Natural Language Processing: part 2 of lecture notes

2003

Ann Copestake (aac@cl.cam.ac.uk)
http://www.cl.cam.ac.uk/users/aac/

5 Lecture 5: Parsing and generation II

The CFG approach which we've looked at so far has some serious deficiencies as a model of natural language. In this lecture, I'll discuss some of these and give an introduction to a more expressive formalism which is widely used in NLP, again with the help of a sample grammar. I will also sketch how we can use this approach to do computational compositional semantics.

(Note: it seems likely that this lecture has too much in it, so compositional semantics may actually be discussed in lecture 6.)

5.1 Deficiencies in atomic category CFGs

If we consider the grammar we saw in the last lecture, several problems are apparent. One is that there is no account of agreement, so, for instance, **it fish* is allowed by the grammar as well as *they fish*. There was also no account of *case*: this is only reflected in a few places in modern English, but **they can they* is clearly ungrammatical (as opposed to *they can them*, which is grammatical with the transitive verb use of *can*).

We could, of course, allow for agreement by increasing the number of atomic symbols in the CFG, introducing NP-sg, NP-pl, VP-sg and VP-pl, for instance. But this approach would soon become very tedious as we expanded the grammar, since it leads to doubling all the rules. If we also allow for case this way, the number of symbols increases still further:

```
S -> NP-sg-subj VP-sg
S -> NP-pl-subj VP-pl
VP-sg -> V-sg NP-sg-obj
VP-sg -> V-sg NP-pl-obj
VP-pl -> V-pl NP-sg-obj
VP-pl -> V-pl NP-pl-obj
```

Note that we have to expand out the symbols even when there's no constraint on agreement, since we have no way of saying that we don't care about the value of number for a category.

Another linguistic phenomenon that we are failing to deal with is *subcategorization*. This is the lexical property that tells us how many *argument*s a verb can have (among other things). A verb such as *adore*, for instance, is transitive: a sentence such as **Kim adored* is strange, while *Kim adored Sandy* is usual. A verb such as *give* is *ditransitive*: *Kim gave Sandy an apple* (or *Kim gave an apple to Sandy*). Without going into details of exactly how subcategorization is defined, or what an argument is, it should be intuitively obvious that we're not encoding this propery with our CFG. The grammmar allows the following, for instance:

> *they fish fish it*
> (S (NP they) (VP (V fish) (VP (V fish) (NP it))))

Again this could be dealt wth by multiplying out symbols (V-intrans, V-ditrans etc), but the grammar becomes extremely cumbersome.

Finally, consider the phenomenon of *long-distance dependencies*, exemplified, for instance, by:

> which problem did you say you don't understand?
> who do you think Kim asked Sandy to hit?
> which kids did you say were making all that noise?

Traditionally, these sentences are said to contain 'gap's, corresponding to the place where the noun phrase would normally appear: the gaps are marked by underscores below:

> which problem did you say you don't understand _?
> who do you think Kim asked Sandy to hit?
> which kids did you say _ were making all that noise?

Notice that, in the third example, the verb *were* shows plural agreement.

Doing this in standard CFGs is possible, but extremely verbose, potentially leading to millions of rules. Instead of having simple atomic categories in the CFG, we want to allow for features on the categories, which can have values indicating things like plurality. As the long-distance dependency examples should indicate, the features need to be complex-valued. For instance,

> * what kid did you say _ were making all that noise?

is not grammatical. The analysis needs to be able to represent the information that the gap corresponds to a plural noun phrase.

In what follows, I will illustrate a simple *constraint-based grammar* formalism, using *feature structures*. A constraint-based grammar describes a language using a set of independently stated constraints, without imposing any conditions on processing or processing order. A CFG can be taken as an example of a constraint-based grammar, but usually the term is reserved for richer formalisms. The simplest way to think of feature structures (FSs) is that we're replacing the atomic categories of a CFG with more complex data structures. I'll first illustrate this idea intuitively, using a grammar fragment like the one in lecture 4 but enforcing agreement and subcategorization (I'll ignore case and long-distance dependencies). I'll then go through the feature structure formalism in more detail.

5.2 A simple FS grammar

The following section should be read in conjunction with the grammar overleaf. In a FS grammar, rules are described as relating FSs: i.e., lexical entries and phrases are FSs. In these formalisms, the term *sign* is often used to refer to lexical entries and phrases collectively. In fact, rules themselves can be treated as FSs, although I'll ignore this for the moment, and continue to use the arrow notation. Feature structures are singly-rooted directed acyclic graphs, with arcs labelled by features and terminal nodes associated with values. So a particular feature in a structure may be *atomic-valued*, meaning it points to a terminal node in the graph, or *complex-valued*, meaning it points to a non-terminal node. A sequence of features is known as a *path*. Since these are graphs, rather than trees, a particular node may be accessed from the root by more than one path: this is known as *reentrancy*.

FSs are usually drawn as *attribute-value matrices* or AVMs. In AVMs, reentrancy is conventionally indicated by boxed integers, with node identity indicated by integer identity. The actual integers used are arbitrary.

When using FSs in grammars, structures are combined by *unification*. This means that all the information in the two structures is combined. The empty square brackets ($\left[\ \right]$) in an AVM indicate that a value is unspecified: i.e. this is a node which can be unified with a terminal node (i.e., an atomic value) or a complex value. More details are given below.

When FSs are used in a particular grammar, all signs will have a similar set of features (although sometimes there are differences between lexical and phrasal signs). Feature structure grammars can be used to implement a variety of linguistic frameworks. In the particular very simple grammar below, signs have three features at the toplevel: HEAD, COMP and SPR. This reflects a portion of a linguistic framework which is described in great detail in Sag and Wasow (1999).[1] Briefly, HEAD contains information which is shared between the lexical entries and phrases of the same category: e.g., nouns share this information with the noun phrase which dominates them in the tree, while verbs share head information with verb phrases and sentences. Here, HEAD is used for agreement information and for category information (i.e. noun, verb etc) indicated by the feature CAT. In contrast, COMP and SPR are about subcategorization: they contain information about what can combine with this sign. For instance, an intransitive verb will have a SPR corresponding to its subject 'slot' and a value of **filled** for its COMP.[2]

The grammar below has two rules, one for combining a sign with its complement, another for combining a sign with its specifier. Rule 1 says that, when building the phrase, the COMP value of the first daughter is to be equated (unified) with the whole structure of the second daughter (indicated by $\boxed{2}$). The head of the mother is equated with the head of the first daughter ($\boxed{1}$). The spr of the mother is also equated with the spr of the first daughter ($\boxed{3}$). The comp value of the mother is stipulated as being **filled**: this means the mother can't act as the first daughter in another application of the rule, since **filled** won't unify with a complex feature structure. The specifier rule is fairly similar, in that a spr

[1] You aren't expected to know any details of the linguistic framework for the exam.

[2] There's a more elegant way of doing this using lists, but since this complicates the grammar quite a lot, I won't show this here.

'slot' is being instantiated, although in this case it's the second daughter that contains the slot and is sharing its head information with the mother.[3] The rule also stipulates that the AGR values of the two daughters have to be unified and that the specifier has to have a filled complement. These rules are controlled by the lexical entries in the sense that it's the lexical entries which determine the required complements and specifier of a word.

Note that the grammar also has a root FS: a structure only counts as a valid parse if it is unifiable with the root.

As an example, consider analysing *they fish*. The verb entry for *fish* can be unified with the second daughter position of rule 2, giving the following partially instantiated rule:

$$\begin{bmatrix} \text{HEAD } \boxed{1} \begin{bmatrix} \text{CAT } \textbf{verb} \\ \text{AGR } \boxed{3}\ \textbf{pl} \end{bmatrix} \\ \text{COMP } \textbf{filled} \\ \text{SPR } \textbf{filled} \end{bmatrix} \rightarrow \boxed{2} \begin{bmatrix} \text{HEAD } \begin{bmatrix} \text{CAT } \textbf{noun} \\ \text{AGR } \boxed{3} \end{bmatrix} \\ \text{COMP } \textbf{filled} \\ \text{SPR } \textbf{filled} \end{bmatrix}, \begin{bmatrix} \text{HEAD } \boxed{1} \\ \text{COMP } \textbf{filled} \\ \text{SPR } \boxed{2} \end{bmatrix}$$

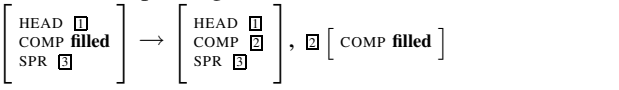
The first daughter of this result can be unified with the structure for *they*, which in this case returns the same structure, since it adds no new information. The result can be unified with the root structure, so this is a valid parse.

