Introduction to computational semantics and the syntax-semantics interface

Aurélie Herbelot

University of Trento/Geneva

November 2016

Introduction

What is semantics?

- Semantics is the study of meaning.
- What is meaning?
- How come words and sentences *have* meaning?
- *What* is the meaning of words and sentences?
- Do two people *mean* the same thing when they utter the word *cat*?
- ...

What is meaning?

- No one knows for sure...
- Go read *http://plato.stanford.edu/entries/meaning/*...
- One of the most fundamental human faculties.
- Without the ability to process and communicate meaning, humans wouldn't have built complex artifacts and societies.
- Complex thoughts need a powerful tool to be conveyed efficiently and accurately, without relying on pointing/miming, etc.

Meaning is reference

- We use words to *refer*, i.e. talk *about* things in the world.
- There is a *correspondence* between fragments of a language and state-of-affairs in the world.
- So... we can evaluate the truth of sentences.
- Tarski: *Snow is white* is true iff snow is white.
- Referential theories of meaning are truth-theoretic.

Sense and reference (Frege)

- The morning star and the evening star are the same planet in the real world: Venus.
- The referent of both morning star and evening star are the same.
- But for someone who doesn't know they are the same, they are two different concepts.
- Separate 'sense' from 'reference'?

Meaning is conceptual

- Psycholinguistic tradition.
- Words are linked through associations which can be reliably extracted from humans.
  - Similarity: is a dog more similar to a cat or to an elephant?
  - Priming: is the word *cat* recognised quicker once I've seen *dog*?
  - Categories: cluster the following into sensible categories: *dog, fork, knife, cat, mouse, hammer, plate, screwdriver*

Meaning is use

- Distributionalist tradition (Harris, Firth in linguistics; Wittgenstein in philosophy).
- Firth: “You shall know a word by the company it keeps”.
- Wittgenstein: no essential properties, just context of utterance.

Meaning is signal passing

- The point of language is to communicate.
- Communication involves some signal passing between A and B. It assumes A $\neq$ B.

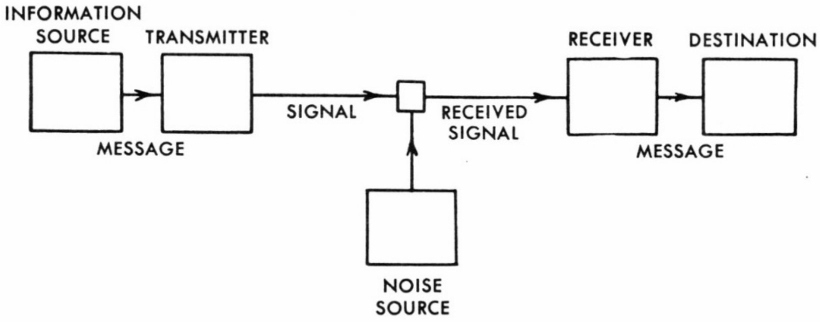
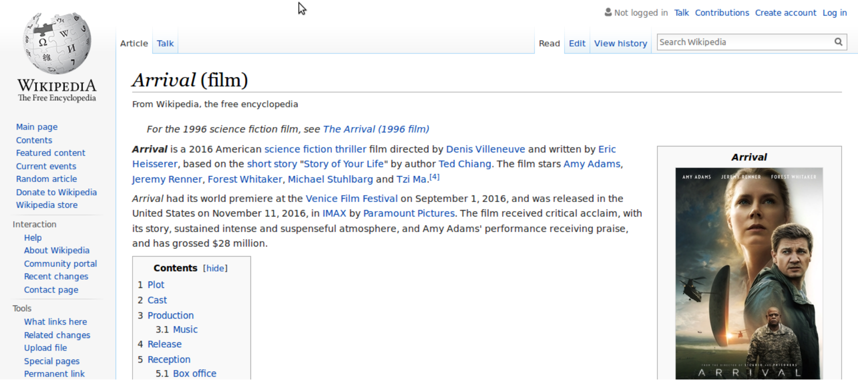


Fig. 1. — Schematic diagram of a general communication system.

