Assignment 2: Segmentation of different languages

Your assignment this week is to explore how frequency-based segmentation strategies work in different conditions. You will make up some artificial languages, and you will try the strategies on some real languages other than English. The files for this assignment can be found in the course locker, in the directory Assignments/Assignment2

Part 1: $n$-grams in artificial languages

In class on Wednesday, we discussed how to construct an artificial language, examine its bigram transition probabilities, and try to use them to re-segment the language into words. In theory, it is possible to construct a language in which the within-word and across-word bigram transitional probabilities are so different that a text could be segmented perfectly solely on the basis of these probabilities. (In other words, running SegmentBigrams.pl on a text in this language would produce all of the right word boundaries.) Conversely, it is also possible to construct a language in which this procedure would fail completely. Your task is to construct one of each.

1. First, construct a language in which the transitional probabilities between words are consistently lower than the transitional probabilities within words, and re-segmentation works perfectly.
   - It should have six words; make a small text file containing the words, and call it Artificial-Segmentable.vocab.
   - Then make a text of 1800 words (300 of each vocabulary item), as described in Wednesday's handout; call it Artificial-Segmentable.txt.
   - Run ComputeBigrams.pl and SegmentBigrams.pl on it to confirm that it really can be re-segmented perfectly. (Hint: you may need to adjust the $initial\_transition\_threshold$ variable at the beginning of SegmentBigrams.pl, depending on the exact transitional probability values in your artificial language). For your convenience, I have made duplicate copies of these perl scripts in the Assignment2 directory.

2. Then, construct a language in which the transitional probabilities between words are not consistently lower than the probabilities within words.
   - In this language, it should be impossible to find a setting of the $initial\_transition\_threshold$ variable that allows accurate segmentation.
   - Call the file of vocabulary items Artificial-Unsegmentable.vocab, and create a Artificial-Unsegmentable.txt with a complete 1800 word text in this language.
   - Run ComputeBigrams.pl and SegmentBigrams.pl on it to confirm that it cannot be re-segmented very accurately.

What is it about these languages that makes them easy or hard to segment based on transitional probabilities? Explain what the relation is between the structures of your artificial words and the bigram transitional probabilities that you get for the texts.

Part 2: Application to natural languages

Different languages have radically different phonological structures, and different orthographies.

- Hawai’ian has a very small phoneme inventory, and allows only CV (consonant+vowel) or plain V syllables. Thus, an unsegmented Hawai’ian text is going to have a lot of alternating consonant+vowel pairs, with occasional vowel+vowel sequences as well.
Polish has a slightly larger consonant inventory than English, and many of its consonant phonemes are written with digraphs (e.g., sz for ⟨ʃ⟩, etc.) Furthermore, Polish syllables can contain long strings of consonants, as in grzbiet ‘back’, and so on. To an English speaker, Polish texts appear to have huge long strings of consonants (and indeed, they do!).

Given what you learned in Part 1, what predictions can you make about the successfulness of \( n \)-gram based segmentation in these languages? Your task in this section is to see whether these predictions are correct.

- The Assignment2 directory has some relevant files in it:
  - HawaiianCorpus.txt is a small corpus of Hawai’ian texts
  - HawaiianStory.txt is a short story, to try segmenting
  - PolishCorpus.txt is a somewhat bigger (but haphazard) corpus of Polish texts
  - PolishStory.txt is a short story, to try segmenting

- You should compute the bigrams from the corpus files, and then use the results .bigrams files to try segmenting the short stories. You may need to adjust the threshold value at the beginning of SegmentBigrams.pl, in order to compensate for different probability profiles of the different languages. See how well you can get the script to perform for each language.

- If you want, you can use SegmentationAccuracy.pl to test the overall accuracy of the results, but you will learn more by actually comparing and inspecting the files.

- A technical note: Polish uses a bigger alphabet than English does, and unfortunately, unix terminals are often not well-equipped to handle Unicode. The files will appear to have a lot of weird junk in them, but they will run just fine.
  - If you are using the Terminal from Mac OS X, you can make things look a lot better by doing the following: select Show Info (command-I), and use the pop-up menu at the top to select the Display options. In the “Character Set Encoding” options, switch to Unicode (UTF-8). This won’t make things perfect, but it helps a lot.
  - Other ssh clients can probably handle Unicode in some fashion too, but I’m not sure what the instructions are for your particular platform

Are your predictions true? Why or why not?

**What you will turn in to me**

Please send/submit/give me the following things:

- Part 1: Submit your vocabulary and text files, either by using the submit command (submit ling163-aa.s03 Assignment2 Artificial-Segmentable.vocab, etc.) If you need to use a different threshold value to get these files to work/fail, please make a note of it in your write-up (see below)

- Part 2: You do not need to submit any files for part 2.

- Write-up: please give me a brief write-up of your findings. For part 1, explain what properties make it easy/hard your artificial languages work/fail. For part 2, explain your predictions (and why), and whether or not they were right. Try to diagnose why the files behave in the way they do, even if it is not what you would have predicted. Also, if you have to adjust the probability threshold for segmentation in any of your simulations, let me know what you used.

---

1You will also need to have a Unicode-enabled font selected; if things still look weird, try changing the font setting in the same page of preferences. I personally recommend Lucida Sans Typewriter