# Natural Language Processing: Part II
# Overview of Natural Language Processing (L90): Part III/ACS

2017, 12 Lectures, Michaelmas Term (these notes, lectures 5 to 8)
October 15, 2017
Ann Copestake (`aac@cl.cam.ac.uk`)
`http://www.cl.cam.ac.uk/users/aac10/`

# 5 Lecture 5: Dependencies

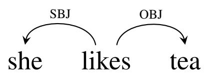
At the end of the last section, we saw some difficulties with CFGs. There are many alternative formalisms for natural language, which have different strengths and weaknesses. One alternative which is very popular in NLP (though not so much in English linguistics) is the use of *dependency structures*. This lecture is an introduction to the use of dependency parsing in NLP: at the end, there is a background section which discusses some other alternatives to CFG.[1]

## 5.1 Introduction to dependency structures

Dependencies relate words (tokens) in the sentence directly via unlabelled or labelled relationships. With labelled dependency approaches, labels are chosen from a smallish fixed set. As used in NLP, dependencies are usually weakly-equivalent to a CFG, but more powerful variants exist.

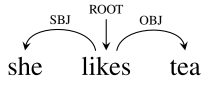
There are a number of different schemes for representing dependency structures. The relationships encoded are usually syntactic (although see next lecture for semantic dependencies): syntactic dependency structures can be seen as an alternative to parse trees which have the advantage of capturing meaningful relationships more directly. Dependency parsers produce such representations directly, but in NLP dependencies are also constructed by converting another representation such as a parse tree (though such conversions are generally imperfect/lossy).

The following is a very simple example of a dependency structure:

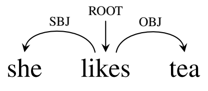


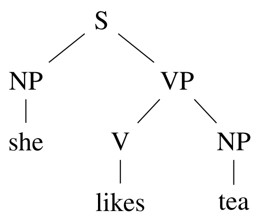
Dependency structures have arcs from a head (here the verb *likes*) to its dependents (the subject and direct object).

It is usual to identify one node as the ROOT, typically the main verb in an utterance:



The following structure shows a syntax tree for the same sentence, for comparison:

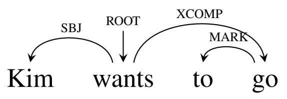




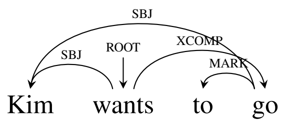
[1]This lecture is partly based on Chapter 14 of the Jurafsky and Martin third edition draft, but that goes into many aspects which I ignore or skim over here.

Unlike the syntax tree, the dependency structure has no intermediate nodes such as VP. There is no direct notion of constituency in dependency structures (although some notions of constituency can be recovered). Lack of intermediate node labels is actually helpful for generality in NLP annotation in that notions of constituency vary a lot between different approaches. However it means that we can't model some syntactic and semantic phenomena so directly/easily. Dependency structures are intuitively closer to meaning than the parse trees. They are also more neutral to word order variations, as discussed below.

In some varieties of dependency grammar, it is possible to have non-tree structures, which are required to represent *control*, for instance. Consider the following example:



Here I am using labels from the Universal Dependency set (Nivre et al, 2016): XCOMP is clausal complement and MARK stands for a semantically empty marker (the infinitival *to*, which most linguists treat as having no meaning, although in some analyses of English it is treated as a type of auxiliary). But this representation is inadequate because *Kim* is also the agent of *go*. A better representation would be:



However, since this is not a tree, standard dependency parsing approaches will not work, and the computational complexity of parsing is higher. It is therefore usual within NLP to either ignore this problem entirely, or to regard parsing as a two-phase process, and add non-tree arcs to the dependency structure in a second phase.

In NLP, it is also usual to restrict attention to dependency trees which are *projective*. I will not go through the formal definition of projectivity, but non-projective structures can easily be seen because they have to be drawn with crossing arcs. Non-projective structures are fairly rare in English, but they arise more often in free word order languages.

In principle, one could add additional power to dependency formalisms by adding a notion of feature in order to represent agreement, for instance. In fact, dependency formalisms can be encoded in sufficiently rich feature structure frameworks. However, as with probabilistic CFGs, it is more normal in NLP to allow the probabilities and features to (partially) model case, agreement and so on. This is discussed further below.

## 5.2 Word order across languages

One reason to prefer dependency formalisms over CFG-based approaches is their higher degree of neutrality to word order variation which varies substantially cross-linguistically.

English word order is predominantly subject verb object (SVO) and 'who did what to whom' meaning is indicated by order. For example *the dog bites that man* means something different from *that man bites the dog*. In the right context, topicalization (OSV order) is possible:

(1) That man, the dog bites.

(2) Her pets are mostly very friendly, but that man, the dog bites.

Passive has a different structure where the syntactic subject is the 'underlying' or 'deep' object.[2]

(3) The man was bitten by the dog.

(4) * Him was bitten by the dog.

[2]Semantic roles names, such as agent and patient, are sometimes used to avoid this terminology, but this gives rise to another set of problems.

Many languages allow freer word order. Sometimes 'who did what to whom' is indicated by case. For instance, in German:

(5) Der Hund beißt den Mann

(6) Den Mann beißt der Hund

both mean 'the dog bites the man'. However only masculine gender changes between nominative and accusative in German, and if nominative and accusative case are not distinguished, word order determines the meaning.

(7) Die Kuh hasst eine Frau

means 'the cow hates a woman' and cannot mean 'a woman hates the cow' (at least, not in any normal text).

In contrast, even when English marks case (as with some pronouns), word order is fixed:

(8) * him likes she

is ungrammatical, and will not be interpreted by most hearers as 'she likes him'.[3]

All variations on the SVO word order are found in some natural languages, although most are SOV or SVO. OSV is only found in four natural languages (see Bender 2013:pp91–93). Some languages have very free word order and can't be categorized by this scheme. A Russian example (from Bender, 2013: p92):

| | | |
|---|---|---|
| Chelovek | ukusil | sobaku |
| man.NOM.SG.M | bite.PAST.PFV.SG.M | dog-ACC.SG.F |
| the man bit the dog | | |

All word orders are possible with same meaning (in different discourse contexts, compare English topicalization):

Chelovek ukusil sobaku
Chelovek sobaku ukusil
Ukusil chelovek sobaku
Ukusil sobaku chelovek
Sobaku chelovek ukusil
Sobaku ukusil chelovek

Because of word order variability, CFG rules like:

```
S -> NP VP
```

do not work in all languages. There are a range of possible approaches to this (besides simply ignoring the problem and just working on English). It is possible to modify a CFG-based approach to ignore the order of the rule's daughters, and allow discontinuous constituency e.g., the VP is split for *sobaku chelovek ukusil* ('dog man bit') etc. However parsing is more difficult. Richer frameworks than CFG can cope with such examples. For instance, Bender (2008) describes a feature-structure grammar for Wambaya which has many cases of split constituents. But again, the richer formalisms makes parsing more difficult.

The option we will consider here is the use of dependency parsing. While dependency representations (especially as used in NLP) do not entirely avoid word-order problems, they are less problematic than CFGs for many languages.

## 5.3 Introduction to dependency parsing

For NLP purposes, we assume dependency trees which are projective and weakly-equivalent to CFGs. A range of different algorithms have been tried: here I will describe *transition-based dependency parsing*, a variant of shift-reduce parsing. This is deterministic, and thus in strong contrast to chart-parsing which we saw in the previous lecture, although other dependency parsing algorithms exist which do incorporate search.

[3]Though, to an extent, weird word order is surprisingly comprehensible:

(9) Found someone, you have.

Dependency parsing is nearly always treated as a type of machine learning, with parsers trained on dependency banks (sometimes constructed by automatically converting CFG-based treebanks). There is thus no grammar as such.

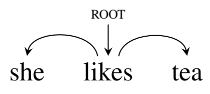
An example of transition-based dependency parsing (without labels) is shown below:

| stack | word list | action | relation added |
|---|---|---|---|
| ROOT | she, likes, tea | SHIFT | |
| ROOT, she | likes tea | SHIFT | |
| ROOT, she, likes | tea | LeftArc | she ← likes |
| ROOT, likes | tea | SHIFT | |
| ROOT, likes, tea | | RightArc | likes → tea |
| ROOT, likes | | RightArc | ROOT → likes |
| ROOT | | Done | |

There is a stack, a list of words to be processed, and a record of actions. For now, we assume an *oracle* which chooses the correct action each time (this is the part where we need machine learning, as discussed below). For the unlabelled case, the oracle chooses between only three actions: LeftArc, RightArc or SHIFT. Informally:

1. At the start, the stack just contains the ROOT symbol.
2. Parsing is deterministic: at each step either SHIFT a word from the list onto the stack, or link the top two items on the stack (via LeftArc or RightArc).
3. Only the head word is left on the stack after an arc is added.
4. All added arcs are recorded.
5. The algorithm terminates when there is nothing in the word list and only ROOT on the stack.

The output for the example above is: she ← likes, likes → tea, ROOT → likes.
This corresponds to unlabelled dependencies, as follows:



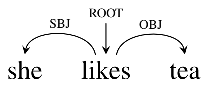
We now consider the oracle. This is treated as a machine learning *classifier*.[4] Given the stack and the word list, the classifier must choose an action at each step. This is a form of supervised machine learning: trained by extracting parsing actions from correctly annotated data. The *features* extracted from the training instances can be the actual word forms, the morphological analysis, parts of speech etc. In fact *feature templates* are used: these are automatically instantiated to give huge number of actual features. A wide range of different machine learning models are possible: MaxEnt, SVMs — recently deep learning approaches have become popular.

