J1799d: Spell Checking with Lextrees

Due: 17 Nov 2014

This homework builds on your previous spellchecker assignment. We have two problems:

Problem 1: Write a spell checker that loads a dictionary as a lexical tree.

- The lexical tree stores the collection of words as a tree of characters, eliminating redundancy in their spellings. An example of a lextree representation of the words "bat", "battle" and "banana" might look like:

```
b--a--t
|  |--t--t--l--e
|
|--a--n--a-n--a
```

  Note that although the "t" is common to both "bat" and "battle", we actually do NOT share the "t" between the two. This is because the "t" in "bat" is the last character of the word. By not sharing it, we ensure that each leaf of the tree uniquely identifies a word.

- The conventional string matching procedure cannot deal with deletions and insertions in the first position of a word. To deal with this, we will introduce a dummy symbol "*" at the beginning of each word. Thus, our list of words above gets modified to "*bat", "*battle" and "*banana". The lextree becomes:

```
*--b--a--t
   |  |--t--t--l--e
   |
   |--a--n--a-n--a
```

  To account for the "*" we will also prepend a "*" to the words in our test data. For instance, if we intend to spellcheck the word "brat", we'd expand it first to "*brat". Now both deletions and insertions in the first position can be accommodated.

- Dictionary words may begin with one of 26 characters (in English). By including a "*" in the initial position, we are also able to create a single lextree for all words in the dictionary. For instance, if our vocabulary

consisted of "a", "an", "and", "apple", "bat", "battle" and "banana" our lextree would be:

```
*--a
|
|--a--n
|  |--n--d
|  |--p--p--l--e
|
|--b--a--t
   |  |--t--t--l--e
   |
   |--a--n--a-n--a
```

Note that "a" is not shared with "an", "and" or "apple". Similarly, the "n" in "an" is not shared with "and". This is to ensure that each leaf of the tree represents a unique word.

- Use relative pruning with a beam width of 3.

For the spellchecker use the dictionary at dict_1.txt Run the spellchecker on typos.txt.

Problem 2: Use the lextree structure to also automatically segment the text in unsegmented0.txt and unsegmented.txt to find word boundaries (and insert spaces) in the right places and (for the second file) simultaneously spellcheck the words in it. To do this, permit a transition back from the leaves of the lextree back to the "*". The location of "*" in the best path identifies word boundaries.

Note that the procedure above is likely to make many errors. Can you think of any variation to the procedure that may result in better segmentation?

For problem 2, try relative beam widths of 5,10, and 15. Compare the segmentation and corrected spellings in segmented.txt to determine which works best. The "accuracy" of an output is computed as the difference in the number of words in the hypothesized segmentation and the number of words in the correct transcription PLUS the number of mispelled words. If possible, plot accuracy as a function of beam width.