Using Excel to do simple morphological parsing

(1) Goal for today:
- Look at a very simple procedure for parsing morphemes using Excel
- Practical reason: not all linguists are comfortable writing computer programs or using fancier, more “traditional” computational techniques (though perhaps we should be!), but all linguists should be comfortable with a database or spreadsheet package, such as Microsoft Excel
- Using Excel will force us to take the problem step by step, and we can examine the intermediate stages

(2) The task: take a list of words and parse them into their morphemes
- Figure out which morphemes are present
- Figure out what the meaning and function of the word is

(3) Start up Excel and create a new file
- Let’s put some words in the first column, such as:
  cat, cats, mat, mats, dog, dogs, foot, feet, bark, barks, meowing, the, dog, is, walking, the, cat, sat, on, the, mat
  - They are listed here as a list with commas to save space; in Excel, it is most convenient to list them going doing the left of the first column, with no commas
- Our task will be to recognize these words analytically; that is, without having to list each inflected form separately—e.g., meowing = meow–PRES.PART.

(4) Step 1: Let’s create a lexicon
- Go to a new, empty worksheet (switch worksheets using the tabs at the bottom of the window)
- You can double-click on the tab to rename the sheet; let’s name it nouns
- Make sheets called noun_suff, verbs, verb_suff, and func (you might need to create more worksheets, using the “Insert worksheets” command

(5) Now let’s put some lexical items into these worksheets; starting with nouns
- The reason for listing things twice (in both columns) will become apparent later
- Make some entries for verbs, too

(6) Adding suffixes to the lexicon:
(7) Also for some verbs:
- In the \textit{nounsuff} worksheet, add whatever verb roots seem necessary (\textit{walk, bark, meow})
- And in the \textit{verbsuff} worksheet:

\begin{tabular}{|c|c|}
\hline
\textbf{s} & 3SG. PRES. \\
\textbf{ing} & PRES. PART. \\
\hline
\end{tabular}

(8) Finally, in the \textit{func} worksheet, add function words like the, a, on, etc.

(9) Now, go back to the list of words that we're trying to parse.
- Our strategy will be to try to parse these words by seeing if they can be analyzed as known words, or legal combinations of known words and suffixes
- In order to do this, we will use Excel's built-in functions

(10) Using functions in Excel
- The format of entering a function in Excel is: \texttt{=FUNCTIONNAME(ARGUMENT1, ARGUMENT2, \ldots)}
- For example, \texttt{=SUM(2, 2)} will give you 4.
- You can refer to values in other cells with \textit{references}, which are codes consisting of a letter+number. The letter refers to which column the cell is in, and the number is the row.
  - For example, the function \texttt{=A1} simply puts a copy of the value from cell A1 into the current cell
  - While you are typing a function, you can get a reference to a cell simply by clicking on that cell (that is, you don't always need to enter the code manually)
  - By default, Excel treats cell references as \textit{relative}, meaning if you enter \texttt{=A1} in cell B1, the function is really saying "Put a copy of whatever is immediately to my left"
  - If you want to refer literally to cell A1, you can make the reference \textit{absolute} with dollar signs: \texttt{=$A$1}
  - See class demo for an example of the difference
- Make sure that the "View Formula Bar" option is selected, so that you can see your formulas at the top of the screen.

(11) Some of the functions that will come in handy for us in this task:

\textbf{String manipulation:}
- \texttt{LEFT(string, n)}: returns just the left \textit{n} characters of the string—e.g., \texttt{=LEFT("cat", 2)} gives you cat
- \texttt{RIGHT(string, n)}: same as \texttt{LEFT}, but on the right
- \texttt{CONCATENATE(string1, string2, string3, \ldots)}: concatenates all of the listed strings into a single string—e.g., \texttt{=CONCATENATE("c", "a", "t")} gives you cat
- \texttt{LEN(string)}: returns a value that is the length of a string (how many characters it contains)

\textbf{Comparing values in different cells:}
- \texttt{IF(condition, action if true, action if false)}: puts different values into the cell, depending on whether condition is true or not—e.g., \texttt{=IF(2=2, "yes", "no")} returns the value yes (because 2 does equal 2), but \texttt{=IF(2=3, "yes", "no")} will yield no
- \texttt{EXACT(string1, string2)}: compares string1 and string2, and returns \texttt{TRUE} if they are identical, and \texttt{FALSE} if not\footnote{It is also possible to compare strings with the '=' operators (\texttt{string1=string2}). Unfortunately, the '=' operator is not case sensitive, so \texttt{blah=BLAH} comes out as true. Phonetic symbols often involve case differences (such as 's' for [s] and 'S' for [ʃ]), so it is a good idea to get into the habit of using \texttt{EXACT} rather than '=' for string comparison.}
And now the most complex one, which is also the one we need for this parsing task:

\[ \text{VLOOKUP(string,table,column,FALSE)} \] looks up string in a table to see what its value is in a certain column. (See example below for an illustration)

Combining functions:

- \[ \text{AND(function1, function2)} \] returns TRUE if both functions come out true; FALSE otherwise
- \[ \text{OR(function1, function2)} \] returns TRUE if one or the other function comes out as true; FALSE otherwise
- \[ \text{NOT(function)} \] reverses the value

Breaking up the task:

- What would be some good steps to take in deciding whether a given word is either a known word or a legal concatenation of known morphemes?
- What would be a good order to try them in?

Looking up a word in the lexicon:

- Go to the list of words that you want to parse, and highlight the cell to the right of the first word (cell B1)
- Enter the following formula: \[ =\text{VLOOKUP}(A1,\text{nouns!A:B,2,FALSE)} \]
  - This says to look up the values of cell A1 in the worksheet called nouns, in the table in columns A and B, and if it's there, return the value from the 2nd column (column B). The FALSE is an annoying technicality; it means that Excel should try to match the string exactly, not approximately.
  - If all has gone well, you should get the value cat, meaning that your word is contained in the lexicon.
  - You can do the same for verbs and for function words in other columns

That was the easy part (looking up words in the lexicon); how about determining if something is a legal concatenation of a root plus a suffix?

- Many possible approaches. I’ll illustrate just one very simple one
- We’ll make use of the fact that suffixes are relatively short, and occur one at a time (*bark-s-ing)
- Strategy: Take small chunks off the end of the word and see if the chunk is a suffix and what’s left is a root

Checking for suffixes with length 1:

- The last character of the string in cell A1 is: \[ \text{RIGHT(A1,1)} \]
- The string that’s left if we take that off is: \[ \text{LEFT(A1,LEN(A1)-1)} \]
- So we want to see if \[ \text{LEFT(A1,LEN(A1)-1)} \] is in the list of nouns, and \[ \text{RIGHT(A1,1)} \] is a possible noun suffix:
  - Checking if the “root” is a known noun: \[ \text{VLOOKUP(LEFT(A1,LEN(A1)-1),nouns!A:B,2,FALSE)} \]
  - Checking if the last character is a known suffix: \[ \text{VLOOKUP(RIGHT(A1,1),nounsuff!A:B,2,FALSE)} \]
  - Putting it together: \[ \text{=CONCATENATE(VLOOKUP(LEFT(A1,LEN(A1)-1),nouns!A:B,2,FALSE)},",", \text{VLOOKUP(RIGHT(A1,1),nounsuff!A:B,2,FALSE)})} \]
  - This looks terrible, but the pieces should now be clear!
  - Paste this formula down a column, and see what values it returns!
  - How would we check for a verb suffix of length 1? For a suffix with a length > 1?

A useful refinement: irregulars

- How can we recognize feet as foot-PL? Where is the best place to get this information? (Similarly is as be-3SG)