Homework #4: CMPT-825

Anoop Sarkar – anoop@cs.sfu.ca

Only the question marked with † will be checked. The others are optional.

(1) **Paraphrasing**

A *paraphrase* of a sentence is an alternative method to render the same or similar information. Use a language model to find the corpus probability of a corpus and its paraphrase. Report which version is *better* according to the language model. The data is available at `/cs/natlang-data/kjv-bbe`

A 5-gram language model in ARPA format is available at:
`/cs/natlang-data/wmt10/lm/eparl_nc_news_2m.en.lm`

The kenlm language model package is available at: `http://kheafield.com/code/kenlm/`. For `x86_64` machines the LM in kenlm binary format: `eparl_nc_news_2m.en.binlm`. Loading the binary version is much faster.

(2) **Using Moses**

Build a machine translation system using Moses for a language pair of your choice from the European Parliament (EuroParl) corpus: `http://statmt.org/europarl`. Follow the step by step instructions given in `http://www.statmt.org/moses/?n=Moses.Tutorial`.

(3) † **Decoder for Phrase-based Statistical Machine Translation**

Given an input sentence $\mathbf{f}$ we wish to produce the best translation $\mathbf{e}$ according to a phrase-based machine translation model which uses a model $\Pr(\mathbf{e} \mid \mathbf{f})$ in order to find $\mathbf{e}$. The basic model consists of the following components:

1. the phrase translation probability $\phi(\bar{f} \mid \bar{e})$.
2. the re-ordering model $d(x) = \alpha^{\text{abs}(x)}$, and $\alpha \in (0, 1]$ is set by hand or tuning data.
3. the language model $p_{\text{LM}}(\mathbf{e})$.

The source sentence $\mathbf{f}$ is split up into I phrases $\bar{f}_i, 1 \leq i \leq I$. However, this split is not unique. There are many different values of I and we search over all of them. We look up the phrase table to get phrase pair probability $\phi(\bar{f}_i, \bar{e}_i)$ for all $\bar{e}_i$. To speed up decoding we will limit ourselves to the top ten $\bar{e}_i$ sorted by value of $\phi(\bar{f}_i, \bar{e}_i)$. The most likely $\mathbf{e}$ is computed using the following method:

$$\begin{aligned}
\mathbf{e}_{\text{best}} &= \operatorname*{argmax}_{\mathbf{e}} \Pr(\mathbf{e} \mid \mathbf{f}) \\
&= \operatorname*{argmax}_{\mathbf{e}=\bar{e}_1,\ldots,\bar{e}_I} \prod_{i=1}^{I} \left(\phi(\bar{f}_i \mid \bar{e}_i) \times d(\text{start}_i - \text{end}_{i-1} - 1)\right) \times \prod_{i=1}^{I} p_{\text{LM}}(e_i \mid e_1, \ldots, e_{i-1})
\end{aligned}$$

You do not need to add exponents $\lambda_\phi, \lambda_d, \lambda_{\text{LM}}$ for the three components of the model above for this homework.

Given a suitable phrase table implement the pseudo-code for stack decoding given in Figure 6.6 in the Koehn SMT book. The decoder will run faster if you assume a distortion limit of 5-10 but this is optional. A small phrase table for testing your decoder is provided to you.

Provide the decoder source code along with instructions to run on the sample test set (provided to you). If you use any parameters in your decoder (e.g. α) then include values for those parameters so that your decoder is ready to run on input sentences.

Previously trained language models are also provided to you along with an API (in `kenlm_wrapper`) that will let you query the language model from Python.