Meaning is all of this



Not logged in Talk Contributions Create account Log in

Article Talk

Read Edit View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link

Arrival (film)

From Wikipedia, the free encyclopedia

For the 1996 science fiction film, see The Arrival (1996 film)

Arrival is a 2016 American science fiction thriller film directed by Denis Villeneuve and written by Eric Heisserer, based on the short story "Story of Your Life" by author Ted Chiang. The film stars Amy Adams, Jeremy Renner, Forest Whitaker, Michael Stuhlbarg and Tzi Ma.[4]

Arrival had its world premiere at the Venice Film Festival on September 1, 2016, and was released in the United States on November 11, 2016, in IMAX by Paramount Pictures. The film received critical acclaim, with its story, sustained intense and suspenseful atmosphere, and Amy Adams' performance receiving praise, and has grossed $28 million.

Contents [hide]

1 Plot
2 Cast
3 Production
3.1 Music
4 Release
5 Reception
5.1 Box office

Arrival

AMY ADAMS JEREMY RENNER FOREST WHITAKER

ARRIVAL

Computational semantics: an overview

Aspects of computational semantics

- Syntax-semantics interface, parsing.
- Inference and entailment.
- Compositionality.
- Representation of meaning (including multimodal aspects).
- Semantic ambiguity.
- Lexical semantics and ontologies.
- ...

Syntax-semantics interface

- To what extent does the meaning of an expression depend on its syntactic structure? (And the other way round!)
- Note: at least in its referential sense, meaning is universal. Given the real world, it is generally true that *dogs are mammals*. But this may be expressed in many different ways in different languages.
- More on this today!

Inference and entailment

- Humans excel at inference:
  - The cat is on the sofa $\longrightarrow$ There is an animal on the sofa.
  - Google may buy Twitter $\longrightarrow$ Twitter may be sold to Google.
  - All cats are mammals $\longrightarrow$ My cat is a mammal.
  - Is the window open? $\longrightarrow$ I am cold.
- Some aspects of inference can be obtained through referential theories of meaning (the more logical ones). Others require more 'soft' reasoning.

Compositionality

- Frege: the meaning of an utterance is a function of the meaning of its parts.
  - *Bob is the fastest man on Earth.*
- Although... the meaning of words depends on the utterance they occur in.
  - *Bob is the fastest man on Earth.*
  - *Bob is the fastest man on Mars.*
  - Who is faster?

  - *I want a new bat for Christmas.*
  - *I saw several bats in the cave.*

Representations of meaning

- How we represent meaning depends heavily on the aspect we wish to emphasise. The meaning of a word can be:
  - a set (referential theories);
  - a structure in an ontology (conceptual theories);
  - a vector (distributional theories);
  - a combination of the above??
- The latest distributional representations do not only encapsulate linguistic meaning but also perceptual information.

Semantic ambiguity

- As in syntax, we find in ambiguity in semantics, both at the word and structural level.
- Word sense ambiguity: *bat, bank,*
- Structural ambiguity: *All students read a book.*
    - There is a book such that all students read that book.
    - For every student, that student read a book.
- Does it make sense to speak about word senses? Those are often arbitrary (compare two dictionaries!)
- Do humans always disambiguate? Probably not. (See underspecification.)

Lexical semantics and ontologies

- Can we write down the world's knowledge? Manually? Automatically?
- Can we formalise lexical relations such as synonymy, antonymy, hypernymy:
  - Synonymy: two words mean the same thing (or nearly the same thing?)
  - Antonymy: two (or several) words have incompatible meanings (*dead/alive*).
  - Hypernymy: the *is-a* relationship.