On the other hand, the lexical entry for the noun *fish* does not unify with the second daughter position of rule2. The entry for *they* does not unify with the first daughter position of rule1. Hence there is no other parse.

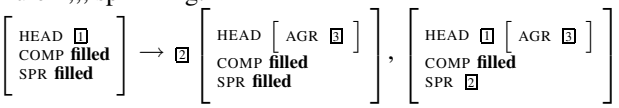
[3]Note that the reentrancy indicators are local to each rule: the $\boxed{1}$ in rule 1 is not the same structure as the $\boxed{1}$ in rule 2.

Simple FS grammar fragment

Rule1 ;;; comp filling



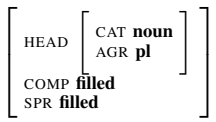
Rule2 ;;; spr filling:



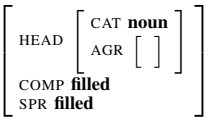
Lexicon:

;;; noun phrases

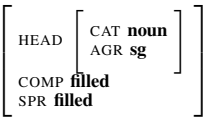
they



fish

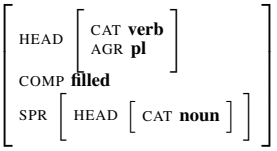


it

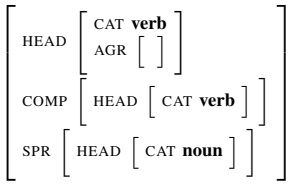


;;; verbs

fish

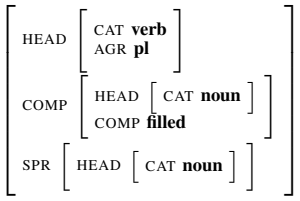


can



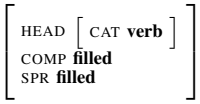
;;; auxiliary verb

can



;;; transitive verb

Root structure:



5.3 Feature structures in detail

FSs can be thought of as graphs which have labelled arcs connecting nodes (except for the case of the simplest FSs, which consist of a single node with no arcs) The labels on the arcs are the features. Arcs are regarded as having a direction, conventionally regarded as pointing into the structure, away from the single root node. The set of features and the set of atomic values are assumed to be finite.

Properties of FSs

Connectedness and unique root A FS must have a unique root node: apart from the root node, all nodes have one or more parent nodes.

Unique features Any node may have zero or more arcs leading out of it, but the label on each (that is, the feature) must be unique.

No cycles No node may have an arc that points back to the root node or to a node that intervenes between it and the root node. (Some variants of FS formalisms allow cycles.)

Values A node which does not have any arcs leading out of it may have an associated atomic value.

Finiteness A TFS must have a finite number of nodes.

Sequences of features are known as *path*s.

Feature structures can be regarded as being ordered by information content — an FS is said to *subsume* another if the latter carries extra information.

Properties of subsumption FS1 subsumes FS2 if and only if the following conditions hold:

Path values For every path P in FS1 there is a path P in FS2. If P has a value t in FS1, then P also has value t in FS2.

Path equivalences Every pair of paths P and Q which are reentrant in FS1 (i.e., which lead to the same node in the graph) are also reentrant in FS2.

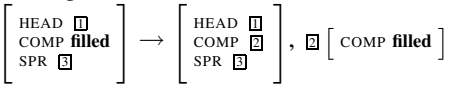
Unification corresponds to conjunction of information, and thus can be defined in terms of subsumption, which is a relation of information containment. The unification of two FSs is defined to be the most general FS which contains all the information in both of the FSs. Unification will fail if the two FSs contain conflicting information. This will prevent *it fish* getting an analysis with the grammar above, for instance, because the AGR values will conflict.

Properties of unification The unification of two TFSs FS1 and FS2 is the most general TFS which is subsumed by both FS1 and FS2, if it exists.

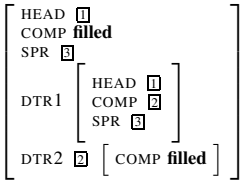
5.4 Grammar rules as feature structures

Grammar rules are actually implemented as FSs. The example below should make this clear.

In the grammar above, rule 1 was written as:



It is actually a single FS, as follows:



Note that this means that a rule corresponds to an entire phrase, including the daughters.

5.5 Feature structure grammars viewed as constraints

We can formalize feature structure grammars as constraints on the feature structures that constitute valid phrases:
Properties of a feature structure grammar

Grammar A grammar consists of a set of grammar rules, G, a set of lexical entries, L, and a start structure, Q, which are all FSs. Each grammar rule in G has one or more daughter paths, $D_1 \ldots D_n$ (here DTR1 and DTR2).

Lexical sign A lexical sign is a pair $\langle L, S \rangle$ of a FS L and a string list S, such that L is a lexical entry that matches the string S.

Valid phrase A valid phrase P is a pair $\langle F, S \rangle$ of a FS F and a string list S such that either:

1. P is a lexical sign, or
2. F is subsumed by some rule R and there are valid phrases $\langle F_1, S_1 \rangle \ldots \langle F_n, S_n \rangle$ such that the values of each of R's daughter paths $D_1 \ldots D_n$ subsume $F_1 \ldots F_n$ respectively and S is the ordered concatenation of $S_1 \ldots S_n$.

Sentences A string list S is a well-formed sentence according to the grammar if there is a valid phrase $\langle F, S \rangle$ such that the start structure Q subsumes F.

This formalisation simply tells us what the valid structures are — it is neutral between parsing and generation. For parsing, we start with a string and try and construct a valid phrase corresponding to the string. For generation, we start with a partial structure representing some semantics, and try and construct a corresponding string.

5.6 Parsing with feature structure grammars

Although formally we can treat feature structure grammars in terms of subsumption, implementations have to be more efficient. The standard approach is to use chart parsing, as described in the previous lecture, but the notion of a grammar rule matching an edge in the chart is more complex. In a naive implementation, when application of a grammar rule is checked, all the feature structures in the edges in the chart that correspond to the possible daughters have to be copied, and the grammar rule feature structure itself is also copied. The copied daughter structures are unified with the daughter positions in the copy of the rule, and if unification succeeds, the copied structure is associated with a new edge on the chart.

The need for copying is often discussed in terms of the destructive nature of the standard algorithm for unification (which I won't describe here), but this is perhaps a little misleading. Unification, however implemented, involves sharing information between structures. Assume, for instance, that the FS representing the lexical entry of the noun for *fish* is underspecified for number agreement. When we parse a sentence like:

> the fish swims

the part of the FS in the result that corresponds to the original lexical entry will have its AGR value instantiated. Perhaps more surprisingly, the FS for *the* will also have its AGR value instantiated (think about the definition of parsing in terms of subsumption to see this). Although I won't go into the details here, linguistic frameworks that use feature structures often rely on this property. This means that the structure corresponding to a particular edge cannot be reused in another analysis, because it will contain 'extra' information. Consider, for instance, parsing:

> the fish in the lake which is near the town swim

A possible analysis of:

> fish in the lake which is near the town

is:

> (fish (in the lake) (which is near the town))

i.e., the fish (sg) is near the town. If we instantiate the AGR value in the FS for *fish* as sg while constructing this parse, and then try to reuse that FS when we get to the sentence as a whole, analysis will fail. Hence the need for copying, so we can use a fresh structure each time. Copying is potentially extremely expensive, because realistic grammars involve FSs with many hundreds of nodes.

So, although unification is very near to linear in complexity, naive implementations of FS formalisms are very inefficient. Furthermore, packing is not straightforward, because two structures are rarely identical in real grammars (especially ones that encode semantics).

Reasonably efficient implementations of FS formalisms can nevertheless be developed. Copying can be greatly reduced:

1. by doing an efficient pretest before unification, so that copies are only made when unification is likely to succeed
2. by sharing parts of FSs that aren't changed
3. by taking advantage of *locality principles* in linguistic formalisms which limit the need to percolate information through structures

Packing can also be implemented: the test to see if a new edge can be packed involves subsumption rather than equality.

As with CFGs, for real efficiency we need to control the search space so we only get the most likely analyses. Defining probabilistic FS grammars in a way which is theoretically well-motivated is much more difficult than defining a PCFG. Practically it seems to turn out that treating a FS grammar much as though it were a CFG works fairly well, but this is an active research issue.

5.7 Templates

The lexicon outlined above has the potential to be very redundant. For instance, as well as the intransitive verb *fish*, a full lexicon would have entries for *sleep*, *snore* and so on, which would be essentially identical. We avoid this redundancy by associating names with particular feature structures and using those names in lexical entries. For instance:
fish INTRANS_VERB
sleep INTRANS_VERB
snore INTRANS_VERB
where the template is specified as:

INTRANS_VERB

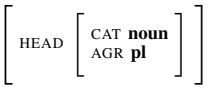
$$\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{verb} \\ \text{AGR} & \textbf{pl} \end{bmatrix} \\ \text{COMP} & \textbf{filled} \\ \text{SPR} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{noun} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The lexical entry may have some specific information associated with it (e.g., semantic information) which will be expressed as a FS: in this case, the template and the lexical feature structure are combined by unification.