Labels on arcs increase the number of classes and this complicates the oracle. In the example below, I have shown the labels and also the POS tags on the input words, to give some better idea of the features available:

| | | | |
|---|---|---|---|
| ROOT | she_PNP, likes_VVZ, tea_NN1 | SHIFT | |
| ROOT,she_PNP | likes_VVZ, tea_NN1 | SHIFT | |
| ROOT,she_PNP, likes_VVZ | tea_NN1 | LeftArcSUBJ | she ← likes SUBJ |
| ROOT,likes_VVZ | tea_NN1 | SHIFT | |
| ROOT,likes_VVZ, tea_NN1 | | RightArcObj | likes → tea OBJ |
| ROOT,likes_VVZ | | RightArc | ROOT → likes |
| ROOT | | Done | |

The dependency structure would be:

[4]We will look at another example of a machine learning classifier in more detail when we consider anaphora resolution in a later lecture.



Dependency parsing can be very fast. The greedy algorithm described can go wrong, but usually has reasonable accuracy. (Note that humans process language incrementally and (mostly) deterministically.) There is no notion of grammaticality, so parsing is robust to typos and Yodaspeak, but will not directly indicate ungrammaticality. The oracle's decisions are sensitive to case, agreement etc because of the features. For instance, when analyzing example 6, *den Mann beißt der Hund*, the choice between LeftArcSubj and LeftArcObj is conditioned on the case of the noun phrase as well as its position.

## 5.4 Universal dependencies

I now briefly return to a discussion of the appropriate labels for dependencies. Recent work on *universal dependencies* (UD) is an ongoing attempt to define a set of dependencies which will work cross-linguistically (Nivre et al 2016: `http://universaldependencies.org`). This work draws on attempts to define a 'universal' set of POS tags. There are now UD depbanks for over 50 languages (though most small).

In fact, no single set of dependencies is useful for all languages, and some languages do not demonstrate some of the UDs. There is tension between universality and meaningful dependencies, which is acknowledged by the project. It has been necessary to fins a balance between developing a linguistically-motivated scheme, ease of human annotation, parsing efficiency and so on. There are some vague 'catch all' classes in UD: e.g., MARK since words like English infinitival *to* resist clean classification.

Many linguistic generalizations can't be captured by dependencies alone, although in one form or other, a related notion is part of most modern linguistic approaches. But for NLP, dependency parsing is clearly useful. I will briefly discuss semantic dependencies in the next lecture.

## 5.5 Background: other alternatives to context-free grammars

There are a huge number of approaches to grammar and parsing in the CL/NLP literature, though not so many systems in practical use.[5] Nearly all the parsers designed to work on arbitrary text use some form of statistical parse ranking component in combination with a symbolic component. The symbolic grammar may be written by a linguist (though there will be some automatic induction of lexical entries) or automatically learned from a manually-annotated Treebank. It is also possible to use a Treebank constructed by manually selecting from alternative analyses provided by a symbolic grammar in order to train a statistical ranking component. There is also work on unsupervised grammar induction, but this has been primarily aimed at understanding human language acquisition rather than practical performance. Grammars vary in the depth of annotation they provide for an analysis (e.g., whether they support compositional semantics, see next lecture) and the degree of overgeneration they assume: a few are bidirectional, which makes them usable for generation from logical form, but such grammars will not have 100% coverage on arbitrary text and have to be used with some form of additional robustness component if complete coverage is required for an application.

There has been a considerable degree of cooperation between linguists and computational linguists in the development of constraint-based frameworks that are linguistically adequate for a wide range of languages and yet sufficiently tractable that they could be used in automatic parsing. Two such frameworks which have active communities of both linguists and computational linguists are Lexical Functional Grammar (LFG), initially developed by Bresnan and Kaplan in the 1970s and Head-driven Phrase Structure Grammar (HPSG), due to Pollard and Sag, which started in the 1980s. Much of the recent academic work involves two international collaborations, PARGRAM (`https://pargram.b.uib.no/`) for LFG and DELPH-IN (`http://www.delph-in.net`) for HPSG. Both frameworks allow the specification of formal grammars (including morphology, syntax and compositional semantics) which can be used for both parsing and generation. They have both been used to develop efficient computational grammars with

[5]Although the Computer Laboratory is associated with the development of three quite different approaches to parsing which are all widely used: DELPH-in (more below), RASP (`http://www.sussex.ac.uk/Users/johnca/rasp/`) and the C&C parser (`http://svn.ask.it.usyd.edu.au/trac/candc/wiki/Publications`).

high coverage for English and a small number of other languages. The large grammars are more than competitive with automatically-induced Treebank grammars for some applications. In both LFG and HPSG, illustrative grammars have been developed for a very wide range of human languages and various cross-linguistic generalizations can be captured in the grammars.

DELPH-IN distributes Open Source FS grammars for a variety of languages. The English Resource Grammar (ERG, Flickinger 2000) is probably the largest freely available bidirectional grammar for English. It has been widely used in research and commercial applications, including, for instance, a system for teaching written English used by over 30,000 schoolchildren in the US.

## 6 Lecture 6: Compositional semantics

Compositional semantics is the study of how meaning is conveyed by the structure of a sentence (as opposed to lexical semantics, which is primarily about word meaning, which we'll discuss in lectures 7, 8 and 9). In lecture 4, we looked at grammars primarily as a way to describe a language: i.e., to say which strings are part of the language and which are not, or (equivalently) as devices that, in principle, could generate all the strings of a language. In lecture 5, we looked at syntactic structure more abstractly, but this was still syntax rather than semantics. What we usually want for language analysis is some idea of the meaning of a sentence. At its most basic, this is 'who did what to whom?' but clearly there is much more information that is implied by the structure of most sentences. The parse trees we saw in lecture 4 and the dependency structures in lecture 5 have some of this information, but it is implicit rather than explicit. In the simple examples covered by those grammars, syntax and semantics are very closely related, but if we look at more complex examples, this is not the case.

Consider the following examples:

(10) a Kitty chased Rover.

b Rover was chased by Kitty.

The meaning these two sentences convey is essentially the same (what differs is the emphasis) but the parse trees are quite different.[6] A possible logical representation would be $\text{chase}'(k, r)$, assuming that $k$ and $r$ are constants corresponding to *Kitty* and *Rover* and $\text{chase}'$ is the two place predicate corresponding to the verb *chase*.[7] Note the convention that a predicate corresponding to a lexeme is written using the stem of the lexeme followed by $'$: $\text{chase}'$. A logical meaning representation constructed for a sentence is called the *logical form* of the sentence. Here and in what follows I am ignoring tense for simplicity although this is an important aspect of sentence meaning.

Another relatively straightforward example that shows a syntax/semantics mismatch is *pleonastic pronouns*: i.e., pronouns that do not refer to actual entities. See the examples below (with indicative logical forms).

(11) a It barked.

b $\exists x[\text{bark}'(x) \wedge \textsc{pron}(x)]$

(12) a It rained.

b $\text{rain}'$

In *it rains*, the *it* does not refer to a real entity (it will not be resolved, see the discussion of anaphora resolution in a later lecture), so the semantics should not involve any representation for the *it*.

More complex examples include verbs like *seem*: for instance *Kim seems to sleep* means much the same thing as *it seems that Kim sleeps* (contrast this with the behaviour of *believe*). An indicative representation for both would be $\text{seem}'(\text{sleep}'(k))$ — but note that there is no straightforward way of representing this in a first order logic.

There are many more examples of this sort that make the syntax/semantics interface much more complex than it first appears and demonstrate that we cannot simply read the compositional semantics off a parse tree, dependency structure or other syntactic representation.

Grammars that produce explicit representations for compositional semantics are often referred to as *deep grammars*. Deep grammars that do not overgenerate (much) are said to be *bidirectional*. This means they can be used in the realization step in a Natural Language Generation system to produce text from an input logical form (see lecture 11). This generally requires somewhat different algorithms from parsing (although *chart generation* is a variant of parsing), but this will not be discussed in this course.

In this lecture, I will start off by showing how simple logical representations can be produced from CFGs. I will outline how such structures can be used in inference and describe experiments on robust textual entailment.

[6] Very broadly speaking, the second sentence is more appropriate if Rover is the topic of the discourse: we'll very briefly return to this issue in lectures 9 and 10.

[7] Although introductory logic books invariably assume proper names correspond to constants, this does not work well for a broad coverage system. Another option is: $\exists x, y[\text{chase}'(x, y) \wedge \text{Kitty}'(x) \wedge \text{Rover}'(y)]$.

## 6.1 Compositional semantics using lambda calculus

The assumption behind compositional semantics is that the meaning of each whole phrase must relate to the meaning of its parts. For instance, to supply a meaning for the phrase *chased Rover*, we need to combine a meaning for *chased* with a meaning for *Rover* in some way.

