Homework 2 - Decoding

Release Date: Friday, Feb 3rd, 2023
Due Date: Tuesday, Feb 21st, 2023 at 11:59PM

Note: For homeworks you can either work independently or in your course project groups.

Decoding is process of taking input in one language (e.g. French):

```
honorables sénateurs , que se est - il passé ici , mardi dernier ?
```

and searching for the best translation in the target language (e.g. English):

```
honourable senators , what happened here last Tuesday ?
```

To decode, we need a model of English sentences conditioned on the French sentence. In this homework you are provided with a **phrase-based translation model** $P_{LM}(f, a|e)$ and an **n-gram language model** $P_{LM}(e)$. Your challenge is to find the most probable English translation under the model.

$$
\begin{aligned}
\mathbf{e}^* &= \arg\max_{\mathbf{e}} (\mathbf{e}|\mathbf{f}) \\
&= \arg\max_{\mathbf{e}} \frac{p_{TM}(\mathbf{f}|\mathbf{e}) \times p_{LM}(\mathbf{e})}{p(\mathbf{f})} \\
&= \arg\max_{\mathbf{e}} p_{TM}(\mathbf{f}|\mathbf{e}) \times p_{LM}(\mathbf{e}) \\
&= \arg\max_{\mathbf{e}} \sum_{\mathbf{a}} p_{TM}(\mathbf{f},\mathbf{a}|\mathbf{e}) \times p_{LM}(\mathbf{e})
\end{aligned}
$$

Within the homework package **nlp267-hw2.zip** you will find the python program `decode.py`, which contains a simple but complete decoding algorithm.

```
> python decode.py > translations
 Reading translation model from data/tm...
 Reading language model from data/lm...
 Decoding data/input...
```

This creates the translations of the 48 French sentences in `data/input`.

You can then compute $p(e|f)$ using `compute-model-score.py`

```
> python compute-model-score.py < translations
 Reading translation model from data/tm...
 Reading language model from data/lm...
 ...
 Total corpus log probability (LM+TM): -1439.873990
```

`compute-model-score.py` sums over all possible ways that the model could have generated each English sentence from the French input, including translations that permute the phrases. This sum is intractable, but the phrase dictionary is fixed and sparse, so we can compute it. This is still easier to do than to search for the optimal translation. We suggest that you look at this command (`compute-model-score.py`) to get some hints about how to do the assignment.

The decoder (`decode.py`) generates the most probable translations that it can find, using three common approximations.

(1) It uses the Viterbi approximation to find the most probable translation. Instead of computing the intractable sum over all alignments for each sentence, we simply find the best single alignment and use its translation.

(2) It translates French phrases into English without changing their order. So, it only reorders words if the reordering has been memorized as a phrase pair.

For example, in the first sentence, we see that **`mardi dernier`** is correctly translated as **`last Tuesday`**. If we consult `data/tm`, we will find that the model has memorized the phrase pair **`mardi dernier ||| last Tuesday`**.

But in the second sentence, we see that **`Comité de sélection`** is translated as **`committee selection`**, rather than **`selection committee`**, which is the correct translation in English.

To show that this is a search problem rather than a modeling problem, we can generate the latter output by hand and check that the model actually prefers it.

```
> head -2 data/input | tail -1 > example

> python decode.py -i example |
  python compute-model-score.py -i example
    ...
    Total corpus log probability (LM+TM): -20.150189

> echo "a selection committee was achievement . " |
  python compute-model-score.py -i example
    ...
    Total corpus log probability (LM+TM): -16.717326
```

The scores are reported as log-probabilities, and higher scores (with lower absolute value) are better. We see that the model prefers `selection committee`, but the decoder does not consider this word order.

(3) Finally, our decoder uses strict pruning. As it consumes the input sentence from left to right, it keeps only the highest-scoring partial hypothesis up to that point. You can vary the number of partial hypotheses stored at each point in the translation using the `-s` parameter.

```
> python decode.py | python compute-model-score.py
    ...
    Total corpus log probability (LM+TM): -1439.873990

> python decode.py -s 10000 | python compute-model-score.py
    ...
    Total corpus log probability (LM+TM): -1436.360138
```

In Homework 2 your task is to implement a beam-search decoder which is capable of reordering. Our model assumes that any segmentation of the French sentence into phrases followed by a one-for-one substitution and permutation of those phrases is a valid translation.

We make the simplifying assumption that segmentation and ordering probabilities are uniform across all sentences, hence constant. This means that $p(f, a|e)$ is the product of the n-gram probabilities $P_{LM}(e)$ and the phrase translation probabilities in $P_{LM}(f, a|e)$. To avoid numerical underflow we compute the probabilities in logspace. The baseline decoder works with log probabilities, so you can simply follow what it does.

$$\arg\max_{\mathbf{e}} \max_{\mathbf{a}} \log p_{TM}(\mathbf{f}, \mathbf{a}|\mathbf{e}) + log\ p_{LM}(\mathbf{e})$$

When implementing reordering any permutation of phrases can be considered as a valid translations, so we strongly suggest searching over all or part of this larger search space. This search is NP-Hard, however you can trade efficiency for search effectiveness by implementing histogram pruning, threshold pruning, and/or by using reordering limits.

To improve the word alignment further you are welcome to explore other methods for a bonus of up to 20% for this homework. However, when exploring other methods:

- You must implement the majority of the algorithm yourself. Any functions or code you use from other repositories must be explicitly cited in your source code and writeup
- You cannot use any existing SMT decoding toolkits (such as MOSES)
- You must obtain advance permission to use other software libraries, toolkits or resources

Gradescope Submission:

1. **Homework 2 - Code (80%)**: You can upload your decoder as often as you like up until the homework deadline. The file that you submit to “Homework 2 - Code” must be named `decode.py`
2. **Homework 2 - Write up (20%)**: Provide a short written description (approx 1-page) of what you implemented describing the overall algorithm and the important functions and data structures in your implementation

Credits: This homework assignment is adapted from prior works from Philipp Koehn, John DeNero, Chris Dyer and Adam Lopez.