Formal semantics, briefly

Model-theoretic semantics

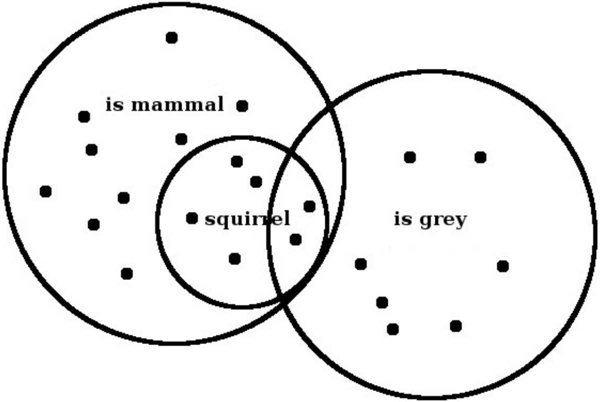


Photo: kuhmi, flickr, CC

Argument-predicate structure

- A model can be formally expressed in terms of predicates and their arguments:
  - One-place predicates: *squirrel, sleep, grey...* Those select a set of individuals (the set of squirrels, the set of things that sleep, the set of grey things...)
  - Two-place predicates: *love, friend-of, born-in...* Those select a set of ordered pairs of individuals (e.g. (Ada_Lovelace,1815), (Ferdinand_de_Saussure,1857)).
- We usually speak of the 'extension' or 'denotation' of the expression to refer to the individuals picked by the predicate.

Truth

- The functional view of model theory is that we can see the predicate-argument structure as a set of functions returning true of false according to the state of the world under consideration.
- Examples:
  - I(love(squirrel,nut)) = 1
  - I(born_in(Ada_Lovelace,2016)) = 0
- We talk of an 'interpretation function'. The interpretation function can also return actual entities in the model:
  - I(squirrel(x)) = the set of squirrels

First-order logic

- Fix the state-of-affairs under consideration.
- Fix operators and their meaning.
- Define variables and their meaning.
- Evaluate formulas (predicates, sentences) according to the interpretation function.

Example

- Let's have a world with two squirrels, Steve and Squeaky. Both are sleeping.
- The denotation of the predicate squirrel, *squirrel'*, is the set {Steve, Squeaky}.
- Let's introduce some operators:
  - quantifiers: $\exists$ and $\forall$;
  - conjunction operators: $\wedge \longrightarrow$;
  - negation: $\neg$.

Example

- $P_1 = \exists x[squirrel'(x) \wedge sleep'(x)]$
  There exists an x such that x is a squirrel and x sleeps.
  $I(P_1) = 1$
- $P_2 = \forall x[squirrel'(x) \longrightarrow sleep'(x)]$
  For all x, it holds that if x is a squirrel then x sleeps.
  $I(P_2) = 1$
- $P_3 = \neg squirrel'(Steve)$
  Steve is not a squirrel.
  $I(P_3) = 0$

Syntax-semantics interface

- How can we relate the syntactic structure of sentences to their meaning?
- There isn't a one-to-one correspondence between syntactic and semantic structures.
- Squirrels like nuts:
  - S(NP(N(squirrels)) VP(V(like) (NP(N(nuts)))))
  - like'(squirrel',nut')
- Nuts are liked by squirrels:
  - S(NP(N nuts)) VP(V(are) V(V(liked) PP(P(by) N(squirrels))))
  - like'(squirrel', nut')

Syntax-semantics interface

- We need a representation which allows us to compactly express generalisations about the correspondence between syntax and semantics.
- In other words, we need to build the semantic representation of a sentence as we build its syntax (composition process).
- Let's first look at the formal semantics interpretation of this process.
- Then, we will consider a computational representation that allows us to do the same.

Lambda calculus

- Lambda calculus has a variable-binding operator λ, which can be thought as a place-holder for missing information.
- Lambdas tell us where we should substitute information in the process of composition.
- An operation called β-conversion performs the required substitutions.

Example

- Here is a lambda term:
  - $\lambda x.sleep(x)$
- The prefix λx. binds the variable x in $sleep(x)$. It is said to *abstract* over x.
- Let's add an argument to the right of our expression:
  - $\lambda x.sleep(x)(kitty)$
- This is expression is called a *functional application*. The left-hand side is the *functor* and the right-hand side the *argument*.
- Performing β-conversion, we obtain $sleep(kitty)$.

Functional application

- Functional application has the form *Functor*(*Argument*).
- It triggers β-conversion, by which the lambda-bound variables are replaced by the argument:
  - we strip off the λ prefix;
  - we remove the argument;
  - we replace all occurrences of the *lambda*-bound variable by the argument.

Beyond reference

Ontologies and lexical resources

Aurélie Herbelot

University of Trento/Geneva

November 2016

Problems with set theory: The meaning of life

life’

Problems with set theory: What is (a) football?

(Search results on duckduckgo.com.)