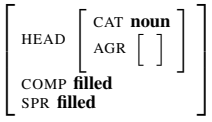
More recent FS formalisms assume the use of *type*s. Each node on a feature structure has an associated type: terminal nodes in the structure have atomic types rather than values, but non-terminal nodes are also typed. Types are arranged in their own subsumption hierarchy, hence allowing another dimension of underspecification. For instance, we could have a type **num** which had subcases **sg** and **pl**. Types also constrain the values of feature structures and the type hierarchy acts as an inheritance hierarchy, replacing the need for templates.

5.8 Interface to morphology

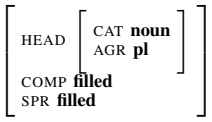
So far we have assumed a full-form lexicon, but we can now return to the approach to morphology that we saw in lecture 2, and show how this relates to feature structures. Recall that we have spelling rules which can be used to analyse a word form to return a stem and list of affixes and that each affix is associated with an encoding of the information it contributes. For instance, the affix *s* is associated with the template `PLURAL_NOUN`, which would correspond to the following information in our grammar fragment:



A stem for a noun is generally assumed to be uninstantiated for number (i.e., neutral between sg and pl). So the lexical entry for the noun *dog* in our fragment would be:



One simple way of implementing inflectional morphology in FSs is simply to unify the contribution of the affix with that of the stem. If we unify the FS corresponding to the stem for *dog* to the FS for `PLURAL_NOUN`, we get:



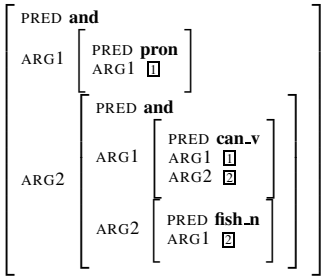
This approach assumes that we also have a template `SINGULAR_NOUN`, where this is associated with a 'null' affix.

In the case of an example such as *feed* incorrectly analysed as *fee -ed*, discussed in §2.5, the affix information will fail to unify with the stem, ruling out that analysis.

There are other ways of encoding inflectional morphology with FS, which I won't discuss here. Note that this simple approach is not, in general, adequate for derivational morphology. For instance, the affix *-ize*, which combines with a noun to form a verb (e.g., *lemmatization*), cannot be represented simply by unification, because it has to change a nominal form into a verbal one. This can be implemented by some form of *lexical rule* (which are essentially grammar rules with single daughters), but I won't discuss this in this course. Note, however, that this reflects the distinction between inflectional and derivational morphology that we saw in §2.2:while inflectional morphology can be seen as simple addition of information, derivational morphology converts feature structures into new structures. However, derivational morphology is often not treated as productive, especially in limited domain systems.

5.9 Simple semantics in feature structures

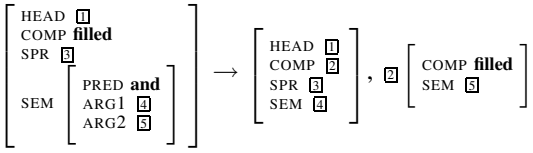
The grammar fragment below is intended as a rough indication of how it is possible to build up semantic representations using feature structures. The lexical entries have been augmented with pieces of feature structure reflecting predicate-argument structure. With this grammar, the FS for *they can fish* will have a SEM value of:



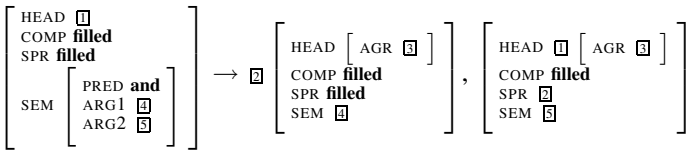
This can be taken to be equivalent to the logical expression $\text{pron}(x) \wedge (\text{can}(x, y) \wedge \text{fish}(y))$ by translating the reentrancy between argument positions into variable equivalence.

Simple FS grammar with crude semantic composition

Rule1 ;;; comp filling

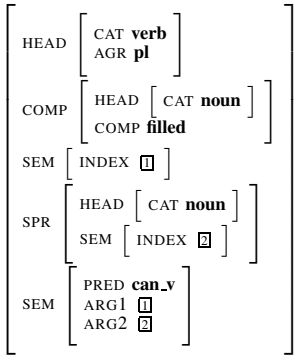


Rule2 ;;; spr filling:



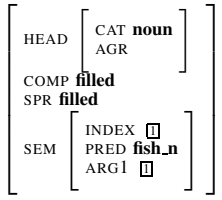
Lexicon:

can



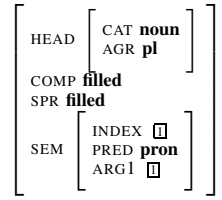
;;; transitive verb

fish



;;; noun phrase

they



;;; noun phrase

An alternative approach to encoding semantics is to write the semantic composition rules in a separate formalism such as *typed lambda calculus*. This corresponds more closely to the approach most commonly assumed in formal linguistics: variants of lambda calculus are sometimes used in NLP, but I won't discuss this further here.

In general, a semantic representation constructed for a sentence is called the *logical form* of the sentence. The semantics shown above can be taken to be equivalent to a form of predicate calculus without variables or quantifiers: i.e. the 'variables' in the representation actually correspond to constants. It turns out that this very impoverished form of semantic representation is adequate for many NLP applications: template representations, used in information extraction or simple dialogue systems can be thought of as equivalent to this. But for a fully adequate representation we need something richer — for instance, to do negation properly. Minimally we need full first-order predicate calculus (FOPC). FOPC logical forms can be passed to theorem-provers in order to do inference about the meaning of a sentence. However, although this approach has been extensively explored in research work, especially in the 1980s, it hasn't so far led to practical systems. There are many reasons for this, but perhaps the most important is the difficulty of acquiring detailed domain knowledge expressed in FOPC. There is also a theoretical AI problem, because we seem to need some form of probabilistic reasoning for many applications. So, although most researchers who are working in computational compositional semantics take support for inference as a desideratum, many systems actually use some

form of shallow inference (e.g., semantic transfer in MT, mentioned in lecture 8).

FOPC also has the disadvantage that it forces quantifiers to be in a particular scopal relationship, and this information is not (generally) overt in NL sentences. One classic example is:

Every man loves a woman

which is ambiguous between:

$$\forall x[\text{man}'(x) \Rightarrow \exists y[\text{woman}'(y) \wedge \text{love}'(x, y)]]$$

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

$$\exists y[\text{woman}'(y) \wedge \forall x[\text{man}'(x) \Rightarrow \text{love}'(x, y)]]$$

Most current 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.

5.10 Generation

We can generate from a semantic representation with a suitable FS grammar. Producing an output string given an input logical form is generally referred to as *tactical generation* or *realization*, as opposed to *strategic generation* or *text planning*, which concerns how you might build the logical form in the first place. Strategic generation is an open-ended problem: it depends very much on the application and I won't have much to say about it here. Tactical generation is more tractable, and is useful without a strategic component in some contexts, such as the semantic transfer approach to MT, which I'll briefly discuss in lecture 8.

Tactical generation can use similar techniques to parsing: for instance one approach is *chart generation* which uses many of the same techniques as chart parsing. There has been much less work on generation than on parsing in general, and building bidirectional grammars is hard: most grammars for parsing allow through many ungrammatical strings. Recently there has been some work on statistical generation, where n-grams are used to choose between realizations constructed by a grammar that overgenerates. But even relatively 'tight' bidirectional grammars may need to use statistical techniques in order to generate natural sounding utterances.

5.11 Further reading

J&M describe feature structures as augmenting a CFG rather than replacing it, but most of their discussion applies equally to the FS formalism I've outlined here. They go into quite a lot of detail about compositional semantics including underspecification.

LinGO (Linguistic Grammars Online: `http://lingo.stanford.edu`) distributes Open Source FS grammars for a variety of languages. The LinGO English Resource Grammar (ERG) is probably the largest freely available bidirectional grammar.