To enforce a notion of compositionality, we can require that each syntactic rule in a grammar has a corresponding semantic rule which shows how the meaning of the daughters is combined. In linguistics, this is usually done using lambda calculus, following the work of Montague. The notion of lambda expression should be familiar from previous courses (e.g., Computation Theory, Discrete Maths). Informally, lambda calculus gives us a logical notation to express the argument requirements of predicates. For instance, we can represent the fact that a predicate like bark′ is 'looking for' a single argument by:

$$\lambda x[\text{bark}'(x)]$$

Syntactically, the lambda behaves like a quantifier in FOPC: the *lambda variable* $x$ is said to be within the scope of the *lambda operator* in the same way that a variable is syntactically within the scope of a quantifier. But *lambda expressions* correspond to functions, not propositions. The lambda variable indicates a variable that will be bound by function application. Applying a lambda expression to a term will yield a new term, with the lambda variable replaced by the term. For instance, to build the semantics for the phrase *Kitty barks* we can apply the semantics for *barks* to the semantics for *Kitty*:

$$\lambda x[\text{bark}'(x)](k) = \text{bark}'(k)$$

Replacement of the lambda variable is known as *lambda-conversion*. If the lambda variable is repeated, both instances are instantiated: $\lambda x[\text{bark}'(x) \wedge \text{sleep}'(x)]$ denotes the set of things that bark and sleep

$$\lambda x[\text{bark}'(x) \wedge \text{sleep}'(x)](r) = \text{bark}'(r) \wedge \text{sleep}'(r)$$

A partially instantiated transitive verb predicate has one uninstantiated variable, as does an intransitive verb: e.g., $\lambda x[\text{chase}'(x, r)]$ — the set of things that chase Rover.

$$\lambda x[\text{chase}'(x, r)](k) = \text{chase}'(k, r)$$

Lambdas can be nested: this lets us represent transitive verbs so that they apply to only one argument at once. For instance: $\lambda x[\lambda y[\text{chase}'(y, x)]]$ (which is often written $\lambda x \lambda y[\text{chase}'(y, x)]$ where there can be no confusion). For instance:

$$\lambda x[\lambda y[\text{chase}'(y, x)]](r) = \lambda y[\text{chase}'(y, r)]$$

That is, applying the semantics of *chase* to the semantics of *Rover* gives us a lambda expression equivalent to the set of things that chase Rover.

The following example illustrates that bracketing shows the order of application in the conventional way:

$$(\lambda x[\lambda y[\text{chase}'(y, x)]](r))(k) = \lambda y[\text{chase}'(y, r)](k) = \text{chase}'(k, r)$$

In other words, we work out the value of the bracketed expression first (the innermost bracketed expression if there is more than one), and then apply the result, and so on until we're finished.

**A grammar fragment** In this fragment, I'm using $X'$ to indicate the semantics of the constituent $X$ (e.g. NP′ means the semantics of the NP), where the semantics may correspond to a function: e.g., VP′(NP′) means the application of the semantics of the VP to the semantics of the NP. The numbers are not really part of the CFG — they are just there to identify different constituents.

```
S -> NP VP
```
VP′(NP′)
```
VP -> Vditrans NP1 NP2
```
(Vditrans′(NP1′))(NP2′)
```
VP -> Vtrans NP
```
Vtrans′(NP′)

```
VP -> Vintrans
```
Vintrans$'$
```
Vditrans -> gives
```
$\lambda x \lambda y \lambda z[\text{give}'(z, y, x)]$
```
Vtrans -> chases
```
$\lambda x \lambda y[\text{chase}'(y, x)]$
```
Vintrans -> barks
```
$\lambda z[\text{bark}'(z)]$
```
Vintrans -> sleeps
```
$\lambda w[\text{sleep}'(w)]$
```
NP -> Kitty
```
$k$
```
NP -> Lynx
```
$l$
```
NP -> Rover
```
$r$

The post-lecture exercises ask you to work through some examples using this fragment.

## 6.2 Logical representations in broad coverage grammars

It is possible to extend this style of compositional semantics (and various alternatives) so that some sort of representation is produced for all sentences of a language covered by a broad coverage grammar. However, the extent to which it is possible to do this within first order predicate calculus (FOPC) is a matter of debate, and even the most optimistic researcher would not claim that the representations currently being produced are fully adequate to capture the meaning of the expressions.

In some cases, a FOPC representation is possible by careful choice of representation. For example, to represent an adverbial modifier such as *loudly*, rather than make the adverbial take the verb as an argument (e.g., $\text{loud}'(\text{bark}'(r))$), it is common to introduce an extra variable for all verbs to represent the event.

(13) a Rover barked.

b $\exists e[\text{bark}'(e, r)]$

(14) a Rover barked loudly.

b $\exists e[\text{bark}'(e, r) \wedge \text{loud}'(e)]$

This can be roughly glossed as: 'there was a barking event, Rover did the barking, and the barking was loud'. The events are said to be *reified*: literally 'made into things'.

Another issue is that FOPC forces quantifiers to be in a particular scopal relationship, and this information is not (generally) overt in NL sentences.

Every dog chased a cat.

is ambiguous between:

$$\forall x[\text{dog}'(x) \implies \exists y[\text{cat}'(y) \wedge \text{chase}'(x, y)]]$$

and the less-likely, 'one specific cat' reading:

$$\exists y[\text{cat}'(y) \wedge \forall x[\text{dog}'(x) \implies \text{chase}'(x, y)]]$$

Some current NLP systems construct an underspecified representation which is neutral between these readings, if they represent quantifier scope at all. There are several different alternative formalisms for underspecification.

There are a range of natural language examples which FOPC cannot handle, however. FOPC only has two quantifiers (roughly corresponding to English *some* and *every*), but it is possible to represent some other English quantifiers in

FOPC, though representing numbers, for instance, is cumbersome. However, the quantifier *most* cannot be represented in a first order logic. The use of event variables does not help with the representation of adverbs such as *perhaps* or *maybe*, and there are also problems with the representation of modal verbs, like *can*. In these cases, there are known logics which can provide a suitable formalization, but the inference problem may not be tractable and the different logics proposed are not necessarily compatible.

There are other cases where the correct formalization is quite unclear. Consider sentences involving *bare plural* subjects, such as:

(15) Dogs are mammals.

(16) Birds fly.

(17) Ducks lay eggs.

(18) Voters rejected AV in 2011.

What quantifier could you use to capture the meaning of each of these examples? In some cases, researchers have argued that the correct interpretation must involve non-monotonicity or defaults: e.g., birds fly unless they are penguins, ostriches, baby birds ... But no fully satisfactory solution has been developed along these lines.

Another issue is that not all phrases have meanings which can be determined compositionally. For instance, *puce sphere* is compositional: it refers to something which is both puce and a sphere. In contrast, *red tape* may be compositional (as in 19), but may also refer to bureaucracy (as in 20), in which case it is a non-compositional *multiword expression* (MWE).

(19) She tied red tape round the parcel.

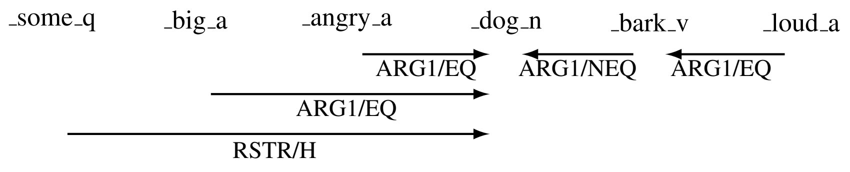
(20) The deputy head of department was dismayed at the endless red tape.

Clear cases of MWEs, like *red tape*, can be accounted for explicitly in the grammar (so *red tape* is treated as being ambiguous between the compositional reading and the MWE), but the problem is that many phrases are somewhere in between being an MWE and being fully compositional. Consider the phrase *real pleasure* (as in *it was a real pleasure to work with you*): this isn't non-compositional enough to be listed in most dictionaries, but is still a conventional expression particularly appropriate in certain social contexts. It appears that something else is necessary besides compositional semantics for such cases. A distributional approach (lecture 8) may help, but this is an active research issue.

Finally, although there is an extensive linguistic literature on formal semantics which describes logical representations for different constructions, the coverage is still very incomplete, even for well-studied languages such as English. Furthermore, the analyses which are suggested are often incompatible with the goals of broad-coverage parsing, which requires that ambiguity be avoided as much as possible. Overall, there are many unresolved puzzles in deciding on logical representations, so even approaches which do use logical representations are actually using some form of approximate meaning representation. Under many circumstances, it is better to see the logical representation as an annotation that captures some of the meaning, rather than a complete representation.

## 6.3 Dependency semantics

In the previous lecture, we saw some examples of syntactic dependency structure. It is also possible to produce semantic dependencies: Copestake (2009) describes an approach to semantic dependencies which is interconvertible with an underspecified logical representation. A very simple example is shown below, though note that some information encoded on the nodes is omitted. The leading underscores are notational equivalents to the ′ used previously and the _v etc notation is a broad indication of sense (so _bark_v cannot refer to tree bark, for instance).



This structure can be converted into the following logical form:

$$\_some\_q\ (x,\ \_big\_a(e1,x) \land \_angry\_a(e2,x) \land \_dog\_n(x),\ \_bark\_v(e3,x) \land \_loud\_a(e4,e3))$$

which in this case can be converted into FOPC:

$$\exists x\ [\ \_big\_a(e1,x) \land \_angry\_a(e2,x) \land \_dog\_n(x) \land \_bark\_v(e3,x) \land \_loud\_a(e4,e3)\ ]$$

This approach has been the basis for a considerable amount of research, including development of a representation which combines compositional semantics with (distributional) lexical semantics.

## 6.4 Inference

There are two distinct ways of thinking about reasoning and inference in language processing. The first can be thought of as inference on an explicit formally-represented knowledge base, while the second is essentially language-based. It is potentially possible to use theorem provers with either approach, although relatively few researchers currently do this: it is more common to use shallower, more robust, techniques.

**Inference on a knowledge base** This approach assumes that there is an explicit underlying knowledge base, which might be represented in FOPC or some other formal language. For instance, we might have a set of axioms such as:

$C(k, b)$
$C(r, b)$
$H(r)$
$U(k)$
$\forall x[C(k, x) \implies H(x)]$

There is also a link between the natural language terms appropriate for the domain covered and the constants and predicates in the knowledge base: e.g. *chase* might correspond to $C$, *happy* to $H$, *unhappy* to $U$, *Bernard* to $b$ and so on. The approaches to compositional semantics discussed above are one way to do this. Under these assumptions, a natural language question can be converted to an expression using the relevant predicates and answered either by direct match to the knowledge base or via inference using the meaning representation in the knowledge base. For instance *Is Bernard happy?* corresponds to querying the knowledge base with $H(b)$. As mentioned in Lecture 1, the properties of such a knowledge base can be exploited to reduce ambiguity. This is, of course, a trivial example: the complexity of the system depends on the complexity of the domain (and the way in which out-of-domain queries are handled) but in all cases the interpretation is controlled by the formal model of the domain.

