DD1418 Language Engineering with Introduction to Machine Learning
Johan Boye
2019-11-03

Assignment 3

Readings: Read chapter 8.4 and sections A1-A4 in Jurafsky-Martin.

Code: The skeleton code can be downloaded from Canvas or from

```
http://www.csc.kth.se/~jboye/teaching/language_engineering/a03/HMM.zip
```

Unzip the code in your home directory. Go to the folder `HMM` and type:

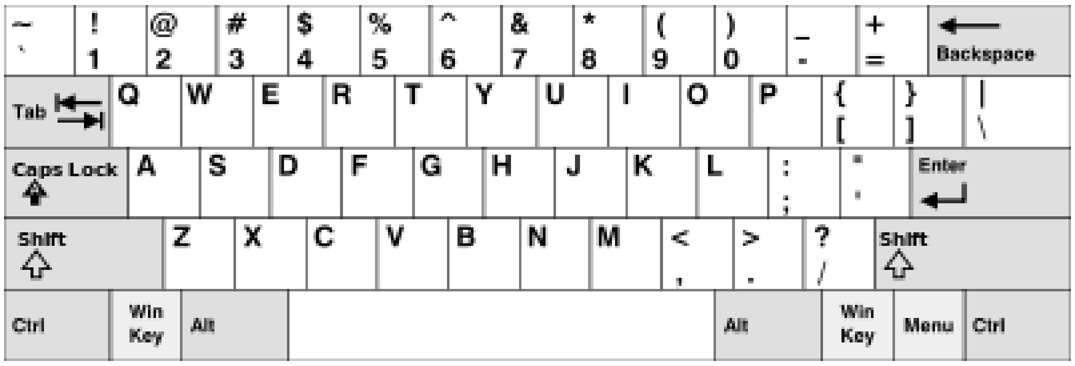
```
pip install -r requirements.txt
```

Now everything needed for the assignment should be installed.

Problems:

1. As a simplified model of keystroke errors, let us assume that a person typing a text has a 0.1 probability of pressing a key neighboring the intended key by mistake. For instance, if we mean to type an A, then a Q, a W, an S, or a Z might be produced instead with probability 0.1 each. The probability of actually hitting an A when we intend to is $1.0 - 0.4 = 0.6$.

   To simplify the problem further, we will disregard all keys on the keyboard except A-Z and the SPACE key, and assume that we always manage to hit the SPACE key with probability 1, and that the SPACE key is never pressed by mistake.

   Given a text which contains keystroke errors according to the distribution outlined above, **your task is to recreate the intended text as well as possible**. To your help, you have two files containing letter-bigram probabilities and letter-trigram probabilities, respectively.

   Have a look in the file `bigram_probs.txt`. Each line in the file contains three numbers. The two initial integers encode letters: 0 for 'a', 1 for 'b', ..., 25 for 'z', and finally 26 for the SPACE/START/END symbol, representing word boundaries, as well as the beginning and end of a sentence. The final number of each line is the (natural logarithm of the) bigram probability.

Continued on the next page.

For instance,

```
0 1 -3.748896861435106
```

represents that $P(\text{“ab”}) = P(\text{b}|\text{a}) = -3.748896861435106$, while

```
0 26 -2.481764189851945
```

expresses the probability for the bigram “’a’ followed by space/end-of-sentence”. Similarly, the line

```
0 1 2 -5.969906514008791
```

in the file `trigram_probs.txt` represents that $P(\text{“abc”}) = P(\text{c}|\text{ab}) = -5.969906514008791$.

(a) Explain how this model of keystroke errors can be cast as a Hidden Markov Model. Which are the **hidden states**, the **observations**, the **state transition probabilities**, and the **observation probabilities**?

(b) The `ViterbiBigramDecoder` program contains a code skeleton for applying the Viterbi algorithm to the text correction problem, making the text more legible. Extend the code so that it works correctly (look for the comments `YOUR CODE HERE` and `REPLACE THE STATEMENT BELOW WITH YOUR CODE` in the program). Try your implementation on some test cases by running the script `run_bigram_decoder.sh`. You may compare with the expected results in the file `test_bigram_decoding.txt` and/or by using the `--check` flag.

Hint: Note that the program adds a `START_END` symbol at the end of the input string. To recover the result, your implementation can simply follow the backpointers from the `START_END` state in the final time step.

2. Perhaps a better result can be obtained by including more context into the model?

(a) Implement the Viterbi algorithm using trigram probabilities by extending the class `ViterbiTrigramDecoder`, so that it works correctly. Try your implementation on the test cases by running the script `run_trigram_decoder.sh`, and compare with the expected results in the file `test_trigram_decoding.txt`, and/or by using the `--check` flag.

(b) Run your bigram and trigram decoders on the 5 files `mistyped_1.txt` to `mistyped_5.txt`. Report the output of the two programs.

(c) What are the topics of these five texts? Can you identify (or guess) the sources?