6 Lecture 6: Lexical semantics

This lecture will give a rather superficial account of lexical semantics and some of its computational aspects:

1. Meaning postulates
2. Classical lexical relations: hyponymy, meronomy, synonymy, antonymy
3. Taxonomies and WordNet
4. Classes of polysemy: homonymy, regular polysemy, vagueness
5. Word sense disambiguation

6.1 Meaning postulates

Inference rules can be used to relate open class predicates: i.e., predicates that correspond to open class words. This is the classic way of representing lexical meaning in formal semantics within linguistics:[4]

$$\forall x[bachelor(x) \leftrightarrow man(x) \wedge unmarried(x)]$$

Linguistically and philosophically, this gets pretty dubious. 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.[5]

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 NLP, we're normally concerned with implication rather than definition and this is less problematic philosophically:

$$\forall x[bachelor(x) \rightarrow man(x) \wedge unmarried(x)]$$

However, the big computational problems with meaning postulates are their acquisition and the control of inference once they have been obtained. Building meaning postulates for anything other than a small, bounded domain is an AI-complete problem.

The more general, shallower, relationships that are classically discussed in lexical semantics are currently more useful in NLP, especially for broad-coverage processing.

6.2 Hyponymy: IS-A

Hyponymy is the classical IS-A relation: e.g. *dog* is a *hyponym* of *animal*. That is, the relevant sense of *dog* is the hyponym of *animal*: 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.

animal is the *hypernym* of *dog*. Hyponyms can be arranged into *taxonomies*: classically these are tree-structured: i.e., each term has only one *hypernym*.

Despite the fact that hyponymy is by far the most important meaning relationship assumed in NLP, many questions arise which don't currently have very good answers:

[4] 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!

[5] 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.

1. What classes of words can be categorized 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 essentially useless for adjectives.
2. Do differences in quantization 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.

6.3 Meronomy: 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!

There are several different types of part-of relationships. e.g., entity in a collection (e.g., a soldier is part of an army).

6.4 Synonymy

True synonyms are relatively uncommon: most cases of true synonymy are correlated with dialect differences (e.g., *eggplant* / *aubergine*, *boot* / *trunk*) where the differences can be national/regional or temporal (or gender-based or class-based — in British English *napkin* vs *serviette*, *sofa* vs *settee* etc are often cited as examples where class makes a difference to the term used).

Most synonymy involves register distinctions, slang or jargons: e.g., *policeman*, *cop*, *rozzer* ...

Near-synonyms generally convey nuances of meaning: *thin*, *slim*, *slender*, *skinny*. Note that in some of these cases, some words have wider applicability than others, and that collocations differ — e.g., *slender means* but not **skinny means*, *thin wallet*, *slender wallet* (perhaps), but probably not *skinny wallet*.

6.5 Antonymy

Antonymy is mostly discussed with respect to adjectives: e.g., *big*/*little*, though it's only relevant for some classes of adjectives. It is also possibly relevant for some nouns and verbs: e.g., *like*/*dislike*, *hate*/*love*.

6.6 Other relationships

The classical relationships do not exhaust the possible systematic relationships between words. e.g. consider *buy*/*sell*: it s maybe a little like *antonymy*, but actually what's going on is argument swapping:

> X bought Y from Z $\leftrightarrow$ Z sold Y to X
> buy(x,y,z) $\leftrightarrow$ sell(z,y,x)

Notice:

> X bought Y from Z willingly does not imply Z sold Y to X willingly

The *buy*/*sell* style of relationship really requires some form of logical representation in order to encode it. The classical lexical semantic relationships don't look at relationships between the arguments of predicates with multiple arguments.

