Please note that some of the resources used in this assignment require a Stanford Network Account and therefore may not be accessible.

A Greedy Decoder for Programming Assignment 2

- Some changes have been made to the decoder for PA2. If you've made changes to these files, You might want to make a backup of your *.java files before you copy the new ones over.
  Files added/changed in /afs/ir/class/cs224n/pa2/
  - `java/src/cs224n/assignments/DecoderTester.java`
  - `java/src/cs224n/decoder/Decoder.java`
  - `java/src/cs224n/decoder/GreedyDecoder.java`
  - `java/src/cs224n/decoder/Hypothesis.java`
  - `java/src/cs224n/metrics/SegStats.java`
  - `java/src/cs224n/util/Alignment.java`

(The old starter code is in `/afs/ir/class/cs224n/pa2/java.old` in case you need them)

- Options added for "DecoderTest"
  - "-lmweight" : a weight for the log(English LM prob) of a hypothesis. Default = 0.6
  - "-transweight" : a weight for the log(WordAlignment prob) of a hypothesis (source, target and alignment). Default = 0.4
  - "-lengthweight" : a weight for the length(e) of the final score. This is to prevent the decoder to predict really short sentence in order to get higher LM score. Default = 1.0

- Example usage:
  - First, run
    ```
    ./run-dec
    ```
    You'll notice that there are lots of "`trans prob:-Infinity`" in the output. It's because the `getAlignmentProb` method in `BaselineWordAligner` always return 0, and the "`trans prob`" in the output is the logarithmic of `getAlignmentProb`.
  - Once you have your Model 1 and Model 2 aligner implemented, try running:
    ```
    ./run-dec -wamodel cs224n.wordaligner.YourModel1WordAligner
    ./run-dec -wamodel cs224n.wordaligner.YourModel2WordAligner
    ```

- Output:
  - At the end of each run, you'll get numbers like this:
    ```
    WER: 0.6165982959924267

    BLEU: 0.11349539343790904
    --> log Ngram Scores: -0.5171279110178139/
    -1.6198672087310861/-2.620151512608046/-3.3524072174927233
    --> BP: 0.8619098752385267 (refLen=3169.0,hypLen=2759.0)
    ```

- Once your Word Aligner works reasonably on AER, you're <u>required</u> to compare the performance your Model 1 and Model 2 on the decoder with the default weights and the default LM. Other than that, there are more things you can try:
  - Error analysis of the guessed English sentences you get
  - Use different LMs you implement in PA1
  - Tune the 3 weights to see what different results you get
  - Run on different languages
  - Look into the decoder and see if there's anything you can do to improve it (speed, performance, anything you find interesting), etc.

- What numbers should we expect to get?
  Here's the number I got with Model 1 and Model 2, using the EmpiricalUnigramLanguageModel. The language listed are the different source languages (translating into English). You might get different results based on your implementation details. The BLEU scores here are quite low compared to state-of-the-art systems. Keep in mind this is a simple greedy decoder, and the original paper was using higher IBM models. The important point is to see if your Model 2 performs better than Model 1, and see if you can find something interesting on the error analysis.

BLEU	french	german	spanish
Model 1	0.009614	0.006381	0.013086
Model 2	0.113495	0.06767	0.087263

Brevity Penalty	french	german	spanish
Model 1	0.15646	0.128221	0.165661
Model 2	0.86191	0.557334	0.632112

WER	french	german	spanish
Model 1	0.784475	0.796668	0.764453
Model 2	0.616598	0.649779	0.6125