Assignment 2

Alex Fine
LSA 125, UC Berkeley

July 14, 2009

1 Background

Domain minimization accounts of sentence production predict that, given a choice between multiple ways of encoding the same message, speakers will prefer orders that minimize the distance between dependent phrasal heads (Hawkins 1994, Hawkins 2004; for more evidence compatible with this account, see Wasow 1997, Lohse, Hawkins, and Wasow 2004, Choi 2007, Yamashita and Chang 2001).

Hawkins (2000) tests the predictions of this theory on the ordering of post-verbal prepositional phrases in English. Given a choice between (1) and (2), domain minimization predicts that speakers will be more likely to choose (1), *ceteris paribus*.

(1) John [$_{VP}$ ate [$_{PP1}$ at noon] [$_{PP2}$ in the kitchen]].

(2) John [$_{VP}$ ate [$_{PP1}$ in the kitchen] [$_{PP2}$ at noon]].

Hawkins (2000) addresses this question using a corpus of written English.

2 Assignment

Your assignment is to take the first steps necessary to test whether Hawkins' findings are borne out in a corpus of spoken English. This will involve several steps.

2.1 Construct a TGrep2 search pattern

First start by thinking about what exactly you want to look for. We want verb phrases that dominate two PPs, where the PPs are sisters (we don't want cases where one PP dominates the other).
Here's a first approximation.[1]

```
/^VP/=VP1 < (/^PP/=PP1 $ /^PP/=PP2)
```

The problem with this pattern is that "$" alone simply means "is a sister to", without any order information. Since order is the main thing we're interested in, that is not ideal. Therefore, we want to use either 1) "`$.`"—*is a sister to and immediately precedes* or 2) "`$..`"—*is a sister to and precedes*. The second one is preferable if we are interested in looking at, say, disfluencies intervening between the two PPs.

```
/^VP/ =VP1 < (/^PP/=PP1
            $.. (/^PP/=PP2))
```

However, this will allow *anything* to intervene between the two PPs. We want to allow disfluencies only.

```
/^VP/=VP1 < (/^PP/=PP1
           $.. (/^PP/=PP2
           !,, (* !< *,, =PP1
           !>> (EDITED|UH|PRN/-UNF/ >> =VP1))))
```

Which means...we want a VP which immediately dominates a PP (PP1) which in turn is a sister to and precedes a PP (PP2), and PP2 does not follow a terminal node ("`* !< *`") following PP1, unless that terminal node is a disfluency.
...Finally, this pattern does not rule out cases with more than two PPs, and we don't want more than two. Therefore we make a slight modification (on the third line).

[1] Since we don't care what part of speech the VP or the PPs are, we use regular expressions (covered in lecture 2); also, you'll want to label the VP and each of the PPs, so that they can be kept track of.

```
/^VP/=VP1 < (/^PP/=PP1
          $.. (/^PP/=PP2
          !$ (/^PP/ != =PP1)
          !,, (* !< * ,, =PP1
          !>> (EDITED|UH|PRN|/-UNF/ >> =VP1))))
```

This specifies that PP2 is not the sister to a PP that is not PP1.

This pattern is a mouthful and very difficult to read. The best way to clean it up is to create a set of **macros**, which can be called in your TGrep2 pattern. A macro is just a bit of TGrep2 syntax that can be used with other TGrep2 code like a variable. An especially good candidate for a macro in our example is the part which identifies disfluencies. You should add this to MACROS.ptn located in `<insert project name>/shellscripts`. Refer to the TDT manual for tips on how to edit this file.

```
@<space>DISFL<tab>/EDITED|UH|PRN|-UNF/;
```

The "@" is followed by a single space, and "DISFL" is followed by a tab.
It may also be useful to create macros for terminal nodes, VPs, and PPs:

```
@<space>TERMINAL<tab>* !< *;
@<space>VP<tab>/^VP/;
@<spacePP<tab>/^PP/;
```