6.7 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 organized 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
```

The following is an overview of the information available in WordNet for the various POS classes:

- all classes
    1. synonyms (ordered by frequency)
    2. familiarity / polysemy count
    3. compound words (done by spelling)
- nouns
    1. hyponyms / hypernyms (also sisters)

2. holonyms / meronyms

- adjectives
  1. antonyms
- verbs
  1. antonyms
  2. hyponyms / hypernyms (also sisters)
  3. syntax (very simple)
- adverbs

Taxonomies have also been automatically or semi-automatically extracted from machine-readable dictionaries, but these are not distributed. Microsoft's MindNet is the best known example (it has many more relationships than just hyponymy). There are other collections of terms, generally hierarchically ordered, especially medical ontologies. There have been a number of attempts to build an ontology for world knowledge: none of the more elaborate ones are generally available. There is an ongoing attempt at standardization of ontologies. Ontology support is an important component of the semantic web.

6.8 Using lexical semantics

By far the most commonly used lexical relation is hyponymy. Hyponymy relations can be used in many ways:

- 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
- Back-off to semantic classes in some statistical approaches
- Word-sense disambiguation
- MT: if you can't translate a term, substitute a hypernym
- Query expansion: 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 cognitive synonyms for customizing text hasn't really been looked at.)

6.9 Polysemy

The standard example of polysemy is *bank* (river bank) vs *bank* (financial institution).

This is *homonymy* — the two senses are unrelated (not entirely true for *bank*, actually, but historical relatedness isn't actually important — it's whether ordinary speakers of the language feel there's a relationship). Homonymy is the most obvious case of polysemy, but is actually 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 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 descisions about whether and how to split up senses.

6.10 Word sense disambiguation

Word sense disambiguation (WSD) is needed for most NL applications that involve semantics (explicitly or implicitly). In limited domains, WSD is not too big a problem, but for large coverage text processing it's a serious bottleneck.

WSD needs depend on the application — there is no objective notion of word sense (dictionaries differ extensively) and it's very hard to come up with good criteria to judge whether or not to distinguish senses. But in order to experiment with WSD as a standalone module, there has to be a standard: most commonly WordNet, because it is the only extensive modern resource for English with no problematic IPR issues. This is controversial, because WordNet has a very fine granularity of senses — it's also obvious that its senses often overlap. However, the only current alternative is a pre-1920 version of Webster's. Recently WSD 'competitions' have een organized: SENSEVAL and SENSEVAL 2.

WSD up to the early 1990s was mostly done by hand-constructed rules (still used in some MT systems). Dahlgren investigated WSD in a fairly broad domain in the 1980s. Reasonably broad-coverage WSD generally depends on:

- frequency
- collocations
- selectional restrictions/preferences

What's changed since the 1980s is that various statistical or machine-learning techniques have been used to avoid hand-crafting rules.

- supervised learning. 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). Most experimentation has been done with a small set of words which can be sense-tagged by the experimenter (e.g., *plant*). Supervised learning techniques do not carry over well from one corpus to another.
- unsupervised learning (see below)
- Machine readable dictionaries (MRDs). Disambiguating dictionary definitions according to the internal data in dictionaries is necessary to build taxonomies from MRDs. MRDs have also been used as a source of selectional preference and collocation information for general WSD (quite successfully).

Until recently, most of the statistical or machine-learning techniques have been evaluated on homonyms: these are relatively easy to disambiguate. So 95% disambiguation in e.g., Yarowsky's experiments sounds good (see below), but doesn't translate into high precision on all words when target is WordNet senses (in SENSEVAL 2 the best system was around 70%).

There have also been some attempts at automatic *sense induction*, where an attempt is made to determine 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 be argued to be artefacts of dictionary publishing. However, so far sense induction has not been much explored in monolingual contexts, though it could be considered as an inherent part of statistical approaches to MT.

6.11 Collocations

Collocations have always been the most useful source of information for WSD, even in Dahlgren's early experiments. For instance:

(1) Striped bass are common.

(2) Bass guitars are common.

striped is a good indication that we're talking about the fish (because it's a particular sort of bass), similarly with *guitar* and music. In both *bass guitar* and *striped bass*, we've arguably got a multi-word expression (i.e., a conventional phrase that might be listed in a dictionary), but the principle holds for any sort of collocation. The best collocates for WSD tend to be syntactically related in the sentence to the word to be disambiguated, but many techniques simply use a window of words.

J&M make a useful (though non-standard) distinction between collocation and cooccurrence: cooccurrence refers to the appearence of another word in a larger window of text than a collocation. For instance, *trout* might cooccur with the fish sense of bass.

6.12 Yarowsky's unsupervised learning approach to WSD

Yarowsky (1995) describes a technique for unsupervised learning using collocates (collocates and cooccurrences in J&M's terms). A few seed collocates are chosen for each sense (manually or via an MRD), then these are used to accurately identify distinct senses. The sentences in which the disambiguated senses occur can then be used to learn other discriminating collocates automatically, producing a decision list. The process can then be iterated. The algorithm allows bad collocates to be overridden. This works because of the general principle of 'one sense per collocation' (experimentally demonstrated by Yarowsky — it's not absolute, but there are very strong preferences).

In a bit more detail, using Yarowsky's example of disambiguating *plant* (which is homonymous between factory vs vegetation senses):

1. Identify all examples of the word to be disambiguated in the training corpus and store their contexts.

sense	training example
?	company said that the *plant* is still operating
?	although thousends of *plant* and animal species
?	zonal distribution of *plant* life
?	company manufacturing *plant* is in Orlando
	etc

2. Identify some seeds which reliably disambiguate a few of these uses. Tag the disambiguated senses and count the rest as residual. For instance, choosing 'plant life' as a seed for the vegetation sense of plant (sense A) and 'manufacturing plant' as the seed for the factory sense (sense B):

sense	training example
?	company said that the *plant* is still operating
?	although thousands of *plant* and animal species
A	zonal distribution of *plant* life
B	company manufacturing *plant* is in Orlando
	etc

This disambiguated 2% of uses in Yarowsky's corpus, leaving 98% residual.

3. Train a *decision list* classifier on the Sense A/Sense B examples. A decision list approach gives a list of criteria which are tried in order until an applicable test is found: this is then applied. The tests are each associated with a reliability metric. The original seeds are likely to be at the top of the initial decision list, followed by other discriminating terms. e.g. the decision list might include:

reliability	criterion	sense
8.10	*plant* life	A
7.58	manufacturing *plant*	B
6.27	*animal* within 10 words of *plant*	B
	etc	

4. Apply the decision list classifier to the training set and add all examples which are tagged with greater than a threshhold reliability to the Sense A and Sense B sets.

sense	training example
?	company said that the *plant* is still operating
A	although thousands of *plant* and animal species
A	zonal distribution of *plant* life
B	company manufacturing *plant* is in Orlando
	etc

5. Iterate the previous steps 3 and 4 until convergence

6. Apply the classifier to the unseen test data

Yarowsky also demonstrated the principle of 'one sense per discourse' (again, a very strong, but not absolute effect). This can be used as an additional refinement for the algorithm above.

Yarowsky argues that decision lists work better than many other statistical frameworks because no attempt is made to combine probabilities. This would be complex, because the criteria are not independent of each other.

Yarowsky's experiments were nearly all on homonyms: these principles probably don't hold as well for sense extension.

6.13 Evaluation of WSD

The baseline for WSD is generally 'pick the most frequent' sense: this is hard to beat! However, in many applications, we don't know the frequency of senses.

SENSEVAL and SENSEVAL-2 evaluated WSD in multiple languages, with various criteria, but generally using WordNet senses for English. The human ceiling for this task varies considerably between words: probably partly because of inherent differences in semantic distance between groups of uses and partly because of WordNet itself, which sometimes makes very fine-grained distinctions. An interesting variant in SENSEVAL-2 was to do one experiment on WSD where the disambiguation was with respect to uses requiring different translations into Japanese. This has the advantage that it is useful and relatively objective, but sometimes this task requires splitting terms which aren't polysemous in English (e.g., *water* — hot vs cold). Performance of WSD on this task seems a bit better than the general WSD task.

6.14 Further reading

WordNet is freely downloadable: the website has pointers to several papers which provide a good introduction.

For a lot more detail of WSD than provided by J&M, see Manning and Schütze who have a very detailed account of WSD and word-sense induction:

Manning, Christopher and Hinrich Schütze (1999), *Foundations of Statistical Natural Language Processing*, MIT Press

Yarowsky's paper is well-written and should be understandable:

Yarowsky, David (1995)
Unsupervised word sense disambiguation rivaling supervised methods,
Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95) MIT, 189–196

Like many other recent NLP papers, this can be downloaded via www.citeseer.com

7 Lecture 7: Discourse

Utterances are always understood in a particular context. Context-dependent situations include:

1. Referring expressions: pronouns, definite expressions etc.
2. Universe of discourse: *every dog barked*, doesn't mean every dog in the world but only every dog in some explicit or implicit contextual set.
3. Responses to questions, etc: only make sense in a context: *Who came to the party? Not Sandy.*
4. Implicit relationships between events: *Max fell. John pushed him* — the second sentence is (usually) understood as providing a causal explanation.

In the first part of this lecture, I give a brief overview of *rhetorical relations* which can be seen as structuring text at a level above the sentence. I'll then go on to talk about one particular case of context-dependent interpretation — anaphor resolution. I will describe an algorithm for anaphor resolution which uses a relatively broad-coverage shallow parser and then discuss a variant of it that relies on POS-tagging and regular expression matching rather than parsing.

7.1 Rhetorical relations and coherence

Consider the following discourse:

> Max fell. John pushed him.

This discourse can be interpreted in at least two ways:

1. Max fell because John pushed him.
2. Max fell and then John pushed him.

There seems to be an implicit relationship between the two original sentences: a *discourse relation* or *rhetorical relation*. (I will use the terms interchangeably here, though different theories use different terminology, and rhetorical relation tends to refer to a more surfacy concept than discourse relation.) In 1 the link is a form of explanation, but 2 is an example of narration. Theories of discourse/rhetorical relations reify link types such as *Explanation* and *Narration*. The relationship is made more explicit in 1 and 2 than it was in the original sentence: *because* and *and then* are said to be *cue phrases*.

7.2 Coherence

Discourses have to have connectivity to be coherent:

> Kim got into her car. Sandy likes apples.

Both of these sentences make perfect sense in isolation, but taken together they are incoherent. Adding context can restore coherence:

> Kim got into her car. Sandy likes apples, so Kim thought she'd go to the farm shop and see if she could get some.

The second sentence can be interpreted as an explanation of the first. In many cases, this will also work if the context is known, even if it isn't expressed.

Strategic generation requires a way of implementing coherence. For example, consider a system that reports share prices. This might generate:

> In trading yesterday: Dell was up 4.2%, Safeway was down 3.2%, Compaq was up 3.1%.

This is much less acceptable than a connected discourse:

> Computer manufacturers gained in trading yesterday: Dell was up 4.2% and Compaq was up 3.1%. But retail stocks suffered: Safeway was down 3.2%.

Here *but* indicates a Contrast. Not much actual information has been added (assuming we know what sort of company Dell, Compaq and Safeway are), but the discourse is easier to follow.

Discourse coherence assumptions can affect interpretation:

> John likes Bill. He gave him an expensive Christmas present.

If we interpret this as Explanation, then 'he' is most likely Bill. But if it is Justification (i.e., the speaker is justifying the first sentence), then 'he' is John.

7.3 Factors influencing discourse interpretation

1. Cue phrases. These are sometimes unambiguous, but not usually. e.g. *and* is a cue phrase when used in sentential or VP conjunction.
2. Punctuation (also prosody) and text structure. For instance, parenthetical information cannot be related to a main clause by Narration, but a list is often interpreted as Narration:

   > Max fell (John pushed him) and Kim laughed.
   > Max fell, John pushed him and Kim laughed.

   Similarly, enumerated lists can indicate a form of narration.
3. Real world content:

   > Max fell. John pushed him as he lay on the ground.

4. Tense and aspect.

   > Max fell. John had pushed him.
   > Max was falling. John pushed him.

It should be clear that it is potentially very hard to identify rhetorical relations. In fact, recent research that simply uses cue phrases and punctuation is proving quite promising. This can be done by hand-coding a series of finite-state patterns, or by a form of supervised learning.

7.4 Discourse structure and summarization

If we consider a discourse relation as a relationship between two phrases, we get a binary branching tree structure for the discourse. In many relationships, such as Explanation, one phrase depends on the other: e.g., the phrase being explained is the main one and the other is subsidiary. In fact we can get rid of the subsidiary phrases and still have a reasonably coherent discourse. (The main phrase is sometimes called the *nucleus* and the subsidiary one is the *satellite*.) This can be exploited in summarization.

For instance:

> We get a binary branching tree structure for the discourse. In many relationships one phrase depends on the other. In fact we can get rid of the subsidiary phrases and still have a reasonably coherent discourse.

Other relationships, such as Narration, give equal weight to both elements, so don't give any clues for summarization.

Rather than trying to find rhetorical relations for arbitrary text, genre-specific cues can be exploited, for instance for scientific texts. This allows more detailed summaries to be constructed.

7.5 Referring expressions

I'll now move on to talking about another form of discourse structure, specifically the link between referring expressions. The following example will be used to illustrate referring expressions and anaphora resolution:

> Niall Ferguson is prolific, well-paid and a snappy dresser. Stephen Moss hated him — at least until he spent an hour being charmed in the historian's Oxford study. (quote taken from the Guardian)

Some terminology:

referent a real world entity that some piece of text (or speech) refers to. e.g., the two people who are mentioned in this quote.

referring expressions bits of language used to perform reference by a speaker. In, the paragraph above, *Niall Ferguson*, *him* and *the historian* are all being used to refer to the same person (they *corefer*).

antecedant the text evoking a referent. *Niall Ferguson* is the antecedant of *him* and *the historian*

anaphora the phenomenon of referring to an antecedant: *him* and *the historian* are *anaphoric* because they refer to a previously introduced entity.

What about *a snappy dresser*? Traditionally, this would be described as predicative: that is, it is a predicate, like an adjective, rather than being a referring expression itself.

Generally, entities are introduced in a discourse (technically, *evoked*) by indefinite noun phrases or proper names. Demonstratives and pronouns are generally anaphoric. Definite noun phrases are often anaphoric (as above), but often used to bring a mutually known and uniquely identifiable entity into the current discourse. e.g., *the president of the US*.

Sometimes, pronouns appear before their referents are introduced: this is *cataphora*. E.g., at the start of a discourse:

> Although she couldn't see any dogs, Kim was sure she'd heard barking.

both cases of *she* refer to Kim - the first is a *cataphor*.

7.6 Pronoun agreement

Pronouns generally have to agree in number and gender with their antecedants. In cases where there's a choice of pronoun, such as *he*/*she* or *it* for an animal (or a baby, in some dialects), then the choice has to be consistent.

(3) A little girl is at the door — see what she wants, please?

(4) My dog has hurt his foot — he is in a lot of pain.

(5) * My dog has hurt his foot — it is in a lot of pain.

Complications include the gender neutral *they* (some dialects), use of *they* with *everybody*, group nouns, conjunctions and discontinous sets:

(6) Somebody's at the door — see what they want, will you?

(7) I don't know who the new teacher will be, but I'm sure they'll make changes to the course.

(8) Everybody's coming to the party, aren't they?

(9) The team played really well, but now they are all very tired.

(10) Kim and Sandy are asleep: they are very tired.

(11) Kim is snoring and Sandy can't keep her eyes open: they are both exhausted.

7.7 Reflexives

(12) John$_i$ cut himself$_i$ shaving. (himself = John, subscript notation used to indicate this)

(13) # John$_i$ cut him$_j$ shaving. (i $\neq$ j — a very odd sentence)

The informal and not fully adequate generalization is that reflexive pronouns must be coreferential with a preceeding argument of the same verb (i.e., something it subcategorizes for), while non-reflexive pronouns cannot be. In linguistics, the study of inter-sentential anaphora is known as *binding theory*: I won't discuss this further, since the constraints on reference involved are quite different from those with intrasentential anaphora.

7.8 Pleonastic pronouns

Pleonastic pronouns are semantically empty, and don't refer:

(14) It is snowing

(15) It is not easy to think of good examples.

(16) It is obvious that Kim snores.

(17) It bothers Sandy that Kim snores.

Note also:

(18) They are digging up the street again

This is an (informal) use of *they* which, though probably not technically pleonastic, doesn't apparently refer to a discourse referent in the standard way (they = 'the authorities'??).

7.9 Salience

There are a number of effects which cause particular pronoun referents to be preferred, after all the hard constraints discussed above are taken into consideration.

Recency More recent referents are preferred. Only relatively recently referred to entities are accessible.

(19) Kim has a fast car. Sandy has an even faster one. Lee likes to drive it.

it preferentially refers to Sandy's car, rather than Kim's.

Grammatical role Subjects $>$ objects $>$ everything else:

(20) Fred went to the Grafton Centre with Bill. He bought a CD.

he is more likely to be interpreted as Fred than as Bill.