Under these assumptions, the valid inferences are given by the knowledge base: language is just seen as a way of accessing that knowledge. This approach relies on the mapping being adequate for any way that the user might choose to phrase a question (e.g., *Is Kitty sad?* should be interpreted in the same way as *Is Kitty unhappy?*) and acquiring such mappings is an issue, although machine learning methods can be used, given the right sort of training data. But the most important difficulty is the limitations of the knowledge base representation itself. This sort of system works very well for cases where there are clear boundaries to the domain and the reasoning is tractable, which is the case for most types of spoken dialogue system. It may even turn out to be possible to approach mathematical knowledge in this way, or at least some areas of mathematics. However, although some researchers in the 1970s and 80s thought that it would be possible to extend this type of approach to common sense knowledge, few people would now seriously advocate this. Many researchers would also question whether it is helpful to think of human reasoning as operating in this way: i.e., as translating natural language into a symbolic mental representation.

**Language-based inference** The alternative way of thinking about inference is purely in terms of language: e.g., deciding whether one natural language statement follows from another. For instance,

Philadelphia is to the east of Pittsburgh.
Pittsburgh is to the west of Philadelphia.

The difference from the approach above is that the inference is entirely expressed in natural language and its validity is determined by human intuition, rather than validity in any particular logic. We may use logic as a way of helping us model these inferences correctly, but the basic notion of correctness is the human judgement, not logical correctness. If an explicit meaning representation is used, it is best seen as an annotation of the natural language that captures some of the meaning, not as a complete replacement for the text. Such language-based reasoning is not tied to a particular knowledge base or domain. With such an approach we can proceed without having a perfect meaning representation for a natural language statement, as long as we are prepared for the possibility that we will make some incorrect decisions. Since humans are not always right, this is acceptable in principle.

The Recognising Textual Entailment task, discussed below, is a clear example of this methodology, but implicitly or explicitly this is the most common approach in current NLP in general. It can be seen as underlying a number of areas of research, including question answering and semantic search, and is relevant to many others, including summarization. A major limitation, at least in the vast majority of the research so far, is that the inferences are made out of context. There will eventually be a requirement to include some form of modelling of the entities referred to, to represent anaphora for instance, although again this is probably best thought of as an annotation of the natural language text, rather than an entirely distinct domain model.

The distinction between these two approaches can be seen as fundamental in terms of the philosophy of meaning. However NLP research can and does combine the approaches. For example, a conversational agent will use an explicit domain model for some types of interaction and rely on language-based inference for others. While logic and compositional semantics is traditionally associated with the first approach, it can also be useful in the second.

**Lexical meaning and meaning postulates** Computational approaches to lexical meaning will be discussed in the following two lectures, but since interesting natural language inference requires some way of relating different lexemes, I will make some preliminary remarks here. Inference rules can be used to relate predicates that correspond to open class words. This is the classic way of representing lexical meaning in formal semantics within linguistics.[8] The standard example is:

$$\forall x[\text{bachelor}'(x) \leftrightarrow \text{man}'(x) \wedge \text{unmarried}'(x)]$$

For computational semantics, perhaps the best way of regarding meaning postulates is simply as one reasonable way of linking compositionally constructed semantic representations to a specific domain (in the first approach outlined above) or as relating lexemes to one another (as required by the second approach). In NLP, we're normally concerned with implication rather than definition:

$$\forall x[\text{bachelor}'(x) \implies \text{man}'(x) \wedge \text{unmarried}'(x)]$$

Such relationships may be approximate, as long as they are sufficiently accurate to be useful. For instance we'll see an example below that requires that *find* and *discover* are related, which could be stated as follows:

$$[\text{find}'(x, y, z) \implies \text{discover}'(x, y, z)]$$

This implies that all uses of *find* could be substituted by *discover*, which is not the case in general, although it may be completely adequate for some domains and work well enough in general to be useful.

## 6.5 Recognising Textual Entailment

A series of shared tasks concerning *Recognising Textual Entailment*, the RTE challenge, were carried out starting in 2004 (Dagan et al, 2005). The data consists of a series of texts, generally single sentences presented without context (labelled T below), and some hypothesis (H) which may or may not follow from that text.

(21) T: The girl was found in Drummondville earlier this month.
H: The girl was discovered in Drummondville.

[8]Generally, linguists don't actually write meaning postulates for open-class words, but this is the standard assumption about how meaning would be represented if anyone could be bothered to do it!

The task is to label the pairs as TRUE (when the entailment follows) or FALSE (when it doesn't follow, rather than when it's known to be an untrue statement) in a way which matches human judgements. The example above was labelled TRUE by the human annotator.

Examples of this sort can be dealt with using a logical form generated from a grammar with compositional semantics combined with inference. To show this in detail would require a lot more discussion of semantics than we had space for above, but I'll give a sketch of the approach here. Assume that the T sentence has logical form T′ and the H sentence has logical form H′. Then if T′ $\implies$ H′ we conclude TRUE, and otherwise we conclude FALSE. For example, the logical form for the T sentence above is approximately as shown below:

(22) T The girl was found in Drummondville earlier this month.

T′ $\exists x, u, e[\text{girl}'(x) \wedge \text{find}'(e, u, x) \wedge \text{in}'(e, \text{Drummondville}) \wedge \text{earlier-this-month}'(e)]$

The verb has the additional argument, $e$, corresponding to the event as discussed in §6.2 above. So the semantics can be glossed as: 'there was a finding event, the entity found was a girl, the finding event was in Drummondville, and the finding event happened earlier this month'. (The real representation would use a combination of predicates instead of earlier-this-month′, but that's not important for the current discussion.)

The hypothesis text is then:

(23) H The girl was discovered in Drummondville.

H′ $\exists x, u, e[\text{girl}'(x) \wedge \text{discover}'(e, u, x) \wedge \text{in}'(e, \text{Drummondville})]$

$A \wedge B \implies A$, which licences dropping *earlier this month*, so assuming a meaning postulate:

$$[\text{find}'(x, y, z) \implies \text{discover}'(x, y, z)]$$

the inference T′ $\implies$ H′ would go through.

An alternative technique is based on matching dependency structures. Instead of an explicit meaning postulate, a similarity metric can be used to relate *find* and *discover*. The similarity could be extracted from a lexical resource (next lecture), or from a corpus using distributional methods (lecture 8).

More robust methods can be used which do not require parsing of any type. The crudest technique is to use a bag-of-words method, analogous to that discussed for sentiment detection in lecture 1: if there is a large enough overlap between the words in T and H, the entailment goes though. Note that whether such a method works well or not crucially depends on the H texts: it would trivially be fooled by hypotheses like:

(24) H: The girl was discovered by Drummondville.

The RTE test set was actually deliberately constructed in such a way that word overlap works quite well.

Further examples (all discussed by Bos and Markert, 2005):

(25) T: Clinton's book is not a big seller here.

H: Clinton's book is a big seller.

a FALSE

(26) T: After the war the city was briefly occupied by the Allies and then was returned to the Dutch.

H: After the war, the city was returned to the Dutch.

a TRUE

(27) T: Four Venezuelan firefighters who were traveling to a training course in Texas were killed when their sport utility vehicle drifted onto the shoulder of a highway and struck a parked truck.

H: Four firefighters were killed in a car accident.

a TRUE

(28) T: Lyon is actually the gastronomic capital of France.
H: Lyon is the capital of France.
FALSE

(29) T: US presence puts Qatar in a delicate spot.
H: Qatar is located in a delicate spot.
a FALSE

(30) T: The first settlements on the site of Jakarta were established at the mouth of the Ciliwung, perhaps as early as the 5th century AD.
H: The first settlements on the site of Jakarta were established as early as the 5th century AD.
a TRUE (sic)

The post-lecture exercises suggest that you try and work out how a logical approach might handle (or fail to handle) these examples.

## 6.6 Further reading

J&M go into quite a lot of detail about compositional semantics including underspecification. A version of Wikipedia automatically annotated with underspecified semantic representations using the DELPH-IN ERG is available from `http://moin.delph-in.net/WikiWoods`: the distributional semantics experiments mentioned in lecture 8 make use of this.

Bos and Markert (2005)'s paper is available from: `http://www.let.rug.nl/bos/pubs/BosMarkert2006MLCW.pdf`

Bos and colleagues are developing the Groningen Meaning Bank (`http://gmb.let.rug.nl/`) a moderately large corpus annotated with logical representations: the initial annotation for this is based on automatic analysis with the C&C parser mentioned in lecture 5.

# 7 Lecture 7: Lexical semantics

Lexical semantics concerns word meaning. In the previous lecture, I briefly discussed the use of meaning postulates to represent the meaning of words. Linguistically and philosophically, any such approach has clear problems. Take the bachelor example: is the current Pope a bachelor? Technically presumably yes, but *bachelor* seems to imply someone who could be married: it's a strange word to apply to the Pope under current assumptions about celibacy. Meaning postulates are also too unconstrained: I could construct a predicate 'bachelor-weds-thurs' to correspond to someone who was unmarried on Wednesday and married on Thursday, but this isn't going to correspond to a word in any natural language. In any case, very few words are as simple to define as *bachelor*: consider how you might start to define *table*, *tomato* or *thought*, for instance.[9]

Rather than try and build complete and precise representations of word meaning therefore, computational linguists work with partial or approximate representations. The *find/discover* implication in the last lecture is an approximate representation, because expanding the rule to stipulate the exact conditions under which *discover* can be used instead of *find* is impossible (at least in any practical system) and individual speakers may well have different intuitions. In this lecture, I will discuss the classical lexical semantic relations and also polysemy. I will also introduce the topic of *grounding* symbolic representations. In the following lectures, we will look at an alternative approach to representing lexical meaning.