Now, you can use these in the TGrep2 search pattern above. But note that when you call the macros in the pattern, the "@"s will not be followed by spaces.

```
@VP=VP1 < (@PP=PP1
         $.. (@PP=PP2
         !$ (@PP != =PP1)
         !,, (@TERMINAL ,, =PP1
         !>> (@DISFL >> =VP1))))
```

Also, this pattern can itself be saved as a macro (again, in MACROS.ptn), which is good because you probably want to type it as few times in your life as possible, which I have learned while making this assignment. When saving this in the MACROS.ptn file, the syntax is the same as before.

```
@<space>PPPP<tab>@VP=VP1 < (@PP=PP1 $.. (@PP=PP2 !$ (@PP != =PP1) !,,
(@TERMINAL ,, =PP1 !>> (@DISFL >> =VP1))));
```

Now, if you want to find those glorious post-verbal PPs, you can just do some variation of this:

```
tgrep2 MACROS.ptn "@PPPP"
```

2.2 Create a database

Create a database from the extracted data using one of the three methods described in the TDT manual in section 3. The database will need to include at least the following factors:

- Length of each PP.
- Whether the NP complement of the PP is a pronoun
- the verb
- verb lemma
- verb type (transitive, intransitive, etc.)
- whether PPs are of the form [P NP]
- part of speech of the verb

If you find that some factors are not possible or practical (given resource limitations) to include, please make a note of these. Also, if there are other factors you think would be interesting to look at, please feel free to include them. The database you create will be in a form that can be read by R (which is relevant to your third assignment).

Remember that portions of this assignment do not have obviously "right" or "wrong" solutions (though some do). Part of the goal of this assignment is to force you to think through all of the technical and conceptual decisions faced by researchers doing corpus work.

Your write-up for this problem should be 1-3 pages long. In your write-up, you should

- Start by describing the TGrep2 search pattern you used to identify the outcome of interest (VPs with two adjacent PPs).
- State how many cases you found.
- For each predictor variable we asked you to extract, please provide basic distributional information, specifically:
  - For categorical predictors, state how often each of its levels (values) occurs in your data set. Also provide a table of how often each level occurs with which PP order.
  - For continuous predictors, state the minimum, mean, and maximum value. If you know how to do it, create a plot that shows the predictor on the x-axis and the percentage of PP-orders on the Y-axis.
- Please provide the TGrep2 pattern for each predictor in an appendix.
- Provide a brief summary statement of the findings. In particular, discuss which of your findings conform to Hawkins' predictions and which ones do not.

Suggestions for collaborations: Recall that you are encouraged to collaborate on this problem. You can divide up the work such that, say, each person extracts the information for one column in your database (e.g. one person finds PP length, another pronominality, etc.).
We suggest a minimum of 3 and no more than 5 collaborators per assignment. You should choose different collaborators for at least one assignment. You are also encouraged to take advantage of the office hours. Meet with us (in groups or alone), ask questions, and discuss ideas. You can find our office hours and location on the class website.

References

Choi, H.-W. (2007). Length and order: A corpus study of korean dative-accusative construction. *Discourse and Cognition 14*(3), 207–227.

Hawkins, J. A. (1994). *A Performance Theory of Order and Constituency*, Volume 73 of *Cambridge Studies in Linguistics*. Cambridge, UK: Cambridge University Press.

Hawkins, J. A. (2000). The relative order of prepositional phrases in english: Going beyond manner-place-time. *Language Variation and Change 11*, 231–266.

Hawkins, J. A. (2004). *Efficiency and complexity in grammars*. Oxford: Oxford University Press.

Lohse, B., J. A. Hawkins, and T. Wasow (2004). Domain minimization in english verb-particle constructions. *Language 80*(2), 238–261.

Wasow, T. (1997). Remarks on grammatical weight. *Language Variation and Change 9*(1), 81–105.

Yamashita, H. and F. Chang (2001). "long before short" preference in the production of a head-final language. *Cognition 81*, 45–55.