Repeated mention Entities that have been mentioned more frequently are preferred:

(21) Fred was getting bored. He decided to go shopping. Bill went to the Grafton Centre with Fred. He bought a CD.

He=Fred (maybe) despite the general preference for subjects.

Parallelism Entities which share the same role as the pronoun in the same sort of sentence are preferred:

(22) Bill went with Fred to the Grafton Centre. Kim went with him to Lion Yard.

Him=Fred, because the parallel interpretation is preferred.

Coherence effects The pronoun resolution may depend on the rhetorical/discouse relation that is inferred.

(23) Bill likes Fred. He has a great sense of humour.

He = Fred preferentially, possibly because the second sentence is interpreted as an explanation of the first, and having a sense of humour is seen as a reason to like someone.

7.10 Algorithms for resolving anaphora

Most work has gone into the problem of resolving pronoun referents. As well as discourse understanding, this is often important in MT. For instance, English *it* has to be resolved to translate into German because German has grammatical gender (though note, if there are two possible antecedants, but both have the same gender, we probably do not need to resolve between the two for MT). I will describe one approach to anaphora resolution and a modification of it that requires fewer resources.

7.11 Lappin and Leass (1994)

The algorithm relies on parsed text (from a fairly shallow, very broad-coverage parser, which unfortunately isn't generally available). The text the system was developed and tested on was all from online computer manuals. The following description is a little simplified:

The discourse model consists of a set of referring NPs arranged into equivalence classes, each class having a global salience value.

For each sentence:

1. Divide by two the global salience factors for each existing equivalence class.
2. Identify referring NPs (i.e., exclude pleonastic *it* etc)
3. Calculate global salience factors for each NP (see below)
4. Update the discourse model with the referents and their global salience scores.
5. For each pronoun:
   (a) Collect potential referents (cut off is four sentences back).
   (b) Filter referents according to binding theory and agreement constraints.
   (c) Calculate the per pronoun adjustments for each referent (see below).
   (d) Select the referent with the highest salience value for its equivalence class plus its per-pronoun adjustment. In case of a tie, prefer the closest referent in the string.
   (e) Add the pronoun in to the equivalence class for that referent, and increment the salience factor by the non-duplicate salience factors pertaining to the pronoun.

The salience factors were determined experimentally. Global salience factors mostly take account of grammatical function — they encode the hierarchy mentioned previously. They give lowest weight to an adverbial NP (achieved by giving every non-adverbial an extra positive score, because we want all global salience scores to be positive integers). Embedded NPs are also downweighted by giving a positive score to non-embedded NPs. Recency weights mean that intrasentential binding is preferred.

Global salience factors.

recency	100
subject	80
existential	70
direct object	50
indirect object	40
oblique complement	40
non-embedded noun	80
non-adverbial	50

The per-pronoun modifications have to be calculated each time a candidate pronoun is being evaluated. The modifications strongly disprefer cataphora and slightly prefer referents which are 'parallel', where parallel here just means having the same syntactic role.
Per pronoun salience factors:

cataphora	-175
same role	35

Applying this to the sample discourse:

> Niall Ferguson is prolific, well-paid and a snappy dresser.
> Stephen Moss hated him — at least until he spent an hour being charmed in the historian's Oxford study.

Assume we have processed up to '—' and are resolving *he*. Discourse referents:

N	*Niall Ferguson*, *him*	435
S	*Stephen Moss*	310

I am assuming that *a snappy dresser* is ignored, although it might actually be treated as another potential referent, depending on the parser.

N has score 155 + 280 ((subject + non-embedded + non-adverbial + recency)/2 + (direct object + head + non-adverbial + recency))
S has score 310 (subject + non-embedded + non-adverbial + recency) + same role per-pronoun 35

So in this case, the wrong candidate wins.

We now add *he* to the discourse referent equivalence class. The additional weight is only 80, for subject, because we don't add weights for a given factor more than once in a sentence.

N	*Niall Ferguson*, *him*, *he*	515

Note that the wrong result is quite plausible:

> Niall Ferguson is prolific, well-paid and a snappy dresser.
> Stephen Moss hated him — at least until he spent an afternoon being interviewed at very short notice.

The overall performance of the algorithm reported by Lappin and Leass was 86% but this was on computer manuals alone. Their results can't be directly replicated, due to their use of a proprietary parser, but other experiments suggest that the accuracy on other types of text could be lower.

7.12 Anaphora for everyone

It is potentially important to resolve anaphoric expressions, even for 'shallow' NLP tasks, such as Web search, where full parsing is impractical. An article which mentions the name 'Niall Ferguson' once, but then has multiple uses of 'he', 'the historian' etc referring to the same person is more relevant to a search for 'Niall Ferguson' than one which just mentions the name once. It is therefore interesting to see whether an algorithm can be developed which does not require parsed text. Kennedy and Boguraev (1996) describe a variant of Lappin and Leass which was developed for text which had just been tagged for part-of-speech.

The input text was tagged with the Lingsoft tagger (a very high precision and recall tagger that uses manually developed rules). Besides POS tags, this gives some grammatical function information: e.g., it notates subjects (for English, this is quite easy to do on the basis of POS-tagged text with some simple regular expressions). The text was then run through a series of regular expression filters to identify NPs and mark expletive *it*. Heuristics defined as regular expressions are also used to identify the NPs grammatical role. Global salience factors are as in Lappin and Leass, but Kennedy and Boguraev add a factor for context (as determined by a text segmentation algorithm). They also use a distinct factor for possessive NPs.

Because this algorithm doesn't have access to a parser, the implementation of binding theory has to rely on heuristics based on the role relationships identified. Otherwise, the algorithm is much the same as for Lappin and Leass.

Overall accuracy is quoted as 75%, measured on a mixture of genres (so it isn't possible to directly compare with Lappin and Leass, since that was only tested on computer manual information). Few errors were caused by the lack of detailed syntactic information. 35% of errors were caused by failure to identify gender correctly, 14% were caused because quoted contexts weren't handled.

7.13 Another note on evaluation

The situation with respect to evaluation of anaphora resolution is less satisfactory than POS tagging or WSD. This is partly because of lack of evaluation materials such as independently marked-up corpora. Another factor is the difficulty in replication: e.g., Lappin and Leass's algorithm can't be fully replicated because of lack of availability of the parser. This can be partially circumvented by evaluating algorithms on treebanks, but existing treebanks are relatively limited in the sort of text they contain. Alternatively, different parsers can be compared according to the accuracy with which they supply the necessary information, but again this requires a suitable testing environment.

7.14 Further reading

J&M discuss the most popular approach to rhetorical relations, *rhetorical structure theory* or RST. I haven't discussed it in detail here, partly because I find the theory very unclear: attempts to annotate text using RST approaches tend not to yield good interannotator agreement (see comments on evaluation in lecture 3), although to be fair, this is a problem with all approaches to rhetorical relations. The discussion of the factors influencing anaphora resolution and the description of the Lappin and Leass algorithm that I've given here are partly based on J&M's account.

The references below are for completeness rather than suggested reading:

Lappin, Shalom and Herb Leass (1994)
An algorithm for pronominal anaphora resolution,
Computational Linguistics 20(4), 535–561

Kennedy, Christopher and Branimir Boguraev (1996)
Anaphora for everyone: pronominal anaphora resolution without a parser,
Proceedings of the 16th International Conference on Computational Linguistics (COLING 96), Copenhagen, Denmark, 113–118

8 Lecture 8: Applications

This lecture considers three applications of NLP: machine translation, spoken dialogue systems and email response. This isn't intended as a complete overview of these areas, but just as a way of describing how some of the techniques we've seen in the previous lectures are being used in current systems or how they might be used in the future.

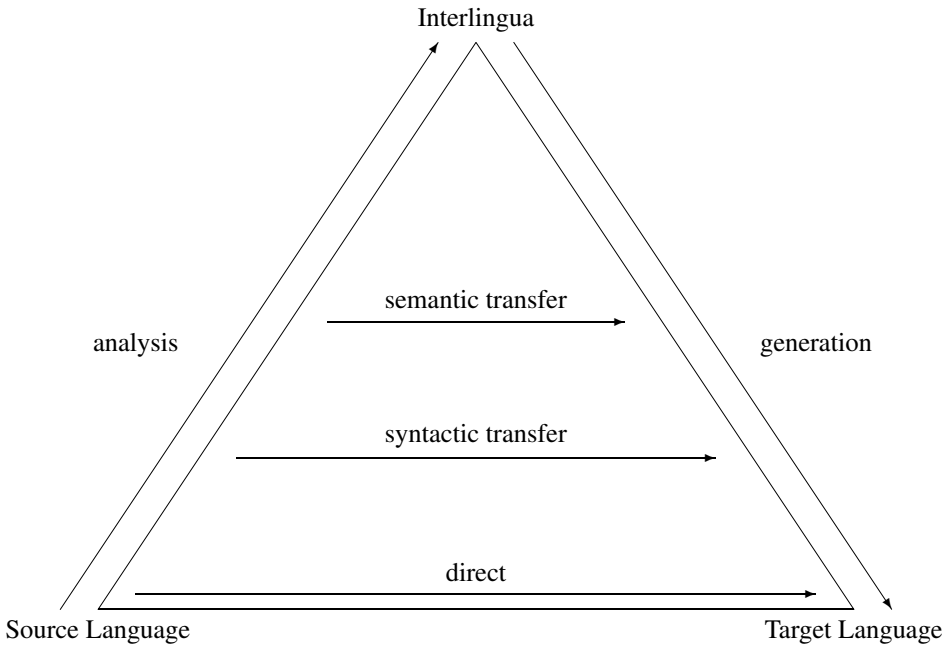
Machine translation

8.1 Methodology for MT

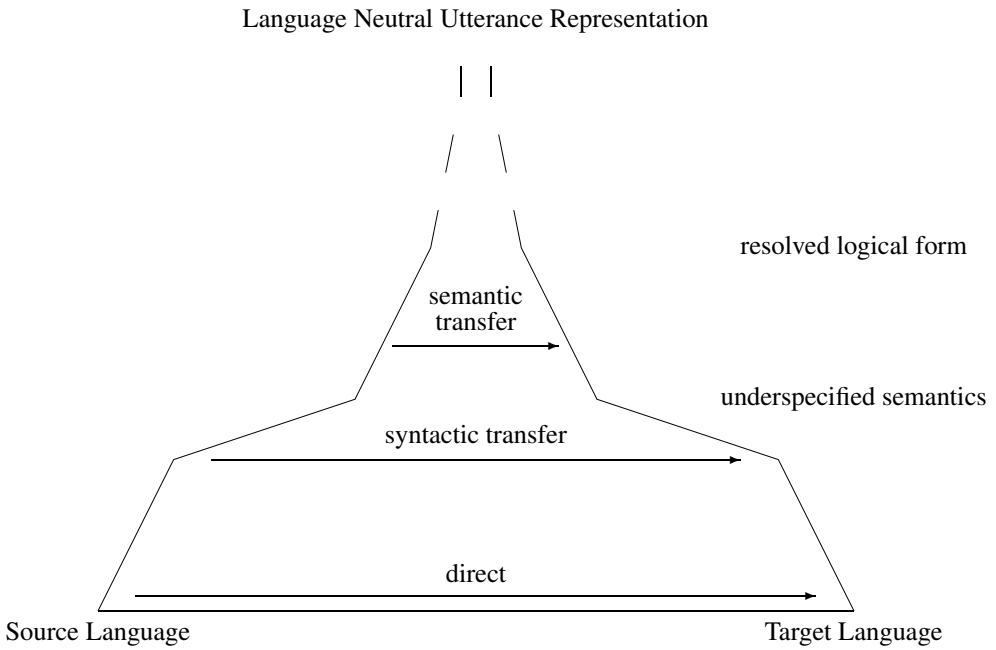
There are four main classical approaches to MT:

- Direct transfer: map between morphologically analysed structures.
- Syntactic transfer: map between syntactically analysed structures.
- Semantic transfer: map between semantics structures.
- Interlingua: construct a language-neutral representation from parsing and use this for generation.

The standard illustration of the different classical approaches to MT is the Vauquois triangle. This is supposed to illustrate the amount of effort required for analysis and generation as opposed to transfer in the different approaches. e.g., direct transfer requires very little effort for analysis or generation, since it simply involves morphological analysis, but it requires more effort on transfer than syntactic or semantic transfer do.



The Vauquois triangle is potentially misleading, because it suggests a simple trade-off in effort. It is at least as plausible that the correct geometry is as below (the Vauquois inverted funnel with very long spout):



