k.sem)\}$
- $A.sem$ (the MR for A) is computed by applying the function f to the MRs of some subset of A's children.

Rules for Nouns

- AyCaramba serves meat with its parse tree
- Rules with semantic attachments for nouns and NPs:
  ProperNoun → AyCaramba {**AyCaramba**}
  MassNoun → meat {**Meat**}
  NP → ProperNoun {**ProperNoun.sem**}
  NP → MassNoun {**MassNoun.sem**}
- Unary rules normally just copy the semantics of the child to the parents (as in NP rules here).

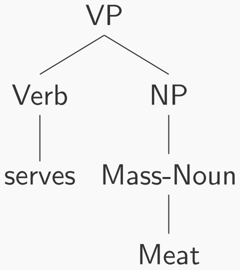
Rules for Verbs

- we could have: $\lambda y.\lambda x.Serving(x, y)$
- the rule:
  Verb $\rightarrow$ serves $\{\lambda y.\lambda x.Serving(x, y)\}$
- or, for **reified events**:
  $\lambda y.\lambda x.\exists e.Serving(e) \wedge Server(e, x) \wedge Served(e, y)$
- the rule:
  Verb $\rightarrow$ serves $\{\lambda y.\lambda x.\exists e.Serving(e) \wedge Server(e, x) \wedge Served(e, y)\}$

Building Larger Constituents

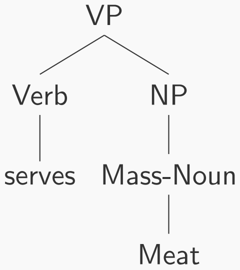
- The remaining rules specify how to apply λ-expressions to their arguments.
- For example, the VP rule is:
  VP $\rightarrow$ Verb NP $\{$**Verb.sem(NP.sem)**$\}$

- The remaining rules specify how to apply λ-expressions to their arguments.
- For example, the VP rule is:
  VP $\rightarrow$ Verb NP {**Verb.sem(NP.sem)**}



Building Larger Constituents

- The remaining rules specify how to apply λ-expressions to their arguments.
- For example, the VP rule is:
  VP $\rightarrow$ Verb NP $\{$**Verb.sem(NP.sem)**$\}$



- VP.sem $= \lambda y.\lambda x.\exists e.Serving(e) \wedge Server(e,x) \wedge Served(e,y)(Meat)$
  $= \lambda x.\exists e.Serving(e) \wedge Server(e,x) \wedge Served(e,Meat)$

Putting Everything Together

- The final rule: S $\rightarrow$ NP VP $\{$**VP.sem(NP.sem)**$\}$
- with NP.sem = AyCaramba

Putting Everything Together

- The final rule: S $\rightarrow$ NP VP $\{$**VP.sem(NP.sem)**$\}$
- with NP.sem $=$ AyCaramba
- then:
  S.sem $=$
  $\lambda x.\exists e.Serving(e) \wedge Server(e, x) \wedge Served(e, Meat)(AyCaramba)$
  $= \exists e.Serving(e) \wedge Server(e, AyCaramba) \wedge Served(e, Meat)$

Putting Everything Together

- The final rule: S $\rightarrow$ NP VP $\{$**VP.sem(NP.sem)**$\}$
- with NP.sem = AyCaramba
- then:
  S.sem =
  $\lambda x.\exists e.Serving(e) \wedge Server(e,x) \wedge Served(e,Meat)(AyCaramba)$
  $= \exists e.Serving(e) \wedge Server(e,AyCaramba) \wedge Served(e,Meat)$
- how about **Every** kid cries
  $\forall x.Child(x) \Rightarrow \exists e.Sleeping(e) \wedge Sleeper(e,x)$

Putting Everything Together

- The final rule: S $\rightarrow$ NP VP $\{$**VP.sem(NP.sem)**$\}$
- with NP.sem = AyCaramba
- then:
  S.sem =
  $\lambda x.\exists e.Serving(e) \wedge Server(e, x) \wedge Served(e, Meat)(AyCaramba)$
  $= \exists e.Serving(e) \wedge Server(e, AyCaramba) \wedge Served(e, Meat)$
- how about **Every** kid cries
  $\forall x.Child(x) \Rightarrow \exists e.Sleeping(e) \wedge Sleeper(e, x)$
- Is this correct?

