

Ambiguity and the Lexicon in Natural Language

Informatics 2A: Lecture 11

Bonnie Webber

School of Informatics
University of Edinburgh
bonnie@inf.ed.ac.uk

20 October 2009

- 1 Ambiguity in Language
 - Derivations and Structural Ambiguity
 - Dealing with Ambiguity
- 2 The Lexicon
 - Word Classes
 - Parts of Speech
 - Part of Speech Ambiguity
 - Word Frequency

Readings:

J&M (2nd edition) Ch. 5 (intro through 5.2)

NLTK Book: Chapter 3, Processing Raw Text

Reminder: NLTK labs start this week (Wed/Fri)

Review: Derivations

- Recall from Lecture 4 that **equivalent derivations** are ones that only differ in the **order** of non-terminal expansion. (Leftmost, rightmost, middle-most, arbitrary – It doesn't matter.)
- Recall also that the set of **equivalent derivations** of a string from a context-free (CF) phrase structure grammar (PSG) can be represented as a **tree**.
- This is because a tree makes no commitment as to the order in which non-terminals are expanded.
- But also recall that **not all** derivations of a given string from a given grammar are equivalent.

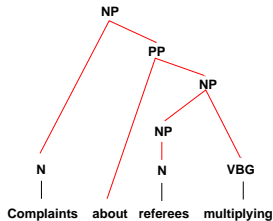
Example

$NP \rightarrow NP\ VBG$
 $NP \rightarrow N\ PP$
 $NP \rightarrow N$
 $PP \rightarrow \textit{about}\ NP$
 $N \rightarrow \textit{complaints} \mid \textit{referees}$
 $VBG \rightarrow \textit{multiplying}$

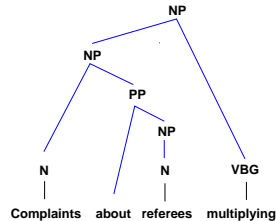
Consider the newspaper headline:

*complaints about referees multiplying*How many **non-equivalent** sets of derivations (ie, different trees) are there for this string?

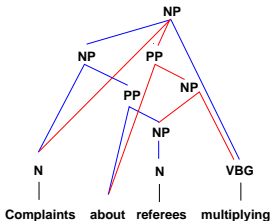
Headline announcing new complaints



Headline announcing new trend in complaints



Complaints about referees multiplying



Derivations and structural ambiguity

- Given a grammar, those strings that can be associated with more than one tree (i.e., non-equivalent derivations) are called **structurally ambiguous**.
- Of course, an **agent** who produces a structurally ambiguous string usually only has one meaning in mind, so only one of the structures corresponds to what s/he intended.

Example: Newspaper Headlines

stolen painting found by tree
lung cancer in women mushrooms
dealers will hear car talk at noon
miners refuse to work after death
juvenile court to try shooting defendant

Avoiding Ambiguity

The designers of formal languages (e.g., XML) or programming languages try to eliminate or reduce structural ambiguity. For example, Python uses indentation to indicate **embedding** and no indentation to indicate **sequence**.

```
if a<b:
  c = 0
  a = a+1
```

vs.

```
if a<b:
  c = 0
a = a+1
```

Avoiding Ambiguity

When we talk, we can use speech rate, pauses and emphasis to indicate what we intend.

Example

lung cancer in WOMEN | mushrooms
dealers will hear CAR.TALK at noon

Also, one reading usually makes more sense in the circumstances than other readings do (cf. Lectures 21–25 on *Semantics*).

These are both reasons why we don't normally notice that what we read, hear and/or say can have multiple analyses (and multiple meanings!).

Handling Ambiguity

- Given a string from a language, the role of a **parser** is to deliver either its most likely structure or all its possible structures. In the latter case, another procedure will assess and choose among them.
- Later on, we'll look at various techniques that parsers use to do this efficiently.
- NLTK and Python allow us to study parsers without having to build them ourselves.
- But **structural ambiguity** is not the only form of ambiguity in language.
- Natural Languages can also have **part-of-speech ambiguity** – ambiguity as to what **class(es)** (aka “parts of speech”) a word belongs to.

Word Classes in Formal/Programming Languages

Every grammar for describing a language contains

- a set of non-terminal symbols
- a set of terminal symbols (Σ) that appear in its strings.

But even within Σ , we can distinguish two classes:

- those symbols that convey information about the structure of a string and the roles that other symbols play.

