22/10/87

Parsing Context-free Grammars

1. Parsing

Having seen that natural languages are best described by means of a Context-free Grammar or equivalent formalism (perhaps with some augmentation to Indexed Grammar power), we turn to the question of computational implementation of a device (a parser) that can apply a grammar of this form and give us some useful output.

In turning from mathematics to computation, there are choices that have to be made. This is particularly true in the case of rewrite grammars, like CFGs, which seem, as mathematical devices, to abstract much more than the equivalent automata from the details of implementation on conventional von Neumann architectures.

One fundamental dimension of parsing strategy is whether we operate in a goal-driven or data-driven manner. Our goal is to prove that a string of words is an S, and our data is the string of words. A goal-driven strategy thus starts with S. This is matched against the lhs of a rule and replaced by the rhs. Some symbol is chosen, and matched against a rule with that as its lhs, and the process continues. If the string of words is a sentence, then we will eventually arrive at the string by rewriting.

A data-driven strategy starts with the string of words, and matches each against the rhs of a rule of the form PT → w, where PT is a non-terminal symbol representing a lexical category, often called a pre-terminal, and w is an actual word. Strings of pre-terminals will then be matched against the rhs's of further rules, introducing non-terminal symbols. The process continues until the whole string has been written as an S (if it is in the language).

Since the application of context-free rules imposes a heirarchical structure on the string, i.e. a syntax tree, and since syntax trees are normally written with their root at the top, we call goal-driven parsing **top-down** and data-driven parsing **bottom-up**.

Another dimension of choice in parsing strategy is the order in which we consider symbols or strings of symbols for matching against rules. Do we work **left-to-right**, or according to some other pattern, e.g. **island-driven**. A related question is the shape of the search space we engender by the order we perform rewritings. Do we try to rewrite those symbols that have been most recently rewritten, so that we do as much processing of one part of a string as we can before we look at another part; or do we try to rewrite each part in turn, before we rewrite any part again. Since the first strategy treats deepening the search tree at any point as a priority, it is called **depth-first**. The second strategy expands the search tree along a broad frontier, hence is described as **breadth-first**.

Finally, how do we handle the non-determinism inherent in the parsing process. Faced with multiple possibilities for rewriting, do we choose one and work on that exclusively, **backtracking** to an earlier choice point if we get into a blind alley; or do we work on alternative pathways simultaneously. Since we currently have only sequential machines at our disposal, what this choice amounts to is whether we encode the non-determinism behind the scenes, in the control regime, or 'up front' as a complex data structure which stores simultaneous possibilities.

It will turn out that the optimal strategies for natural language parsing involve not being rigid about choices at the extreme ends of the above dimensions.

However, we will look first at one particular combination of the above strategies that is

interesting by virtue of being directly embodied in the interpreter for the programming language Prolog. Although it is a sub-optimal combination in various ways, it has the advantage that one doesn't need to write a parser to implement it.

We will also take the opportunity to introduce ways of doing linguistically useful things that are made particularly easy by the Prolog view of the world.

The particular strategies that the Prolog interpreter uses are top-down, left-to-right, depth-first, and backtracking.

It is important to emphasize that the notation used for **Definite Clause Grammars (DCGs)**, grammars that can be directly interpreted by Prolog, is conceptually independent of the strategies embodied in the Prolog interpreter. The problems that arise in interpreting DCGs directly are a function of those strategic decisions, not a function of the notation, and one can easily build a parser (perhaps in Prolog) to interpret the DCG notation according to alternative strategies.

Time flies like an arrow.

NP	Verb	PP	DET	NP	STATEMENT.
ADJ	N	Verb	Det	NP	IMPERATIVE OF "TIME FLIES"
Verb	N	PREP	DET	NP.	IMPERATIVE

Grammar

S → NP VP

S → VP

NP → Det

NP → NP2

NP → NP PP

NP2 → Noun

NP2 → Adj NP2

PP → Prep NP

VP → Verb

VP → Verb NP

VP → VP PP

noun; verb, adjective preposition

Major lexical categories.

26/10/87

A Brief Review of the Syntactic Categories and Structure of English

1. Morphological Categories

We can assign the words of English to categories according to various criteria. The most robust of these tend to be morphological. We recognise that a certain word occurs in several distinct forms, which we call a **morphological paradigm**. The three major ones in English are those of the noun, verb and adjective. Common nouns typically have distinct singular and plural forms (e.g. carton, cartons). Note that certain nouns don't show this alternation, e.g. sheep, deer, and the class of mass nouns, e.g. furniture, toast, though the latter can often be used as common nouns with the meaning 'type of'. Adjectives typically have a base form, an adverb form, a comparative and a superlative, e.g great, greatly, greater and greatest, though many adjectives are defective. Non-gradable adjectives don't have the latter two forms, e.g. unique, uniquely; and polysyllabic adjectives build these forms periphrastically, e.g. beautiful, more beautiful, most beautiful. Verbs have at most eight distinct forms: in the case of the verb 'to be': be, am, are, is, was, were, been, being. Many verbs have only five distinct forms: do, does, did, done, doing. Totally regular verbs also conflate the third and fourth of these: bake, bakes, baked, baking. Syntactically, the important distinctions are between the finite forms (present 3rd sing - writes, other present - write, past - wrote) and the non-finite forms (bare infinitive - write, infinitive - to write, present participle - writing, past participle - written, passive participle - written).