This diagram is intended to indicate that the goal of producing a language-neutral representation may be extremely difficult!

Statistical MT involves learning translations from a *parallel corpus*: i.e. a corpus consisting of multiple versions of a single text in different languages. The classic work was done on the proceedings of the Canadian parliament (the Canadian Hansard). It is necessary to align the texts, so that sentences which are translations of each other are paired: this is non-trivial (the mapping may not be one-to-one). The original statistical MT approach can be thought of as involving direct transfer, with some more recent work being closer to syntactic (or even semantic) transfer.

Example-based MT involves using a database of existing translation pairs and trying to find the closest matching phrase. It is very useful as part of machine-aided translation.

8.2 MT using semantic transfer

Semantic transfer is an approach to MT which involves:

1. Parsing a *source language* string to produce a meaning representation
2. Transforming that representation to one appropriate for the *target language*
3. Generating from the transformed representation

Constraint-based grammars are potentially well suited to semantic transfer.

For instance:

Input:	Kim singt
Source LF:	named$(x,$ "Kim"$)$, singen(e,x)
Target LF:	named$(x,$ "Kim"$)$, sing(e,x)
Output:	Kim sings

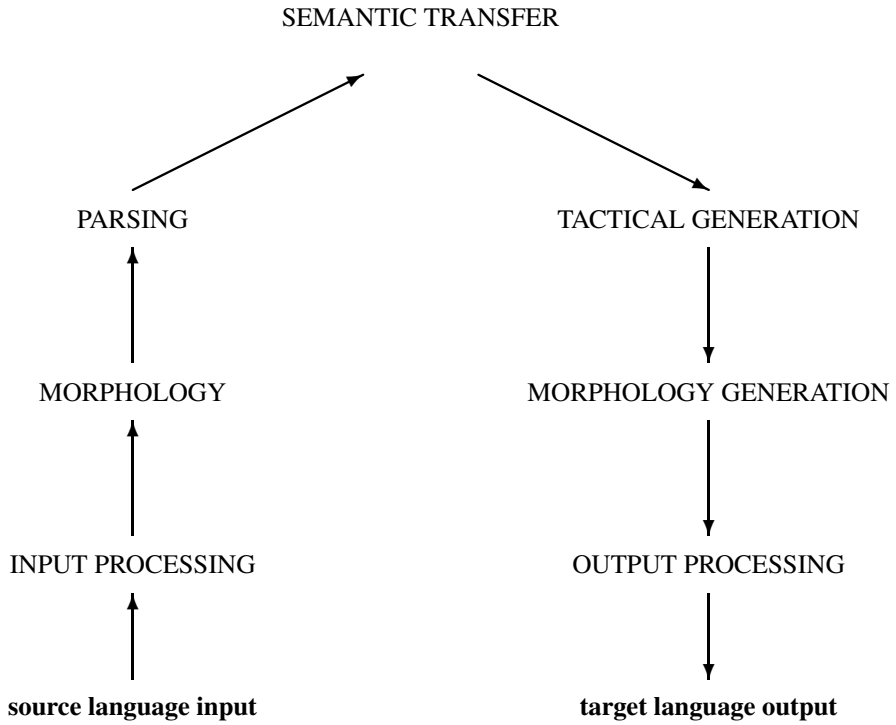
Transfer rules:

singen$(e,x) \leftrightarrow$ sing(e,x)

$\leftrightarrow$ indicates *transfer equivalence* or *translation equivalence*: the double arrow indicates reversibility:

Input:	Kim sings
Source LF:	named$(x,$ "Kim"$)$, sing(e,x)
Target LF:	named$(x,$ "Kim"$)$, singen(e,x)
Output:	Kim singt

'named' can be regarded as a language-neutral predicate, so no transfer is necessary. We also assume we don't change strings like "Kim".



Semantic transfer rules are a form of quasi-inference: they map between meaning representations. Obviously the example above was trivial: more generally some form of *mismatch* is likely to be involved, although the idea of semantic transfer is that there is less mismatch at the semantic level than at a syntactic level. Semantic transfer does not require that quantifier scope be resolved. Semantic transfer requires detailed bidirectional grammars for the languages involved, which currently makes it more suitable for high-precision, limited domain systems.

An anaphora resolution module is potentially needed when translating between languages like English and German, since English *it* can correspond to German *er*, *sie* or *es*, for instance. But the resolution should be done on an 'as-needed' basis, triggered by transfer, since it some contexts there is no ambiguity.

Some deployed MT systems use a form of semantic transfer, but syntactic transfer is more common. In these systems, generation is usually a form of text reconstruction, rather than 'proper' tactical generation. Direct transfer is used as a fallback if syntactic analysis fails. Systran uses a mixture of direct transfer and syntactic transfer: it works reasonably well because it has an enormous lexicon of phrases. Handling multiword expressions (MWEs) is a major problem in MT. Statistical MT is probably the commonest approach in the research community, followed by semantic transfer.

All MT systems require some form of WSD: potentially big improvements could be made in this area. One difficulty, however, is that MT systems often have to operate with rather small amounts of text, which limits the availability of cues.

Dialogue systems

8.3 Human dialogue basics

Turn-taking: generally there are points where a speaker invites someone else to take a turn (possibly choosing a specific person), explicitly (e.g., by asking a question) or otherwise.

Pauses: pauses between turns are generally very short (a few hundred milliseconds, but highly culture specific). Longer pauses are assumed to be meaningful: example from Levinson (1983: 300)

> A: Is there something bothering you or not? (1.0 sec pause)
> A: Yes or no? (1.5 sec pause)

A: Eh?
B: No.

Turn-taking disruption is very difficult to adjust to. This is evident in situations such as delays on phone lines and people using speech prostheses, as well as slow automatic systems.

Overlap: Utterances can overlap (the acceptability of this is dialect/culture specific but unfortunately humans tend to interrupt automated systems — this is known as *barge in*).

Backchannel: Utterances like *Uh-huh*, *OK* can occur during other speaker's utterance as a sign that the hearer is paying attention.

Attention: The speaker needs reassurance that the hearer is understanding/paying attention. Often eye contact is enough, but this is problematic with telephone conversations, dark sunglasses, etc. Dialogue systems should give explicit feedback.

Cooperativity: Because participants assume the others are cooperative, we get effects such as indirect answers to questions.

When do you want to leave?
My meeting starts at 3pm.

All of these phenomena mean that the problem of spoken dialogue understanding is very complex. This together with the unreliability of speech recognition means that spoken dialogue systems are currently only useable for very limited interactions.

8.4 Spoken dialogue systems

1. Single initiative systems (also known as system initiative systems): system controls what happens when.

   System: Which station do you want to leave from?
   User: King's Cross

   Generally very limited: for instance, in the example above the system won't accept anything that's not a station name. So it wouldn't accept *either King's Cross or Liverpool Street, depending on when the next train to Cambridge is*. Designing such systems tends to involve HCI issues (persuading the user not to complicate things), rather than language related ones.

2. Mixed initiative dialogue. Both participants can control the dialogue to some extent.

   System: Which station do you want to leave from?
   User: I don't know, tell me which station I need for Cambridge.

   The user has responded to a question with a question of their own, thereby taking control of the dialogue. Unfortunately, getting systems like this to work properly is incredibly difficult and although research systems have been built, practical performance is currently better if you don't allow this sort of interaction. The term 'mixed-initiative' is often used (somewhat misleadingly) for systems which simply allow users to optionally specify more than one piece of information at once:

   System: Which day do you want to leave?
   User: the twenty-third
   OR
   User: the twenty-third of February

3. Dialogue tracking. Explicit dialogue models may improve performance in other tasks such as spoken language machine translation or summarizing a human-to-human dialogue. Generally it's less critical to get everything right in such cases, which means broader domains are potentially realistic.

The use of FSAs in controlling dialogues was mentioned in lecture 2. Initial versions of simple SDSs can now be built in a few weeks using toolkits developed by Nuance and other companies: CFGs are generally hand-built for each dialogue state. This is time-consuming, but testing the SDS with real users and refining it to improve performance is probably a more serious bottleneck in deploying systems.

Email response using deep grammars

8.5 A large coverage grammar

The email response application that I mentioned in lecture 1 might be addressed using domain-specific grammars, but unlike in dialogue systems, it is much more difficult to make the limitations in the grammar obvious to the user (and if the coverage is very limited a menu-driven system might well work better). It is too expensive to manually build a new broad-coverage grammar for each new application and grammar induction is generally not feasible because the data that is available is too limited. The LinGO ERG constraint-based grammar mentioned in lecture 5 has been used for parsing in commercially-deployed email response systems. The grammar was slightly tailored for the different domains, but this mostly involved adding lexical entries. The ERG had previously been used on the Verbmobil spoken language MT task: the examples below are taken from this.

Indication of coverage of the ERG:

1. The week of the twenty second, I have two hour blocks available.
2. If you give me your name and your address we will send you the ticket.
3. Okay, actually I forgot to say that what we need is a two hour meeting.
4. The morning is good, but nine o'clock might be a little too late, as I have a seminar at ten o'clock.
5. Well, I am going on vacation for the next two weeks, so the first day that I would be able to meet would be the eighteenth
6. Did you say that you were free from three to five p.m. on Wednesday, the third, because if so that would be a perfect time for me.

Coverage was around 80% on Verbmobil.

Efficiency (with the PET system on an 850Mhz CPU):

Item	Word Length	Lexical Entries	Readings	First Reading	All Readings	Passive Edges
1	12	33	15	150 ms	270 ms	1738
2	15	41	2	70 ms	110 ms	632
3	15	63	8	70 ms	140 ms	779
4	21	76	240	90 ms	910 ms	5387
5	26	87	300	1460 ms	8990 ms	41873
6	27	100	648	1080 ms	1450 ms	7850

The ERG and other similar systems have demonstrated that it is possible to use a general purpose grammar in multiple applications. However, it is crucial that there is a fallback strategy when a parse fails. For email response, the fallback is to send the email to a human. Reliability of the automated system is extremely important: sending an inappropriate response can be very costly.

A big difficulty for email response is connecting the semantics produced by the general purpose grammar to the underlying knowledge base or database. This is expensive in terms of manpower, but does not require much linguistic expertise. Hence, this sort of approach is potentially commercially viable for organisations that have to deal with a lot of fairly routine email. Although tailoring the grammar by adding lexical entries is not too hard, it is much more difficult to manually adjust the weights on grammar rules and lexical entries so that the best parse is preferred: automatic methods are definitely required here. Much less training data is required to tune a grammar than to induce one.

8.6 Further reading

J&M discuss MT and spoken dialogue systems.