Problems with set theory: Speaker dependence

- Meaning is speaker-dependent.
- What I mean by *cup* is not what you mean by *cup* (Labov 1978, Wierzbicka 1984).

The need for world knowledge

- It is not sufficient for a robot to be able to translate natural language into logical forms.
- It must be able to *recognise* objects and concepts, and to *reason* over their attributes:
  - A football is a round object made of leather or plastic.
  - If something is round, it will roll when pushed.

Intension

- There are different notions of *intension*. Today, we'll look at the 'informal' one.
- Intension is the conceptual content that allows us to identify an extension.
- E.g. I may never have seen a unicorn, but I have read enough descriptions of unicorns that I would know one if I saw it.

Structural approaches to world knowledge

- Structural representations of conceptual knowledge are known as ‘lexical resources’ or ‘ontologies’.
- **Lexical resources** mostly include information about the meaning of content words (sometimes including some proper nouns).
- The term **ontology** comes from the philosophical study of ‘what there is’. Ontologies mostly contain information about attributes of individuals (e.g. Mount Everest is 8848m high).
- Lexical resources / ontologies are structured in that they encapsulate specific relations about the meaning of a word.

WordNet

WordNet overview

- An online lexical database originally developed at Princeton University (for English!) Available at http://wordnet.princeton.edu/
- Open multilingual WordNet: a project by Francis Bond in Singapore. Available at http://compling.hss.ntu.edu.sg/omw/. (150 languages)
- WordNet can be used through NLTK:
    - *import nltk*
    - *from nltk.corpus import wordnet as wn*

WordNet structure

- Nouns, verbs, adjectives and adverbs organised into synonym sets (*synsets*).
- Each synset represents a *concept*: unlike in dictionaries, words are organised by meaning, not word form.
- Synsets are linked through various relations: synonymy, antonymy, hyponymy, meronymy, troponymy, entailment...
- WordNet 3.0: 117,000 synsets (mostly nouns: 82,000).

The lexical matrix

Word meanings	Word forms: bank	eggplant	aubergine
C1	X		
C2	X		
C3		X	X

- *bank* is polysemous: it has more than one sense (corresponding to more than one concept).
- *eggplant* and *aubergine* are synonymous: the two word forms correspond to one concept only.
- Each WordNet synset corresponds to a word meaning (concept) rather than a word form. It lists different word forms for that concept.

NLTK synsets vs word forms

```
>>> for l in wn.synset('plant.n.01').lemmas():
...     print l.name()
...
plant
works
industrial_plant
>>>
>>> for l in wn.synset('plant.n.02').lemmas():
...     print l.name()
...
plant
flora
plant_life
>>>
```

Lexical relations: hyponymy

- Hyponymy is the *is-a* relation (also called *taxonomic* relation):
    - *cat* is a hyponymy of *mammal*;
    - *mammal* is a hypernym of *cat*.
- Hyponymy is the main relation in the WordNet noun hierarchy.
- Two synsets which are hyponyms of the same synset are called *co-hyponyms*. *cat* and *dog* are co-hyponyms.

NLTK: basic hyponymy relations

```
>>> import nltk
>>> wn.synsets('dog',wn.NOUN)
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'), S
ynset('frank.n.02'), Synset('pawl.n.01'), Synset('andiron.n.01')]
>>>
>>> wn.synset('frank.n.02').definition()
u'a smooth-textured sausage of minced beef or pork usually smoked; often served on a
 bread roll'
>>>
>>> wn.synset('dog.n.01').hypernyms()
[Synset('canine.n.02'), Synset('domestic_animal.n.01')]
>>>
>>> wn.synset('dog.n.01').hyponyms()
[Synset('basenji.n.01'), Synset('corgi.n.01'), Synset('cur.n.01'), Synset('dalmatian
.n.02'), Synset('great_pyrenees.n.01'), Synset('griffon.n.02'), Synset('hunting_dog.
n.01'), Synset('lapdog.n.01'), Synset('leonberg.n.01'), Synset('mexican_hairless.n.0
1'), Synset('newfoundland.n.01'), Synset('pooch.n.01'), Synset('poodle.n.01'), Synse
t('pug.n.01'), Synset('puppy.n.01'), Synset('spitz.n.01'), Synset('toy_dog.n.01'), S
ynset('working_dog.n.01')]
>>>
```