## 7.1 Hyponymy: IS-A

Hyponymy is the classical IS-A relation: e.g. *dog* is a *hyponym* of *animal* and *animal* is the *hypernym* of *dog*. To be more precise, the relevant sense of *dog* is the hyponym of the relevant sense of *animal* (*dog* can also be a verb or used in a metaphorical and derogatory way to refer to a human). As nearly everything said in this lecture is about word senses rather than words, I will avoid explicitly qualifying all statements in this way, but this should be globally understood. In turn, *dog* is the hypernym of *spaniel*, *terrier* and so on. Hyponyms can be arranged into *taxonomies*: classically these are tree-structured: i.e., each term has only one *hypernym*.

Hyponymy relates to ideas which have been formalised in description logics and used in ontologies and the semantic web. However, formalising the linguistic notion of hyponymy is difficult. Despite the fact that hyponymy is by far the most important meaning relationship assumed in NLP, many questions arise which don't have very good answers:

1. What classes of words can be categorised by hyponymy? Some nouns, classically biological taxonomies, but also human artefacts, professions etc work reasonably well. Abstract nouns, such as *truth*, don't really work very well (they are either not in hyponymic relationships at all, or very shallow ones). Some verbs can be treated as being hyponyms of one another — e.g. *murder* is a hyponym of *kill*, but this is not nearly as clear as it is for concrete nouns. Event-denoting nouns are similar to verbs in this respect. Hyponymy is almost useless for adjectives.

2. Do differences in quantisation and individuation matter? For instance, is *chair* a hyponym of *furniture*? is *beer* a hyponym of *drink*? is *coin* a hyponym of *money*?

3. Is multiple inheritance allowed? Intuitively, multiple parents might be possible: e.g. *coin* might be *metal* (or *object*?) and also *money*. Artefacts in general can often be described either in terms of their form or their function.

4. What should the top of the hierarchy look like? The best answer seems to be to say that there is no single top but that there are a series of hierarchies.

Despite this lack of clarity, hyponymy relationships are a good source of inference rules, at least in the language-based conception of inference described in the previous lecture.

[9]There has been a court case that hinged on the precise meaning of *table* and also one that depended on whether tomatoes were fruits or vegetables.

## 7.2 Other lexical semantic relations

**Meronymy** i.e., PART-OF

The standard examples of meronymy apply to physical relationships: e.g., *arm* is part of a *body* (*arm* is a *meronym* of *body*); *steering wheel* is a meronym of *car*. Note the distinction between 'part' and 'piece': if I attack a car with a chainsaw, I get pieces rather than parts!

**Synonymy** i.e., two words with the same meaning (or nearly the same meaning)

True synonyms are relatively uncommon: most cases of true synonymy are correlated with dialect differences (e.g., *eggplant* / *aubergine*, *boot* / *trunk*). Often synonymy involves register distinctions, slang or jargons: e.g., *policeman*, *cop*, *rozzer* ... Near-synonyms convey nuances of meaning: *thin*, *slim*, *slender*, *skinny*.

**Antonymy** i.e., opposite meaning

Antonymy is mostly discussed with respect to adjectives: e.g., *big*/*little*, though it's only relevant for some classes of adjectives.

## 7.3 WordNet

WordNet is the main resource for lexical semantics for English that is used in NLP — primarily because of its very large coverage and the fact that it's freely available. WordNets are under development for many other languages, though so far none are as extensive as the original.

The primary organisation of WordNet is into *synsets*: synonym sets (near-synonyms). To illustrate this, the following is part of what WordNet returns as an 'overview' of *red*:

```
wn red -over

Overview of adj red

The adj red has 6 senses (first 5 from tagged texts)

1. (43) red, reddish, ruddy, blood-red, carmine,
cerise, cherry, cherry-red, crimson, ruby, ruby-red,
scarlet -- (having any of numerous bright or strong
colors reminiscent of the color of blood or cherries
or tomatoes or rubies)
2. (8) red, reddish -- ((used of hair or fur) of a
reddish brown color; "red deer"; reddish hair")
```

Nouns in WordNet are organised by hyponymy, as illustrated by the fragment below:

```
Sense 6
big cat, cat
       => leopard, Panthera pardus
           => leopardess
           => panther
       => snow leopard, ounce, Panthera uncia
       => jaguar, panther, Panthera onca, Felis onca
       => lion, king of beasts, Panthera leo
           => lioness
           => lionet
       => tiger, Panthera tigris
           => Bengal tiger
           => tigress
       => liger
```

```
        => tiglon, tigon
        => cheetah, chetah, Acinonyx jubatus
        => saber-toothed tiger, sabertooth
            => Smiledon californicus
            => false saber-toothed tiger
```

Taxonomies have also been extracted from machine-readable dictionaries: Microsoft's MindNet is the best known example. There has been considerable work on extracting taxonomic relationships from corpora, including some aimed at automatically extending WordNet.

## 7.4 Using lexical semantics

The most commonly used lexical relations are hyponymy and (near-)synonymy. Hyponymy relations can be used in many ways, for instance:

- Semantic classification: e.g., for selectional restrictions (e.g., the object of *eat* has to be something edible) and for named entity recognition.
- Shallow inference: 'X murdered Y' implies 'X killed Y' etc (as discussed in the previous lecture).
- Back-off to semantic classes in some statistical approaches (for instance, WordNet classes can be used in document classification).
- Word-sense disambiguation.
- Query expansion for information retrieval: if a search doesn't return enough results, one option is to replace an over-specific term with a hypernym.

Synonymy or near-synonymy is relevant for some of these reasons and also for generation. (However dialect and register haven't been investigated much in NLP, so the possible relevance of different classes of synonym for customising text hasn't really been looked at.)

## 7.5 Collocation

Informally, a collocation is a group of two or more words that occur together more often than would be expected by chance (there are other definitions — this is not really a precise notion). Collocation includes *multiword expressions* (i.e., conventional phrases that might be listed in a dictionary (e.g., *bass guitar* and *striped bass* — a type of fish: as this example indicates, collocations allow disambiguation). Other examples of collocation include phrases like *heavy rain*: *heavy* is used to indicate high quantity with respect to precipitation words (rain, snow, hail etc) and is much more common than other magnitude terms with such words.

The term collocation is sometimes restricted to the situation where there is a syntactic relationship between the words. J&M (second edition) define collocation as a position-specific relationship (in contrast to *bag-of-words*, where position is ignored) but this is not a standard definition.

Collocation will be discussed further in the next lecture.

## 7.6 Polysemy

Polysemy refers to the state of a word having more than one sense: the standard example is *bank* (river bank) vs *bank* (financial institution).

This is *homonymy* — the two senses are unrelated (not entirely true for *bank*, in fact, but historical relatedness isn't important — it's whether ordinary speakers of the language feel there's a relationship). Homonymy is the most obvious case of polysemy, but is relatively infrequent compared to uses which have different but related meanings, such as *bank* (financial institution) vs *bank* (in a casino).

If polysemy were always homonymy, word senses would be discrete: two senses would be no more likely to share characteristics than would morphologically unrelated words. But most senses are actually related. Regular or systematic polysemy (zero derivation, as mentioned in the morphology lecture) concerns related but distinct usages of words, often with associated syntactic effects. For instance, *strawberry, cherry* (fruit / plant), *rabbit, turkey, halibut* (meat / animal), *tango, waltz* (dance (noun) / dance (verb)).

There are a lot of complicated issues in deciding whether a word is polysemous or simply general/vague. For instance, *teacher* is intuitively general between male and female teachers rather than ambiguous, but giving good criteria as a basis of this distinction is difficult. Dictionaries are not much help, since their decisions as to whether to split a sense or to provide a general definition are very often contingent on external factors such as the size of the dictionary or the intended audience, and even when these factors are relatively constant, lexicographers often make different decisions about whether and how to split up senses.

## 7.7 Word sense disambiguation

NL applications that involve semantics have to take account of word senses, explicitly or implicitly. Classically, explicit word sense disambiguation (WSD) was regarded as a separate task, but, as we will discuss, this is no longer generally seen as appropriate.

In limited domains, sense ambiguity is usually not a problem because most senses of words are irrelevant in the domain, and the remaining cases can be disambiguated via domain knowledge. For large coverage text processing, polysemy is still a serious issue, however.

WSD up to the early 1990s was mostly done by hand-constructed rules. Dahlgren investigated WSD in a fairly broad domain in the 1980s. Her approach depended on:

- frequency
- collocations
- selectional restrictions/preferences

These were still the usual knowledge sources for later approaches, although other options include machine readable dictionaries and Wikipedia disambiguation pages.

While supervised learning for WSD has been tried it requires a sense-tagged corpus, which is extremely time-consuming to construct systematically (examples are the Semcor and SENSEVAL corpora, but both are really too small). Agreement between annotators was poor. Often experiments have been done with a small set of words which can be sense-tagged by the experimenter, but supervised learning techniques do not carry over well from one corpus to another. Semi-supervised or unsupervised learning is preferable (e.g., Yarowsky's approach, discussed in previous versions of these notes and in J+M) .

Initially, most of the statistical or machine-learning techniques were evaluated on homonyms: these are relatively easy to disambiguate. Yarowsky claimed 95% disambiguation, but this doesn't mean there is high precision on all words.

In order to experiment with WSD as a standalone module, there has to be a standard: most commonly WordNet was used, because for decades it was the only extensive modern resource for English that was freely available. This was controversial, because WordNet has a very fine granularity of senses and the senses often overlap. Various WSD 'competitions' were organised (SENSEVAL).