Dealing with issues

- Define S.sem as NP.sem(VP.sem) as well
- Make NP.sem into a functor by adding λ
  $\lambda Q \forall x.Child(x) \Rightarrow Q(x)$
- then, apply it to VP.sem:
  $\lambda Q \forall x.Child(x) \Rightarrow Q(x)(\lambda y.\exists e.Sleeping(e) \wedge Sleeper(e, y))$
  $\forall x.Child(x) \Rightarrow (\lambda y.\exists e.Sleeping(e) \wedge Sleeper(e, y))(x)$
  $\forall x.Child(x) \Rightarrow \exists e.Sleeping(e) \wedge Sleeper(e, x)$

The Right NP.sem?

- We will need a new set of noun rules:
  Noun → Child { $\lambda x.Child(x)$ }
  Det → Every { $\lambda P.\lambda Q.\forall x.P(x) \Rightarrow Q(x)$ }
  NP → Det Noun {**Det.sem(Noun.sem)**}
- for Every Child, we have:
  $\lambda P \lambda Q \forall x.P(x) \Rightarrow Q(x)(\lambda x.Child(x))$
  $\lambda Q \forall x.(\lambda x.Child(x))(x) \Rightarrow Q(x)$
  $\lambda Q \forall x.Child(x) \Rightarrow Q(x)$

For Proper Nouns

- Assign a different MR to proper nouns, allowing them to take VPs as arguments:
  ProperNoun → Kate {λ**P. P(Kate)**}
- For Kate sleeps, we have
  $\lambda P.P(Kate)(\lambda y.\exists e.Sleeping(e) \wedge Sleeper(e, y))$
  $(\lambda y.\exists e.Sleeping(e) \wedge Sleeper(e, y))(Kate)$
  $\exists e.Sleeping(e) \wedge Sleeper(e, Kate)$
- Here, we type-raised the the argument a of a function f, turning it into a function g that takes f as argument.

Finally

- S→NP VP {**NP.sem(VP.sem)**}
- VP → Verb {**Verb.sem**}
- VP→Verb NP {**Verb.sem(NP.sem)**}
- NP → Det Noun {**Det.sem(NP.sem)**}
- NP → ProperNoun {**ProperNoun.sem**}
- Det → Every $\{\lambda P.\lambda Q.\forall x.P(x) \Rightarrow Q(x)\}$
- Noun → Child $\{\lambda x.Child(x)\}$
- ProperNoun → Kate $\{\lambda P.P(Kate)\}$
- Verb → sleeps $(\lambda x.\exists e.Sleeping(e) \wedge Sleeper(e, x))$
- Verb → serves $\{\lambda y.\lambda x.\exists e.Serving(e) \wedge Server(e, x) \wedge Served(e, y)\}$

Semantic parsing algorithms

- Given a CFG with semantic attachments, how do we obtain the semantic analysis of a sentence?
- One option (integrated): Modify syntactic parser to apply semantic attachments at the time syntactic constituents are constructed.
- Second option (pipelined): Complete the syntactic parse, then walk the tree bottom-up to apply semantic attachments.

Learning a semantic parser

- Much current research focuses on learning semantic grammars rather than hand-engineering them.
- Given sentences paired with meaning representations, can we automatically learn
  - Which words are associated with which bits of MR?
  - How those bits combine (in parallel with the syntax) to yield the final MR?
- Second option (pipelined): Complete the syntactic parse, then walk the tree bottom-up to apply semantic attachments.

Readings

- Choi, E., Kwiatkowski, T., and Zettlemoyer, L. (2015). Scalable semantic parsing with partial ontologies. In Proceedings of the Association for Computational Linguistics.
- Kwiatkowski, T., Zettelmoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.
- Reddy, S., Lapata, M., and Steedman, M. (2014). Large-scale semantic parsing without question- answer pairs. Transactions of the Association for Computational Linguistics, 2:377392.
- Zettlemoyer, L. S. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In Proceedings of the Conference on Uncertainty in Artificial Intelligence.