Lexical relations: meronymy

- Meronymy is the *part-of* relation:
  - trunk is a meronym of *tree*;
  - *tree* is a holonym of *trunk*.
- There are several types of meronyms, depending on the notion of 'part' under consideration:
  - part-meronyms: *trunk/tree*
  - substance-meronyms: *heartwood/tree*
  - member-holonyms: *tree/forest*

Lexical relations: troponymy and entailment

- Troponymy and entailment are verb relations.
- The verb Y is a troponym of the verb X if Y means doing X in some particular manner (*fly* is a troponym of *move*).
- The verb Y is entailed by X if by doing X you must be doing Y (*sleep* is entailed by *snore*).
- Note: the verb relations are not well documented in WordNet.

The noun hierarchy

- The noun hierarchy is organised as a tree, with one top node: *entity*.
- Entities can be *abstractions* or *physical entities*.
- Warning: the top of the WordNet hierarchy does not necessarily look sensible...

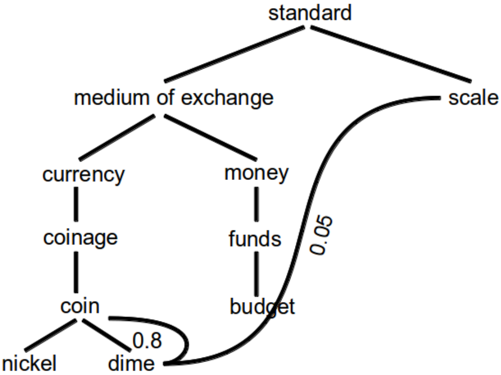
The top of the noun hierarchy

```
>>> wn.synset('entity.n.01').definition()
u'that which is perceived or known or inferred to have its own distinct existence (livi
ng or nonliving)'
>>>
>>> wn.synset('entity.n.01').hyponyms()
[Synset('abstraction.n.06'), Synset('physical_entity.n.01'), Synset('thing.n.08')]
>>>
>>> wn.synset('abstraction.n.06').hyponyms()
[Synset('attribute.n.02'), Synset('communication.n.02'), Synset('group.n.01'), Synset('
measure.n.02'), Synset('otherworld.n.01'), Synset('psychological_feature.n.01'), Synset
('relation.n.01'), Synset('set.n.02')]
>>>
>>> wn.synset('physical_entity.n.01').hyponyms()
[Synset('causal_agent.n.01'), Synset('matter.n.03'), Synset('object.n.01'), Synset('pro
cess.n.06'), Synset('substance.n.04'), Synset('thing.n.12')]
>>>
>>> wn.synset('thing.n.12').definition()
u'a separate and self-contained entity'
>>>
>>> wn.synset('thing.n.08').definition()
u'an entity that is not named specifically'
>>>
>>> wn.synset('thing.n.08').hyponyms()
[Synset('change.n.06'), Synset('freshener.n.01'), Synset('horror.n.02'), Synset('jimdan
dy.n.02'), Synset('pacifier.n.02'), Synset('security_blanket.n.01'), Synset('stinker.n.
02'), Synset('whacker.n.01')]
>>>
```

WordNet similarity

- It is possible to calculate similarity between synsets by following the WordNet graph.
- Intuition: the shorter the path between two nodes, the more similar two words are.

WordNet similarity



WordNet similarity

- Easiest way to compute similarity is by path length:

$$sim_{path}(s1, s2) = -log\, pathlen(s1, s2) \tag{1}$$

where *pathlen* is the number of edges in the shortest path.

- Problem: not every link in the path has the same length:
puppy is-a *dog*
phytoplankton is-a *living_thing*

WordNet similarity: Resnik (1995)

- Let's add a function $p : C \longrightarrow [0, 1]$ to the taxonomy. $p(c)$ is the probability to encounter an instance of concept $c \in C$.
- The probability of the top node (*entity*) is 1.
- If $c1$ is-a $c2$, then $p(c1) \leq p(c2)$.
- The *information content* of a concept c is $-log\, p(c)$. (As probability increases, informativeness decreases.)