The paradigms presented above are normally referred to as **inflectional morphology**, that is, the various forms are considered to be forms of the same word. Another type of morphology often distinguished is **derivational morphology**, which can be considered the process whereby new words, often of different syntactic category, are built. Typical derivational processes are those that build nouns out of adjectives (great - greatness), verbs out of nouns (parameter - parameterise), adjectives out of nouns (academy - academic), and adjectives out of adjectives(likely - unlikely, probable - improbable). Also note the so-called **zero-derivation** in English, which is used to describe noun-verb pairs that have the same form (wolf, bat, fox, dog etc.). The defining characteristic of derivational morphology seems to be its lack of productivity. Derivational affixes are generally rather limited in the number of stems they can attach to, and even when they are not, as in the zero-derivation case, the semantic relation betwen the two words concerned tends to be idiosyncratic.

In describing morphological processes, a distinction can be made between the specification of which affixes go in which order on which sort of stems, **morphotactics**, and the phonetic and orthographic changes that are engendered by such processes of affixation, **morphophonemics and morphographemics**. It is the first of these that should tell us about the bracketing of a word such as 'impossibility' = ((impossible)ity), or the ambiguity of unloadable = (unload)able or un(loadable). The second describes such processes as the doubling of the consonant in e.g. shop, shopping.

2. Syntactic Categories

A fourth class of word, preposition, is added to the three above to give the major lexical categories in English. These are of syntactic import in that they each license phrasal categories having a distinctive structure. We can often recognise categories intermediate between the lexical categories and the phrasal categories **maximal projections**, they correspond to. To describe this phenomena, many syntactic theories use the notion of **bar level**. Lexical categories are of the form X^0, phrasal categories are X^2 or sometimes X^3, and intermediate categories are notated accordingly. One advantage of this sort of notation

26/10/87

is that it allows regularities in the structure of different phrases to be described concisely.

The following types of phrase are recognised.

Noun phrases are typically used to refer to objects, but note the use of the dummy nps 'there' and 'it', as in (1) and (2).

(1) There is a dog howling in the yard

(2) It is impossible for me to see you now

Noun phrases are built by rules of the form:

pre_mod = pre_modifier etc.

np → pronoun
np → proper_noun
np → det n^1
n^1 → pre_mod n^1
n^1 → n^1 post_mod
n^1 → noun comps
np → np appos_mod

complements
opposite_modifier =

Prepositional phrases are generally of the form:

pp → p np

But note forms as in (3)

(3) the mouse ran out from under the table

pp → p pp

Adjective phrases are described by the rules:

ap → deg a^1
a^1 → adv a^1
a^1 → adj comps

Verb phrases are described by rules such as:

vp → v^1
v^1 → v^1 adjunct
v^1 → v comps

The details of what gets introduced at what bar level may vary from grammar to grammar. However, the general intuition is quite clear. A particular lexical item licenses certain complements, which are more tightly bound to it than modifiers. The latter can typically occur in any number with any word of the class. We say that a certain item subcategorises certain complements, and these have to be associated with that item in the lexicon. A standard way to do this is to associate a feature with a word, called its 'subcat' feature, whose value is referred to in a specific phrase structure rule, e.g.

```
n1 → n(31) pp(with) pp(about)    % argument
n1 → n(34) vp(inf)               % plan
a1 → a(26) s(fin)                % afraid
a1 → a(24) pp(of)                % fond
```

$v^1 \rightarrow$ v(5) np np % give
$v^1 \rightarrow$ v(10) s(bse) % prefer

Notice how the complements of an item are always maximal projections. Another important characteristic of complements is that their head imposes **selectional restrictions** upon them. That is, a particular head only makes sense with a particular type of complement, semantically speaking. For example, 'kill' selects for a living object, but is not particular about its subject. 'Murder' requires an entity that can be considered responsible for its actions as its subject, while 'assassinate' requires the rather specialised semantic property 'political figure' to be true of its object.

A given lexical item may occur in several different sub-categorisation patterns. For instance, there is a class of verbs like 'give' that occur in the patterns 'give y x' and 'give x to y'. It is a simple matter to enter these alternatives in the lexicon, or at least to have some productive process going on in the lexicon that produces them both. There is no need to import powerful transformational apparatus to relate them by giving them a common syntactic structure at some abstract level. The same is true of all similar phenomena, e.g. passive, that can be considered purely locally.

3. Long-distance dependencies

There is, however, a class of phenomena that are not purely local, but seem to involve a dependency between elements of the sentence that are separated by an unbounded number of clause boundaries. Examples are **topicalisation**, **relativisation** and **tough-movement**, as in (4)-(6).

(4) This book, I could never manage to persuade my students to read.

(5) The college that I expected John to want Mary to attend.

(6) This film is easy for me to persuade the children not to see.

However, the trend in modern linguistics is to account for these by means of a series of purely local dependencies, in which some information is passed step-by-step through the syntax tree. This style of analysis is illustrated in the definite clause grammar given below, which uses a 'gap threading' technique to identify the filler of a relative clause with a corresponding gap.

26/10/87

```
s0 → s(nogap,nogap).

s(T0,T) →
  np(T0,T1),
  vp(T1,T).

np(T,T) →
  det,
  n,
  rel.

np(T,T) → proper_noun.

np(gap,nogap) → [].

vp(T,T) → v(1).

vp(T0,T) →
  v(2),
  np(T0,T).

vp(T0,T) →
  v(3),
  vp(T0,T).

vp(T0,T) →
  v(4),
  np(T0,T1),
  vp(T1,T).

rel →
  relp,
  s(gap,nogap).

rel → [].

v(1) → laugh
v(2) → love
v(3) → hope
v(4) → persuade
```

→ gap hasn't been dropped in subtree structure

— gap dropped in this substructure

← gap dropped in subtree.