Example

FOL: $S \rightarrow (\forall\exists) \text{ Variable Formula}$

Python: $S \rightarrow \text{for Var in ListOrDictionary} : S^+$
 $S \rightarrow \text{from Module import Namelist}$

- all other symbols

Word Classes in Formal/Programming Languages

Sometimes, structuring symbols (eg. `for`, `in`, `import`, `if`, `else`, `while`, etc.) are **reserved**: They can't be used elsewhere in strings.

Example

Propositional logic: A OR (AND AND C)

Python:

```
>>> four = 7.0
>>> for = 7.0
File <stdin>, line 1
    for = 7.0
    ~
SyntaxError: invalid syntax
```

Lexicon in Natural Languages

- Words (and punctuation) comprise the terminal symbols in (the written form of) a Natural Language. (What corresponds to punctuation in spoken language? Is it the same for all human languages?)
- But NL grammars are **largely** specified in terms of the **classes** that words belong to.
- Several broad word classes are found in all Indo-European languages and in other language families as well: **nouns**, **verbs**, **adjectives**, **adverbs**.
- Other word classes are more specific to particular languages: **prepositions** (English, German), **post-positions** (Hungarian, Urdu, Korean), **particles** (Japanese), **classifiers** (Chinese), etc.

Parts of Speech

How do we tell what word class (**part of speech**) a word belongs to?

At least three different criteria can be used:

- Notional** (semantic) criteria: What does the word refer to?
- Formal** (morphological) criteria: What does the word look like?
- Distributional** (syntactic) criteria: Where is the word found?

We will look at different parts of speech (POS) using these criteria.

Nouns

Notionally, nouns generally refer to living things (*mouse*), places (*Scotland*), things (*harpoon*), or concepts (*marriage*).

Formally, *-ness*, *-tion*, *-ity*, and *-ance* tend to indicate nouns.

Example: happiness, exertion, levity, significance

Distributionally, we can examine the contexts where a noun appears and at other words that appear in the same contexts.

```
>>> from nltk.book import *
>>> text1.concordance('harpoon') # Where 'harpoon' appears in MD
>>> text1.similar('harpoon') # What else appears in such contexts?
>>>
>>> text2.concordance('marriage') # Where 'marriage' appears in S&S
>>> text2.similar('marriage') # What else appears in such contexts?
```

Verbs

Notionally, verbs refer to actions (*observe, think, give*).

Formally, words that end in *-ate* or *-ize* tend to be verbs, and ones that end in *-ing* are often the present participle of a verb.

Example: automate, calibrate, equalize, modernize; rising, washing, grooming.

Distributionally, we can examine the contexts where a verb appears and at other words that appear in the same contexts, which may include their arguments.

```
>>> from nltk.book import *
>>> text2.concordance('marry') # Where 'marry' appears in S&S
>>> text2.similar('marriage') # What else appears in such contexts?
```

Adjectives

Notionally, adjectives convey properties of or opinions about things that are nouns (*small, wee, sensible, excellent*).

Formally, words that end in *-al*, *-ble*, and *-ous* tend to be adjectives.

Example: formal, gradual, sensible, salubrious, parlous

Distributionally, adjectives usually appear before a noun or after a form of *be*.

```
>>> from nltk.book import *
>>> text2.concordance('sensible') # Where 'sensible' appears in S&S
>>> text2.similar('sensible') # What else appears in such contexts?
```

Adverbs

Notionally, adverbs convey properties of or opinions about actions or events (*quickly, often, possibly, unfortunately*) or adjectives (*really*).

Formally, words that end in *-ly* tend to be adverbs.

Distributionally, adverbs can appear next to a verb, or an adjective, or at the start of a sentence.

```
>>> from nltk.book import *
>>> text2.concordance('highly') # Where 'highly' appears in S&S
>>> text2.similar('highly') # What else appears in such contexts?
```

The importance of formal and distributional criteria

Often in reading, we come across words we've never seen before (**unknown words**).

bootloader, distros, whitelist, diskdrak, borked
(<http://www.linux.com/feature/150441>)
revved, femtosecond, dogfooding (<http://hardware slashdot.org/>)

Even if we don't know its meaning, formal and distributional criteria help people (and machines) recognize what class an unknown word belongs to and what the sentence would mean, if we knew what the word meant.