WordNet similarity: Resnik (1995)

- New measure of similarity:

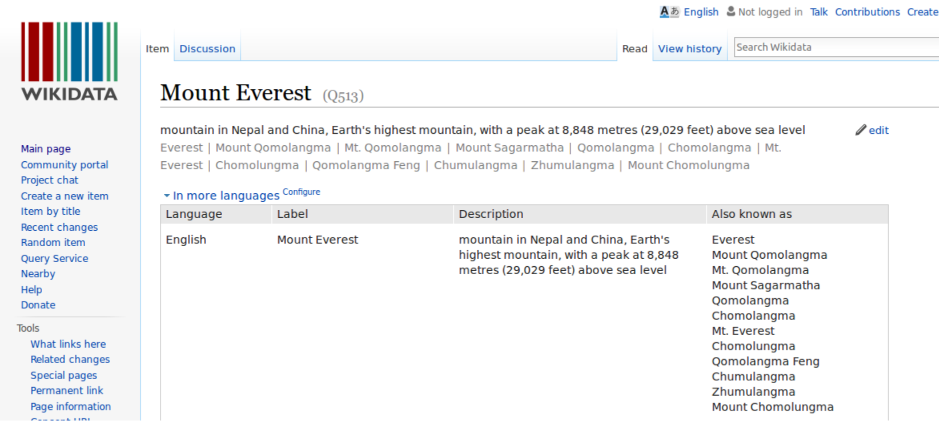
$$sim(c1, c2) = max[-log\, p(c)] \tag{2}$$

  where $c \in S(c1, c2)$ and S is the subsumption relation.
- Intuitively, we select the lowest hypernym of both terms and return its information content.
- The length of the links doesn't matter. We only care about how deep the lowest common hypernym is in the hierarchy.

WordNet similarity: Resnik (1995)

- But how do we calculate $p(c)$?
- Simply, by taking the frequency of the word in a large corpus, divided by the number of words in that corpus.
- But a word is not a concept...
- Set $sim(w1, w2) = max\ sim(c1, c2)$. The assumption is that when calculating the similarity between two words out of context, we are considering their closest senses.

Another type of resource: Wikidata



Another type of resource: Wikidata

- Wikidata contains both lexical and non-lexical information:
  - coordinate_location: 27°59’17”N, 86°55’31”E
  - continent: Asia
  - first ascent (time): 29 May 1953
  - first ascent (participant): {Edmund Hillary, Tenzing Norgay}

Automatic ontology extraction

Why?

- Save person-years... and potentially save lives...
- Automatic fact extraction:
  - *X suppresses Y* (in a chemistry paper)
  - *people suffering from Z have high levels of Y* (in a medical journal)
  - $\longrightarrow$ X could help people with Z???
- If we could automatically extract large amounts of relations from naturally-occuring data, we could infer new facts...

Pattern-based extraction: Hearst (1992)

- Automatically extract hyponyms: Hearst noticed that some surface patterns indicate hyponymic relations with high probability.

(2) *such NP as {NP ,}* {(or | and)} NP*

`... works by such authors as Herrick, Goldsmith, and Shakespeare.`

$\Longrightarrow$ *hyponym("author", "Herrick"), hyponym("author", "Goldsmith"), hyponym("author", "Shakespeare")*

(3) *NP {, NP}* {,} or other NP*

`Bruises, wounds, broken bones or other injuries ...`

$\Longrightarrow$ *hyponym("bruise", "injury"), hyponym("wound", "injury"), hyponym("broken bone", "injury")*

Pattern-based extraction: Hearst (1992)

- To implement Heart's algorithm, we will need parsed (or at least chunked) text, to identify the NPs.
- The rest is just regular expressions...