However WSD is now out of fashion as a standalone NLP subtask: these are several reasons for this, but the most important is that it seems unhelpful to consider it in isolation from applications. WSD is important in speech synthesis, for example, but only for a relatively small number of words where the sense distinction indicates a difference in pronunciation (e.g., *bass* but not *bank* or *plant*). In SMT, the necessary disambiguation happens as part of the general model rather than being a separate step, even conceptually. In any case, for MT, the word senses (or uses) which need to be distinguished depend on the language pair, and may well not be ambiguous to a native speaker. For instance, German *Wald* may be translated as *forest* or *wood*, but the distinction is not seen as an ambiguity by a native German speaker.

There have also been some attempts at automatic *sense induction*, determining the clusters of usages in texts that correspond to senses. In principle, this is a very good idea, since the whole notion of a word sense is fuzzy: word senses can even be argued to be artefacts of dictionary publishing. However, sense induction has not been much explored as a standalone task, because of the difficulty of evaluation, though it could be considered as an inherent part of statistical approaches to MT and, indeed, of deep learning approaches which incorporate distributions/embeddings (see later lectures).

## 7.8 Grounding

The idea of *grounding* is simple and intuitive, but relates to some serious and complex philosophical debate concerning Artificial Intelligence. So far in this course, our notion of meaning has been purely symbolic: we expect our systems to behave correctly based purely on manipulating symbols of natural language, or logical or other abstract symbols which we have introduced. We can only talk about meaning in terms of relationships between terms. If we did, somehow, manage to completely define the word *table*, this would still not mean that a system with that knowledge could recognise a real table.

To be continued . . .

## 7.9 Further reading and background

WordNet is freely downloadable: the website has pointers to several papers which provide a good introduction. Recent initiatives have made many WordNets for other languages freely available: many are cross-linked to each other (generally via the English WordNet).

# 8 Lecture 8: Distributional semantics

The notes from this lecture are partly based on slides written by Aurélie Herbelot.

Distributional semantics refers to a family of techniques for representing word (and phrase) meaning based on (linguistic) contexts of use. Consider the following examples (from the BNC):

> it was authentic scrumpy, rather sharp and very strong
> we could taste a famous local product — scrumpy
> spending hours in the pub drinking scrumpy

Even if you don't know the word *scrumpy*, you can get a good idea of its meaning from contexts like this. Humans typically learn word meanings from context rather than explicit definition: sometimes these meanings are perceptually grounded (e.g., someone gives you a glass of scrumpy), sometimes not.

It is to a large extent an open question how word meanings are represented in the brain.[10] Distributional semantics uses linguistic context to represent meaning and this is likely to have some relationship to mental meaning representation. It can only be a partial representation of lexical meaning (perceptual grounding is clearly important too, and more-or-less explicit definition may also play a role) but, as we'll see, distributions based on language alone are good enough for some tasks. In these models, meaning is seen as a space, with dimensions corresponding to elements in the context (*features*). Computational techniques generally use vectors to represent the space and the terms *semantic space models* and *vector space models* are sometimes used instead of distributional semantics. [11] Schematically:

| | feature$_1$ | feature$_2$ | ... | feature$_n$ |
|---|---|---|---|---|
| word$_1$ | $f_{1,1}$ | $f_{2,1}$ | | $f_{n,1}$ |
| word$_2$ | $f_{1,2}$ | $f_{2,2}$ | | $f_{n,2}$ |
| ... | | | | |
| word$_m$ | $f_{1,m}$ | $f_{2,m}$ | | $f_{n,m}$ |

There are many different possible notions of **features:** co-occur with word$_n$ in some window, co-occur with word$_n$ as a syntactic dependent, occur in paragraph$_n$, occur in document$_n$ ...

The main use of distributional models has been in measuring similarity between pairs of words: such measurements can be exploited in a variety of ways to model language: similarity measurements allow clustering of words so can be used as a way of getting at unlabelled semantic classes. In the rest of this lecture, we will first discuss some possible models illustrating the choices that must be made when designing a distributional semantics system and go through a step-by-step example. We'll then look at some real distributions and then describe how distributions can be used in measuring similarity between words. We'll briefly describe how polysemy affects distributions and conclude with a discussion of the relationship between distributional models and the classic lexical semantic relations of synonymy, antonymy and hyponymy.

## 8.1 Models

Distributions are vectors in a multidimensional semantic space, that is, objects with a magnitude (length) and a direction. The *semantic space* has dimensions which correspond to possible contexts. For our purposes, a distribution can be seen as a point in that space (the vector being defined with respect to the origin of that space). e.g., *cat* [...dog 0.8, eat 0.7, joke 0.01, mansion 0.2, zebra 0.1...], where 'dog', 'joke' and so on are the dimensions.

**Context:**
Different models adopt different notions of context:

- Word windows (unfiltered): $n$ words on either side of the item under consideration (unparsed text).
  **Example:** n=2 (5 words window):

[10]In fact, it's common to talk about concepts rather than word meanings when discussing mental representation, but while some researchers treat some (or all) concepts as equivalent to word senses (or phrases), others think they are somehow distinct.

[11]Vector space models in IR are directly related to distributional models. Some more complex techniques use tensors of different orders rather than simply using vectors, but we won't discuss them in this lecture. Embeddings used in deep learning are a form of dimensionality-reduced distributional model, we will discuss this explicitly in the next lecture.

... the prime **minister** acknowledged that ...

- Word windows (filtered): $n$ words on either side of the item under consideration (unparsed text). Some words will be in a stoplist and not considered part of the context. It is common to put function words and some very frequent content words in the stoplist. The stoplist may be constructed manually, but often the corpus is POS-tagged and only certain POS tags are considered part of the context.
**Example:** n=2 (5 words window), underlined words are in the stop list.

  ... <u>the</u> prime **minister** acknowledged <u>that</u> ...

- Lexeme windows: as above, but a morphological processor is applied first that converts the words to their stems.
- Dependencies: syntactic or semantic. The corpus is converted into a list of directed links between heads and dependants (see §5.1). The context for a lexeme is constructed based on the dependencies it belongs to. The length of the dependency path varies according to the implementation.

Example of distributions for *word* comparing unparsed and parsed data:

| **unparsed data** | | **parsed data** | |
|---|---|---|---|
| meaning_n | ipa_n | or_c+phrase_n | and_c+deed_n |
| derive_v | verb_n | and_c+phrase_n | meaning_n+of_p |
| dictionary_n | mean_v | syllable_n+of_p | from_p+language_n |
| pronounce_v | hebrew_n | play_n+on_p | pron_rel_+utter_v |
| phrase_n | usage_n | etymology_n+of_p | for_p+word_n |
| latin_j | literally_r | portmanteau_n+of_p | in_p+sentence_n |

**Context weighting:**
Different models use different methods of weighting the context elements:

- Binary model: if context $c$ co-occurs with word $w$, value of vector $\vec{w}$ for dimension $c$ is 1, 0 otherwise.

  ... [a long long long **example** for a distributional semantics] model... (n=4)

  ... {a 1} {dog 0} {long 1} {sell 0} {semantics 1}...

- Basic frequency model: the value of vector $\vec{w}$ for dimension $c$ is the number of times that context $c$ co-occurs with $w$.

  ... [a long long long **example** for a distributional semantics] model... (n=4)

  ... {a 2} {dog 0} {long 3} {sell 0} {semantics 1}...

- Characteristic model: the weights given to the vector components express how *characteristic* a given context is for $w$. Functions used include:
  - Pointwise Mutual Information (PMI), with or without discounting factor.

    $$pmi_{wc} = log(\frac{f_{wc} * f_{total}}{f_w * f_c}) \tag{31}$$

    where $f_{wc}$ is the frequency with which word $w$ occurs in context $c$, $f_{total}$ is the total frequency of all the possible contexts, $f_w$ is the frequency of the word $w$ and $f_c$ is the overall frequency of the context item. i.e., if we use words as dimensions, $f_{wc}$ is the frequency with which word $w$ and word $c$ cooccur, $f_c$ is the overall frequency of word $c$ and $f_{total}$ is the total frequency of all the context words.
  - Positive PMI (PPMI): as PMI but 0 if PMI $<$ 0.
  - Derivatives such as Mitchell and Lapata's (2010) weighting function (PMI without the log).

Most work uses some form of characteristic model in order to give most weight to frequently cooccuring features, but allowing for the overall frequency of the terms in the context. Note that PMI is one of the measures used for finding collocations (see previous lecture): the distributional models can be seen as combining the collocations for words.

**Semantic space:**

Once the contexts and weights have been decided on, models also vary in which elements are included in the final vectors: i.e., what the total semantic space consists of. The main options are as follows (with positive and negative aspects indicated):

- Entire vocabulary. All the information is included – even rare contexts may be important. However using many dimensions (many hundreds of thousands) makes it slow. The dimensions will include a lot of noise: e.g. *002.png—thumb—right—200px—graph_n*
- Top $n$ words with highest frequencies. This is more efficient (5000-10000 dimensions are usual) and only 'real' words will be included. However, the model may miss infrequent but relevant contexts.
- Singular Value Decomposition and other dimensionality reduction techniques. SVD was used in LSA – Landauer and Dumais (1997). The number of dimensions is reduced by exploiting redundancies in the data. A new dimension might correspond to a generalisation over several of the original dimensions (e.g. the dimensions for *car* and *vehicle* are collapsed into one). This can be very efficient (200-500 dimensions are often used) and it should capture generalisations in the data. The problem is that SVD matrices are not interpretable. Arguably, this is a theoretical problem, but it is more obviously a practical problem: using SVD makes it impossible to debug the feature space by manual inspection. It is therefore unsuitable for initial experiments.

But there are several other variants.