Example: I really wish mandriva would redesign the *diskdrak* UI. The "orphan" bit is *borked*.

Other Word Classes

Other word classes vary from language to language. English has

- determiners: *the, any, a, ...*
- prepositions: *in, of, with, without, ...*
- conjunctions: *and, because, after, ...*
- auxiliaries: *have, do, be*
- modals: *will, may, can, need, ought*
- pronouns: *I, she, they, which, where, myself, themselves*

English doesn't have clitics (like French *l'*) or particles (like Japanese *ga*). Russian lacks stand-alone reflexive pronouns.

N.B. Functions performed by words in one language may be performed by morphology in another one (e.g., reflexivity in Russian).

Lexical Ambiguity

Two important types of lexical ambiguity:

Part of Speech (PoS) Ambiguity: e.g., *still*:

- 1 *adverb*: at present, as yet
- 2 *noun*: (1) silence; (2) individual frame from a film; (3) vessel for distilling alcohol
- 3 *adjective*: motionless, quiet
- 4 *transitive verb*: to calm

Sense Ambiguity: e.g., *intelligence*:

- 1 Power of understanding
- 2 Obtaining or dispersing secret information; also the persons engaged in obtaining or dispersing secret information

Word Frequency – Properties of Words in Use

Take any large corpus of English like the **Brown Corpus** or the **BNC** and sort its words by how often they occur.

Rank	Word	Tokens	Freq	Rank	Word	Tokens	Freq
1	The	69970	6.8872	...			
2	of	36410	3.5839	30	they	3610	0.3562
3	and	28884	2.8401	31	which	3561	0.3505
4	to	26154	2.5744	32	one	3297	0.3245
5	a	23363	2.2996	33	you	3286	0.3234
6	in	21345	2.1010	34	were	3284	0.3232
7	that	10594	1.0428	...			
8	is	10102	0.9943	130	never	698	0.0687
9	was	9815	0.9661	131	day	695	0.0684
10	He	9542	0.9392	132	same	686	0.0675
11	for	9489	0.9340	...			
12	it	8760	0.8623	1531	realize	69	0.0068
13	with	7290	0.7176	1532	seek	69	0.0068
14	as	7251	0.7137	1533	willing	69	0.0068
15	his	6996	0.6886	1534	League	69	0.0068
16	on	6742	0.6636	...			
17	be	6376	0.6276	1998	plenty	55	0.0054
18	at	5377	0.5293	1999	mile	55	0.0054
19	by	5307	0.5224	2000	components	55	0.0054
20	I	5180	0.5099	...			

<http://www.edict.com.hk/lexiconindex/frequencylists/words2000.htm>

What can we observe?

- The most frequent word ('**the**') occurs 10 times more often than the word at rank 15 ('**his**').
- The second most frequent word ('**of**') occurs 10 times more often than the word at rank 30 ('**they**').
- The word at rank 15 ('**his**') occurs 10 times more often than the word at rank 130 ('**never**').
- The word at rank 130 ('**never**') occurs 10 times more often than the word at rank 1531 ('**realize**').
- None of the most frequent words denote things in the world like words do at lower ranks. Top-ranked words convey relations (eg, '**with**') and formal properties (eg, '**and**').

Zipf's Law

- NL text has been found to obey a **power law** called **Zipf's law**.
- This states that word frequency in a corpus is **inversely proportional** to word rank.
- As a power law, Zipf's law has two main features:
 - A small subset of words will account for half the **word tokens** in the corpus.
 - There will be a **long tail** of words that occur only once.
- Given that any corpus is only a subset of all possible texts, only the set of all texts is likely to contain all the words in the language (at a given time).

The top 135 words account for half of the word tokens (~ 500k) that make up the Brown Corpus. The rest (around 50,000 word types, most of which occur only once) constitute the other half!

Summary

- **Structural ambiguity** occurs when a string can be associated with more than one structure (represented as trees).
- Words in a language fall into different classes.
- **Open classes**, such as nouns and verbs, are found in many languages and are often class-preserving under translation.
- Other classes vary from language to language, and may not preserve their class under translation.
- To identify the class or **part-of-speech** (PoS) of a word, we can use **notional**, **distributional**, and/or **formal** criteria.
- **Lexical ambiguity** occurs when a word belongs to more than one part-of-speech class or has more than one sense.
- Words are found in a **Zipfian distribution**.