Pattern-based extraction: Hearst (1992)

```
>>> s = "works by such authors as Herrick, Goldsmith and Shakespeare"
>>> tokens = nltk.word_tokenize(s)
>>> tokens
['works', 'by', 'such', 'authors', 'as', 'Herrick', ',', 'Goldsmith', 'and', 'Shakespea
re']
>>>
>>> pos_tags = nltk.pos_tag(tokens)
>>> pos_tags
[('works', 'NNS'), ('by', 'IN'), ('such', 'JJ'), ('authors', 'NNS'), ('as', 'IN'), ('He
rrick', 'NNP'), (',', ','), ('Goldsmith', 'NNP'), ('and', 'CC'), ('Shakespeare', 'NNP')
]
>>>
```

Pattern-based extraction: Hearst (1992)

```
>>> s = "works by such famous authors as Herrick, Goldsmith and Shakespeare"
>>> tokens = nltk.word_tokenize(s)
>>> pos_tags = nltk.pos_tag(tokens)
>>> pos_tags
[('works', 'NNS'), ('by', 'IN'), ('such', 'JJ'), ('famous', 'JJ'), ('authors', 'NNS'),
('as', 'IN'), ('Herrick', 'NNP'), (',', ','), ('Goldsmith', 'NNP'), ('and', 'CC'), ('Sh
akespeare', 'NNP')]
>>>
>>> pattern = "NP: {<JJ>?<NNS>}"
>>> NPChunker = nltk.RegexpParser(pattern)
>>> print NPChunker.parse(pos_tags)
(S
  (NP works/NNS)
  by/IN
  such/JJ
  (NP famous/JJ authors/NNS)
  as/IN
  Herrick/NNP
  ,/,
  Goldsmith/NNP
  and/CC
  Shakespeare/NNP)
>>>
```

Augmenting WordNet?

- Can we directly augment WordNet with output from the Hearst patterns?
- No. WordNet is organised by senses. We don't know which senses were extracted through our patterns.
- Use a word sense disambiguation (WSD) algorithm...

Word Sense Disambiguation (Lesk 1986)

- Assumption: words that co-occur together in a sentence *S* share the same topic in *S*.
- We can use dictionary definitions to ‘guess’ the appropriate senses of two words.
- The Lesk algorithm is still at the basis of many WSD applications (albeit not necessarily using dictionaries).

Word Sense Disambiguation (Lesk 1986)

PINE

1. kinds of evergreen trees with needle-shaped leaves
2. waste away through sorrow or illness

CONE

1. solid body which narrows to a point
2. something of this shape whether solid or hollow
3. fruit of certain evergreen trees

PINE CONE?

Word Sense Disambiguation (Lesk 1986)

```
FOR s1 in Senses(w1)
    FOR s2 in Senses(w2)
    ADD word_overlap(s1,s2) to word_overlaps
RETURN max(word_overlaps)
```

- Note: the exact definition of *word overlap* will vary depending on whether all words are taken into account or not, whether stemming has taken place, etc...

Word Sense Disambiguation using WordNet Banerjee & Pedersen (2002)

- Try WSD on running text, using WordNet glosses.
- Let's define context:
  - *context* of the target word: a window of size $2k + 1$.
    E.g. ...the children were **[**gathering pine cones**]** in the forest...
  - If the word to be disambiguated is at the beginning/end of the sentence, use $2k$ words after/before it.
  - Consider only the words that are in WordNet.

Word Sense Disambiguation using WordNet Banerjee & Pedersen (2002)

- The algorithm considers each pair of words in the window under consideration.
- For each word *w* in each pair *P*, select all WordNet synsets of the word, *as well as* synsets related by a direct WordNet relation (e.g. hyponyms, meronyms, etc).
- Apply the Lesk algorithm to *P*, with the following modification: the overlap is not calculated between single words, but overlapping *sequences*.

Word Sense Disambiguation using WordNet Banerjee & Pedersen (2002)

- Sequence overlap:
  - some kinds of **evergreen trees** with needle-shaped leaves
  - fruit of certain **evergreen trees**
  - An overlap of length 2.
- The sequence overlap must have at least one content word!
- Each overlap contributes a score equal to the square of the length of the overlap:
  $S = 2^2 = 4$

Word Sense Disambiguation using WordNet Banerjee & Pedersen (2002)

- First advantage: by considering the relational neighbourhood of a synset, the overlap calculation draws on more complete information.
- Second advantage: using sequence overlap gives us a natural way to take some syntactic/semantic relations into account, without parsing.

Summary so far...

