Decoding

The file `decode` has the code for stack decoding. The best corpus log probability score was achieved with a stack size of 100 and number of translations (k) 10.

The decode file has comments describing what has been changed. The main difference is in how the code iterates over the french phrases, and how the new hypotheses are stored in stacks. Instead of iterating left to right over the french phrase, I iterate over the whole thing, and try to find a segment that has not been translated. The stacks are now indexed based on how many words have been translated. I also keep track of what words have been translated for each hypothesis.

This file can be run with the following syntax:

```
python decode -k 10 -s 100
```

The extension is in decode-ext. I have chosen to implement a greedy decoder, following [1]. The stack decoder was used to provide a seed to the greedy decoder. The greedy decoder finds neighbors of the seed (using the operations given below), and tries to improve decoding.

These neighour operations that have been implemented are:

1. Swap: swap two adjacent phrases in the target language
2. Replace: replace any one translated phrase with another translation
3. Bi-replace: replace any two adjacent translated phrases with alternate translations
4. Split: split a source phrase into 2, and generate translations for both
5. Merge: merge 2 source phrases, and generate a translation for this

One more deviation from the paper is the score function: I have kept it the sum of translation model and language model log probabilities. In addition to this, the paper also uses a distortion model, and subtracts the number of translated words from the score. This decision was made because the final scoring is based on the language model and translation model scores only, and thus I did not want my model to optimize in some other direction.

This file can be run using the following command:

```
python decode-ext
```

The greedy decoder improves upon some sentences given by the stack decoder. However, these improvements were also gained from changing hyperparameters in the stack decoder only.

1. Langlais, Philippe, Alexandre Patry, and Fabrizio Gotti. "A greedy decoder for phrase-based statistical machine translation." *Proc. of TMI* (2007).

decode-ext

Since the greedy decoder code is somewhat dense (and resembling spaghetti), I have included a walk through for it. You may choose to skip this.

For each french sentence, I first find the best translation using a stack decoder. I feed this to the greedy decoder (decode_greedy()).

This function first initializes by generating a list of english and french phrase pairs. At each point, I also keep track of the tm score for the best sentence (so that I don't have to recalculate this every time a new hypothesis is tested).

Next, for each iteration, I generate neighbors for the current best sentence. This is with the neighbors() function. The function applies the operators given on page 1 to the current translation. It returns a 2d list, where each entry is a list of [operation, new french-english pairs, change in translation model score, number of french words translated].

The [change in translation model score] and [number of french words translated] are obsolete variables - I had used the latter to try build a score function similar to the paper, and the former to avoid having to calculate the translation model score too often.

Once the list of neighbors is generated, we check the scores for each of them. If the score of any is higher than the current best score, the best hypothesis is updated. The program continues iterations until the best score becomes stable.

1. Langlais, Philippe, Alexandre Patry, and Fabrizio Gotti. "A greedy decoder for phrase-based statistical machine translation." *Proc. of TMI* (2007).