## 8.2 Getting distributions from text

In this section, we illustrate a word-window model using PMI weightings with a stop list consisting of all closed-class words. We use the following example text (from Douglas Adams, *Mostly harmless*):

> The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong it usually turns out to be impossible to get at or repair.

Naturally, real distributions are calculated from much larger corpora!

**Dimensions (all the open class words in the text, without lemmatization):**

| | | |
|---|---|---|
| difference | major | usually |
| get | possibly | wrong |
| go | repair | |
| goes | thing | |
| impossible | turns | |

**Overall frequency counts (needed for the PMI calculations):**

| | | |
|---|---|---|
| difference 1 | major 1 | usually 1 |
| get 1 | possibly 2 | wrong 4 |
| go 3 | repair 1 | |
| goes 1 | thing 3 | |
| impossible 1 | turns 1 | |

**Conversion into 5-word windows:**

- ∅ ∅ **the** major difference

- ∅ the **major** difference between
- the major **difference** between a
- major difference **between** a thing
- ...

**Context windows for *wrong***:
The major difference between a thing that [might go wrong and a] thing that cannot [possibly go wrong is that] when a thing that cannot [possibly go [wrong goes wrong] it usually] turns out to be impossible to get at or repair.
**Distribution for *wrong* (raw frequencies)**

| | | |
|---|---|---|
| difference 0 | major 0 | usually 1 |
| get 0 | possibly 2 | wrong 2 |
| go 3 | repair 0 | |
| goes 2 | thing 0 | |
| impossible 0 | turns 0 | |

Note that the single token of *goes* is counted twice, because it occurs with two different tokens of *wrong*.
**Distribution for *wrong* (PPMI)**:

| | | |
|---|---|---|
| difference 0 | major 0 | usually 0.70 |
| get 0 | possibly 0.70 | wrong 0.40 |
| go 0.70 | repair 0 | |
| goes 1 | thing 0 | |
| impossible 0 | turns 0 | |

For instance, for the context *possibly*, $f_{wc}$ is 2 (as in the raw frequency distribution table), $f_c$ is the total count of *possibly* which is also 2 (as in the overall frequency count table), $f_w$ is 4 (again, as in the overall frequency count table) and $f_{total}$ is 20 (i.e., the sum of the frequencies in the overall frequency count table), so PMI is log(5).

## 8.3 Real distributions

In this section we give some more realistic examples of distributions, which have been derived using a methodology that we have adopted for our own research. The corpus (WikiWoods: Flickinger et al 2010: `http://moin.delph-in.net/WikiWoods`) is based on a dump of the entire English Wikipedia parsed with the English Resource Grammar (Flickinger, 2000, see lecture 5) and converted into semantic dependencies (see Lecture 6), though the results would be expected to be similar with syntactic dependencies. The dependencies considered include:

- For nouns: head verbs (+ any other argument of the verb), modifying adjectives, head prepositions (+ any other argument of the preposition).
  *e.g. cat: chase_v+mouse_n, black_a, of_p+neighbour_n*
- For verbs: arguments (NPs and PPs), adverbial modifiers.
  *e.g. eat: cat_n+mouse_n, in_p+kitchen_n, fast_a*
- For adjectives: modified nouns; rest as for nouns (assuming intersective composition).
  *e.g. black: cat_n, chase_v+mouse_n*

The model uses a semantic space of the top 100,000 contexts (because we wanted to include the rare terms) with a variant of PMI (Bouma 2007) for weighting:

$$pmi_{wc} = \frac{log(\frac{f_{wc}*f_{total}}{f_w*f_c})}{-log(\frac{f_{wc}}{f_{total}})} \tag{32}$$

An example noun, *language*:

| | | |
|---|---|---|
| 0.54::other+than_p()+English_n | 0.45::foreign_a | 0.42::and_c+culture_n |
| 0.53::English_n+as_p() | 0.45::germanic_a | 0.41::arabic_a |
| 0.52::English_n+be_v | 0.44::German_n+be_v | 0.41::dialects_n+of_p() |
| 0.49::english_a | 0.44::of_p()+instruction_n | 0.40::part_of_rel_+speak_v |
| 0.48::and_c+literature_n | 0.44::speaker_n+of_p() | 0.40::percent_n+speak_v |
| 0.48::people_n+speak_v | 0.42::generic_entity_rel_+speak_v | 0.39::spanish_a |
| 0.47::French_n+be_v | 0.42::pron_rel_+speak_v | 0.39::welsh_a |
| 0.46::Spanish_n+be_v | 0.42::colon_v+English_n | 0.39::tonal_a |
| 0.46::and_c+dialects_n | 0.42::be_v+English_n | |
| 0.45::grammar_n+of_p() | 0.42::language_n+be_v | |

An example adjective, *academic*:

| | | |
|---|---|---|
| 0.52::Decathlon_n | 0.37::journal_n+be_v | 0.34::standard_n |
| 0.51::excellence_n | 0.37::vocational_a | 0.34::at_p()+institution_n |
| 0.45::dishonesty_n | 0.37::student_n+achieve_v | 0.34::career_n |
| 0.45::rigor_n | 0.36::athletic_a | 0.34::Career_n |
| 0.43::achievement_n | 0.36::reputation_n+for_p() | 0.33::dress_n |
| 0.42::discipline_n | 0.35::regalia_n | 0.33::scholarship_n |
| 0.40::vice_president_n+for_p() | 0.35::program_n | 0.33::prepare_v+student_n |
| 0.39::institution_n | 0.35::freedom_n | 0.33::qualification_n |
| 0.39::credentials_n | 0.35::student_n+with_p() | |
| 0.38::journal_n | 0.35::curriculum_n | |

Corpus choice is another parameter that has to be considered in building models. Some research suggests that one should use as much data as possible. Some commonly used corpora:

- British National Corpus (BNC): 100 m words
- Wikipedia dump used in Wikiwoods: 897 m words
- UKWac (obtained from web-crawling): 2 bn words

In general, more data does give better models *but* the domain has to be considered: for instance, huge corpora of financial news won't give models that work well with other text types. Furthermore, more data is not realistic from a psycholinguistic point of view. We encounter perhaps 50,000 words a day (although nobody actually has good estimates of this!) so the BNC, which is very small by the standards of current experiments, corresponds to approximately 5 years' exposure.

It is clear that sparse data is a problem for relatively rare words. For instance, consider the following distribution for *unicycle*, as obtained from Wikiwoods:

| | | |
|---|---|---|
| 0.45::motorized_a | 0.19::man_n+on_p() | 0.13::tall_a |
| 0.40::pron_rel_+ride_v | 0.19::on_p()+stage_n | 0.12::fast_a |
| 0.24::for_p()+entertainment_n | 0.19::position_n+on_p() | 0.11::red_a |
| 0.24::half_n+be_v | 0.17::slip_v | 0.07::come_v |
| 0.24::unwieldy_a | 0.16::and_c+1_n | 0.06::high_a |
| 0.23::earn_v+point_n | 0.16::autonomous_a | |
| 0.22::pron_rel_+crash_v | 0.16::balance_v | |

Note that humans exploit a lot more information from context and can get a good idea of word meanings from a small number of examples.

Distributions are generally constructed without any form of sense disambiguation. The semantic space can be thought of as consisting of subspaces for different senses, with homonyms (presumably) relatively distinct. For instance, consider the following distribution for *pot*:

| | | |
|---|---|---|
| 0.57::melt_v | 0.33::amount_n+in_p() | 0.29::pot_n+and_c |
| 0.44::pron_rel_+smoke_v | 0.33::ceramic_a | 0.28::bottom_n+of_p() |
| 0.43::of_p()+gold_n | 0.33::hot_a | 0.28::of_p()+flower_n |
| 0.41::porous_a | 0.32::boil_v | 0.28::of_p()+water_n |
| 0.40::of_p()+tea_n | 0.31::bowl_n+and_c | 0.28::food_n+in_p() |
| 0.39::player_n+win_v | 0.31::ingredient_n+in_p() | |
| 0.39::money_n+in_p() | 0.30::plant_n+in_p() | |
| 0.38::of_p()+coffee_n | 0.30::simmer_v | |

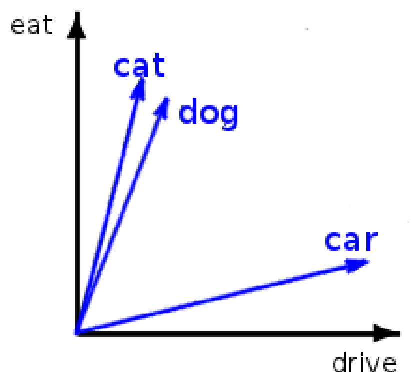
Finally, note that distributions contain many contexts which arise from multiword expressions of various types, and these often have high weights. The distribution of *pot* contains several examples, as does the following distribution for *time*:

| | | |
|---|---|---|
| 0.46::of_p()+death_n | 0.40::pron_rel_+spend_v | 0.37::country_n+at_p() |
| 0.45::same_a | 0.39::sand_n+of_p() | 0.37::age_n+at_p() |
| 0.45::1_n+at_p(temp) | 0.39::pron_rel_+waste_v | 0.37::space_n+and_c |
| 0.45::Nick_n+of_p() | 0.38::place_n+around_p() | 0.37::in_p()+career_n |
| 0.42::spare_a | 0.38::of_p()+arrival_n | 0.37::world_n+at_p() |
| 0.42::playoffs_n+for_p() | 0.38::of_p()+completion_n | |
| 0.42::of_p()+retirement_n | 0.37::after_p()+time_n | |
| 0.41::of_p()+release_n | 0.37::of_p()+arrest_n | |

To sum up, there is a wide range of choice in constructing distributional models. Manually examining the characteristic contexts gives us a good idea of how sensible different weighting measures are, for instance, but we need to look at how distributions are actually used to evaluate how well they model meaning.

