CS378 Assignment 5: Alignment and Machine Translation

Academic Honesty: Please see the course syllabus for information about collaboration in this course. While you may discuss the assignment with other students, **all code you write and your writeup must be your own!**

Goals There are two goals for this assignment. First, you will study what IBM Model 1 does in a Spanish-English alignment setting to better understand what correspondences it learns between these two languages. Second, you will look at outputs from modern machine translation systems to understand how different decoding strategies work.

Dataset and Code

Code For Part 2, you will need Hugging Face Transformers,[1] which you will also use in the final project. Hugging Face is an open-source library implementing all of the standard Transformer variants for NLP (encoders, decoders, seq2seq models, etc.) and offers a hub from which to retrieve datasets and models; you will find that downloading and deploying translation models in this project is quite easy once it is installed.

You should be able to install it with `pip install transformers` in your environment of choice. You will want to have the latest installation of transformers and all its dependencies, as the library can be buggy and more recent versions are likely to be better.

This is installed on CS lab machines, **but you will need to run it with `python3.10`.** (according to instructions from helpreq).

Part 1: Word Alignment (30 points)

In this part, you will experiment with the IBM Model 1 alignment model as discussed in lecture. Recall that these models take a bitext sentence as input: a pair of sentences $\mathbf{w}^{\text{en}}$ (length n) and $\mathbf{w}^{\text{es}}$ (length m) that are translations of each other. For aligning English to Spanish, we augment $\mathbf{w}^{\text{es}}$ to additionally have a special `NULL` token at the end, making this sequence of length $m + 1$.

The dataset for this part is the Europarl dataset.[2] This is a large dataset of professionally translated sentences from the proceedings of the European Parliament. You will be focusing on English to Spanish word alignment (that is, aligning each target English word to one or more source Spanish words that are associated with it) and Spanish to English translation.

The alignment models discussed in lecture (Model 1 and the HMM) have the form:

$$P(\mathbf{w}^{\text{en}}, \mathbf{a} \mid \mathbf{w}^{\text{es}}) = P(\mathbf{a}) \prod_{i=1}^{n} P(w_i^{\text{en}} \mid w_{a_i}^{\text{es}})$$

where $w_{a_i}^{\text{es}}$ is the a_ith token in the Spanish sentence. a_i controls what Spanish word the English word "decides" to condition on to be produced. When aligning English to Spanish, we learn a matrix of word translation probabilities $P(w^{\text{en}} \mid w^{\text{es}})$, the probability of producing an English word given a particular Spanish word. In this project, we use IBM Model 1, which sets $P(\mathbf{a}) = \prod_{i=1}^{n} \frac{1}{m+1}$, a constant uniform distribution, meaning the only model parameters are the translation probabilities.

[1] `https://huggingface.co/docs/transformers/installation`

[2] `http://statmt.org/europarl/`

Getting started The main command to run is:

```
python model1_aligner.py
```

This loads the first 10,000 lines of the data, tokenizes and indexes the data, filters to keep only Spanish sentences of length at most 15, and trains IBM Model 1 on this data using 10 iterations of the expectation maximization (EM) algorithm. This algorithm optimizes $\sum_{i=1}^{D} \log \sum_{\mathbf{a}} P(\mathbf{w}^{\text{en(i)}}, \mathbf{a} \mid \mathbf{w}^{\text{es(i)}})$, the marginal log likelihood of generating the target given the source.[3] With a trained model, we can do posterior inference: `infer_model_1` computes the posterior distributions $P(a_i \mid \mathbf{w}^{\text{en}}, \mathbf{w}^{\text{es}})$ for each i. In Model 1, the a_i can be handled independently.

Because we do not have labeled alignment data, we cannot directly evaluate the model's alignment performance. However, we can still manually look at what the model is doing. The code prints alignment posterior probabilities on the training data; note that because the data is unlabeled, we have no need of a separate train/test dataset split. By default, the code will pretty-print output to the terminal. If you add the `--make_vis` flag, the model will produce alignment figures as PDFs and save these for the first 10 examples in the dataset; `--show_plot` will additionally display these figures.

Q1 (8 points) Identify an instance where a single token in English is aligned to two non-NULL tokens in the corresponding Spanish sentence (probabilities greater than 0.3) and this alignment is **incorrect** by your judgment. (a) List the sentence pair and the two Spanish words with high probability, and (b) describe what Model 1 did here and why it could be happening. (Use Google Translate if you don't know what words mean.) **Note that an alignment could still be okay even if the translation is, strictly speaking, incorrect – use your judgment about what is going on!**

Q2 (8 points) Identify an instance where a single token in English is aligned to two tokens in Spanish and this alignment is **plausible** (seems linguistically correct). Describe what is happening here and why.