- We now know that we can organise lexical knowledge in a graph like WordNet.
- The graph is organised via specific relations (e.g. hyponymy) linking *senses* of words.
- To add elements to the graph, we need:
  - Some reliable patterns, corresponding to a particular relation.
  - A disambiguation algorithm telling us which sense of a word the new term should be linked to.
- How do we find patterns?

Finding new patterns: the Snowball algorithm Agichtein & Gravano (2000)

- Let's assume we want to find out a list of organisations and their headquarters.
- Let's also assume we already know some of them.

Organisation	Location
Microsoft	Redmond
Exxon	Irving
IBM	Armonk
Boeing	Seattle
Intel	Santa Clara

Finding new patterns: the Snowball algorithm Agichtein & Gravano (2000)

- We can use already known relations as ‘seeds’ to discover new patterns.
- For instance, observe that the pair <Microsoft, Richmond> is found in the sentence fragment:
  *computer servers at **Microsoft**’s headquarters in **Richmond**.*
- Build a pattern from observed contexts:
  <STRING1>’s headquarters in <STRING2>

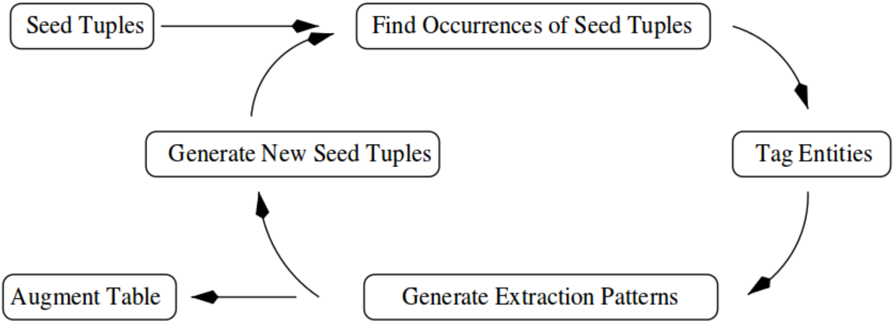
Exploiting named entities

- A good pattern must be *reliable* and have good *coverage*.
- Assume we have found the following pattern:
  <STRING2>-based <STRING1>
  as in *Richmond-based Microsoft*
- Good pattern? It will return:
  - Seattle-based Boeing
  - Chicago-based company
  - ...
- We must make sure that STRING1 is an organisation and STRING2 a location!

Exploiting named entities

```
>>> s = "Microsoft's headquarters are in Richmond."
>>> tokens = nltk.word_tokenize(s)
>>> tokens
['Microsoft', "'s", 'headquarters', 'are', 'in', 'Richmond', '.']
>>>
>>> pos_tags = nltk.pos_tag(tokens)
>>> pos_tags
[('Microsoft', 'NNP'), ("'s", 'POS'), ('headquarters', 'NNS'), ('are', '
VBP'), ('in', 'IN'), ('Richmond', 'NNP'), ('.', '.')]
>>>
>>> print nltk.ne_chunk(pos_tags, binary=True)
(S
  (NE Microsoft/NNP)
  's/POS
  headquarters/NNS
  are/VBP
  in/IN
  (NE Richmond/NNP)
  ./.)
>>>
```

Overview of the Snowball algorithm



Snowball: measure of pattern confidence

- Let's assume we extracted the pattern <ORGANISATION , LOCATION>. How reliable is it?
- The confidence of a pattern is given by:

$$Conf(P) = \frac{P.positive}{P.positive + P.negative} \quad (3)$$

- Example:
  - **Exxon, Irving**, said
  - **Intel, Santa Clara**, cut prices
  - invest in **Microsoft, New York**-based analyst Jane Smith said

$Conf(P) = \frac{2}{2+1} = 0.67$

Snowball: Measure of tuple confidence

- Let's assume that $Conf(P)$ is a rough approximation of the probability of P to produce good tuples.
- We can calculate a tuple T's confidence as a function of the confidence of the patterns that produce it:

$$Conf(T) = 1 - \prod_{i=0}^{|P|}(1 - Conf(P_i)) \tag{4}$$

Snowball: summary

- Snowball is a way to greedily acquire new information, using a pattern-based approach.
- Danger: increasing recall by producing many new patterns can be detrimental to precision.
- The notion of confidence is key to the performance of the system!