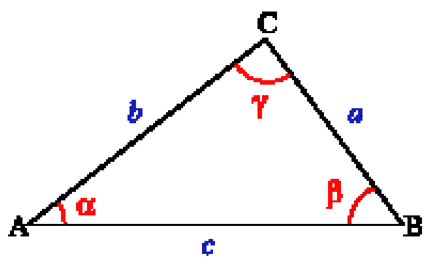
## 8.4 Similarity

Calculating similarity in a distributional space is done by calculating the distance between the vectors.



The most common method is *cosine similarity*.

- Law of cosines: $c^2 = a^2 + b^2 - 2ab\cos\gamma$



Cosine similarity:

$$\frac{\sum v1_k * v2_k}{\sqrt{\sum v1_k^2} * \sqrt{\sum v2_k^2}} \tag{33}$$

This measure calculates the angle between two vectors and is therefore length-independent. This is important, as frequent words have longer vectors than less frequent ones.

The following examples should give some idea of the scales of similarity found:

| | |
|---|---|
| house – building | 0.43 |
| gem – jewel | 0.31 |
| capitalism – communism | 0.29 |
| motorcycle – bike | 0.29 |
| test – exam | 0.27 |
| school – student | 0.25 |
| singer – academic | 0.17 |
| horse – farm | 0.13 |
| man –accident | 0.09 |
| tree – auction | 0.02 |
| cat –county | 0.007 |

Note that perfect similarity gives a cosine of 1, but that even near-synonyms like *gem* and *jewel* have much lower cosine similarity.

Words most similar to *cat*, as chosen from the 5000 most frequent nouns in Wikipedia.

| | | | |
|---|---|---|---|
| 1 cat | 0.29 goat | 0.24 squirrel | 0.22 mammal |
| 0.45 dog | 0.28 snake | 0.24 dragon | 0.21 bat |
| 0.36 animal | 0.28 bear | 0.24 frog | 0.21 duck |
| 0.34 rat | 0.28 man | 0.23 baby | 0.21 cattle |
| 0.33 rabbit | 0.28 cow | 0.23 child | 0.21 dinosaur |
| 0.33 pig | 0.26 fox | 0.23 lion | 0.21 character |
| 0.31 monkey | 0.26 girl | 0.23 person | 0.21 kid |
| 0.31 bird | 0.26 sheep | 0.23 pet | 0.21 turtle |
| 0.30 horse | 0.26 boy | 0.23 lizard | 0.20 robot |
| 0.29 mouse | 0.26 elephant | 0.23 chicken | |
| 0.29 wolf | 0.25 deer | 0.22 monster | |
| 0.29 creature | 0.25 woman | 0.22 people | |
| 0.29 human | 0.25 fish | 0.22 tiger | |

This notion of similarity is very broad. It includes synonyms, near-synonyms, hyponyms, taxonomical siblings, antonyms and so on. But it does correlate with a psychological reality. One of the favourite tests of the distributional semantics community is the calculation of rank correlation between a distributional similarity system and human judgements on the Miller & Charles (1991) test set shown below:

| | | |
|---|---|---|
| 3.92 automobile-car | 3.05 bird-cock | 0.84 forest-graveyard |
| 3.84 journey-voyage | 2.97 bird-crane | 0.55 monk-slave |
| 3.84 gem-jewel | 2.95 implement-tool | 0.42 lad-wizard |
| 3.76 boy-lad | 2.82 brother-monk | 0.42 coast-forest |
| 3.7 coast-shore | 1.68 crane-implement | 0.13 cord-smile |
| 3.61 asylum-madhouse | 1.66 brother-lad | 0.11 glass-magician |
| 3.5 magician-wizard | 1.16 car-journey | 0.08 rooster-voyage |
| 3.42 midday-noon | 1.1 monk-oracle | 0.08 noon-string |
| 3.11 furnace-stove | 0.89 food-rooster | |
| 3.08 food-fruit | 0.87 coast-hill | |

The human similarity results can be replicated: the Miller & Charles experiment is a re-run of Rubenstein & Goodenough (1965): the correlation coefficient between them is 0.97. A good distributional similarity system can have a correlation of 0.8 or better with the human data (although there is a danger the reported results are unreasonably high, because this data has been used in so many experiments).

Another frequently used dataset is the TOEFL (Test of English as a Foreign Language) synonym test. For example:

| | | |
|---|---|---|
| Stem: | | levied |
| Choices: | (a) | imposed |
| | (b) | believed |
| | (c) | requested |
| | (d) | correlated |
| Solution: | (a) | imposed |

Non-native English speakers are reported to average around 65% on this test (US college applicants): the best corpus-based results are 100% (Bullinaria and Levy, 2012) . But note that the authors who got this result suggest the test is not very reliable — one reason is probably that the data includes some extremely rare words.

Similarity measures can be applied as a type of backoff technique in a range of tasks. For instance, in sentiment analysis (discussed in lecture 1), an initial bag of words acquired from the training data can be expanded by including distributionally similar words.

## 8.5 Distributions and classic lexical semantic relationships

Distributions are a usage representation: they are corpus-dependent, culture-dependent and register-dependent. Synonyms with different registers often don't have a very high similarity. For example, the similarity between *policeman* and *cop* is 0.23 and the reason for this relatively low number becomes clear if one examines the highest weighted features:

| | | | |
|---|---|---|---|
| **policeman** | 0.36::uniformed_a | 0.28::incompetent_a | 0.26::on_p()+duty_n |
| 0.59::ball_n+poss_rel | 0.35::uniform_n+poss_rel | 0.28::pron_rel_+shoot_v | 0.25::salary_n+poss_rel |
| 0.48::and_c+civilian_n | 0.35::civilian_n+and_c | 0.28::hat_n+poss_rel | 0.25::on_p()+horseback_n |
| 0.42::soldier_n+and_c | 0.31::iraqi_a | 0.28::terrorist_n+and_c | 0.25::armed_a |
| 0.41::and_c+soldier_n | 0.31::lot_n+poss_rel | 0.27::and_c+crowd_n | 0.24::and_c+nurse_n |
| 0.38::secret_a | 0.31::chechen_a | 0.27::military_a | 0.24::job_n+as_p() |
| 0.37::people_n+include_v | 0.30::laugh_v | 0.27::helmet_n+poss_rel | 0.24::open_v+fire_n |
| 0.37::corrupt_a | 0.29::and_c+criminal_n | 0.27::father_n+be_v | |

| | | | |
|---|---|---|---|
| **cop** | 0.33::pron_rel_+call_v | 0.27::investigate_v+murder_n | 0.23::and_c+interference_n |
| 0.45::crooked_a | 0.32::funky_a | 0.26::on_p()+force_n | 0.23::arrive_v |
| 0.45::corrupt_a | 0.32::bad_a | 0.25::parody_n+of_p() | 0.23::and_c+detective_n |
| 0.44::maniac_a | 0.29::veteran_a | 0.25::Mason_n+and_c | 0.22::look_v+way_n |
| 0.38::dirty_a | 0.29::and_c+robot_n | 0.25::pron_rel_+kill_v | 0.22::dead_a |
| 0.37::honest_a | 0.28::and_c+criminal_n | 0.25::racist_a | 0.22::pron_rel_+stab_v |
| 0.36::uniformed_a | 0.28::bogus_a | 0.24::addicted_a | 0.21::pron_rel_+evade_v |
| 0.35::tough_a | 0.28::talk_v+to_p()+pron_rel_ | 0.23::gritty_a | |

Synonyms and similarity: some further examples:

- Similarity between *eggplant/aubergine*: 0.11
  These are true synonyms but have relatively low cosine similarity. This is partly due to frequency (222 for *eggplant*, 56 for *aubergine*).
- Similarity between *policeman/cop*: 0.23 (as discussed)
- Similarity between *city/town*: 0.73

In general, true synonymy does not correspond to higher similarity scores than near-synonymy.

Antonyms have high similarity, as indicated by the examples below:

- cold/hot 0.29
- dead/alive 0.24
- large/small 0.68
- colonel/general 0.33

It is possible to automatically distinguish antonyms from (near-)synonyms using corpus-based techniques, but this requires additional heuristics. For instance, it has been observed that antonyms are frequently coordinated while synonyms are not:

- a selection of cold and hot drinks
- wanted dead or alive
- lectures, readers and professors are invited to attend

Similarly, it is possible to acquire hyponymy relationships from distributions, but this is much less effective than looking for explicit taxonomic relationships in Wikipedia text.

## 8.6 Historical background

Distributional semantics has been discussed since at least the 1950s: the first computational work published was probably Karen Spärck Jones (1964) Cambridge PhD thesis 'Synonymy and Semantic Classification' which used dictionaries for context. The first experiments on sentential contexts I am aware of were by Harper (1965) (inspired by the work of the linguist Zellig Harris) which were refined to use a more motivated notion of similarity by Spärck Jones (1967). Salton's early work on vector space models in IR was also ongoing from the early 1960s. The early distributional work not followed up within computational linguistics, and in fact was almost entirely forgotten. This was partly because of the limitations of computers and available corpora, but also because the 1966 ALPAC report led to greatly diminished funding for CL and because the dominant Chomskyan approach in linguistics was highly hostile to any quantitative methodology. Spärck Jones switched to working on Information Retrieval, but the early classification experiments influenced her approach to IR, in particular the development of tf*idf. By the late 1980s and early 1990s there were sufficiently large corpora, computer memory and disk space to make simple distributional semantic techniques practical: much of the research at that point was heavily influenced by the IR techniques. By the early 2000s, large scale, robust parsing made more complex notions of distributional context practical and there has been a huge proliferation of CL research in recent years.