Q3 (7 points) There is a phenomenon called *garbage collection* in alignment. In these cases, many words on the target side are aligned to the same source-side word. Find an occurrence of garbage collection in the data: at least three English words aligned to the same Spanish word with probability greater than 0.5, and some of them are erroneous. **Describe the two sentences involved and what you observe.** (Note that this is asking about a different case than Q1, which was asking about one English word aligning to many Spanish words.)

Q4 (7 points) Suppose we have vocabularies consisting of Spanish words s_1 and s_2 and English words e_1 and e_2 and are aligning English to Spanish as we've been doing. Suppose that we have the following translation matrix $P(e_i \mid s_i)$:

	e_1	e_2
s_1	0.9	0.1
s_2	θ	$1-\theta$
NULL	0.5	0.5

[3] The EM algorithm is used because this is a *nonconvex* optimization problem due to the sum inside the logarithm. The problem is therefore significantly harder than the supervised parameter estimation we did for HMMs, where we just counted and normalized the data.

a) Suppose our corpus consists of a single sentence pair with source s_2 and target e_2. What value(s) of θ maximize the marginal likelihood $\sum_{\mathbf{a}} P(\mathbf{w}^{\text{en}}, \mathbf{a} \mid \mathbf{w}^{\text{es}})$ of this sentence under Model 1? If multiple values or a set of values all lead to likelihood being maximized, describe the set of values.

Note that you don't need to know the EM algorithm to do this problem. Hint: write down the marginal likelihood by explicitly summing over alignments (there aren't that many possibilities on a short example). There are two possible alignments.

b) Suppose our corpus consists of a single sentence pair with source s_1 s_2 and target e_1 e_2. What value(s) of θ maximize the marginal likelihood of this sentence under Model 1? If multiple values or a set of values all lead to likelihood being maximized, describe the set of values.

Hint: when summing over alignments, you need to consider summing over all possible $\mathbf{a} = (a_1, a_2)$ pairs. Think about how many pairs there should be.

Part 2: Neural Machine Translation (20 points)

Second, you will conduct a similar analysis on some machine translation output. Specifically, we will see (a) how Transformer-based machine translation systems perform; (b) what effects beam search and sampling have on this performance.

`translation.py` includes an interface to run translation with one of two models.

First, MarianNMT is a long-running open-source MT project offering a large number of models[4] trained on the Opus corpus.[5] The default model loaded by the code is `Helsinki-NLP/opus-mt-fr-en`, which is a French-English model. You are free to use other models and language pairs by changing the path to point to an appropriate model on Hugging Face. These models are on the order of a few hundred megabytes.

You can then translate with:

```
python translation.py --input_sent "find a sentence to put here"
```

Second, you may pass in the `--use_m2m` flag to use a different model. M2M100 is a multilingual encoder-decoder model that supports 100 language. The language for the source is identified by a control token, and the first token of the output indicates the target language to translate into, but the model for the translation is shared across all languages. Note that this model is 1.9GB, so significantly larger than the Marian models. Running inference on a few examples in these models is possible on CPU, although slow, and training would be extremely slow.

Both methods will print out the translation using beam size 1, the translation using beam size 5, five samples with basic sampling, and five samples from nucleus sampling with the cutoff probability $p = 0.9$.

Q5 (10 points) Choose a language pair that you can make sense of and a translation direction where you feel comfortable reading the output (English is a good choice).

Find 5 example sentences and run them on your model of choice. Be sure to include at least 1 short sentence and at least 2 long sentences (at least 30 words). A good source to find long sentences is in news stories written in the source language.

(a) Include the 5 example sentences and their greedy translations (beam size 1) in your writeup.

(b) For each sentence, briefly discuss how the model performed. Do you see any errors in the translation?

[4] `https://huggingface.co/Helsinki-NLP`

[5] `https://opus.nlpl.eu/`

Q6 (10 points) In this part, we will look at how decoding strategies affect the behavior of the neural machine translation model. You can increase the beam size by changing the `num_beams` keyword argument to `model.generate`.

Pick **two** of your translations where there are differences in the outputs from the different strategies.

(a) List the translation and either the beam size 5 or sampled output that differs.

(b) Discuss the changes to each translation. Which one do you think is better?

Deliverables and Submission

You only need to submit a writeup for this assignment. Please submit on Gradescope.