Homework 2: POS Tagging

Consultation:

I consulted with ChatGPT in terms of understanding the provided Perl script and understanding the process of training a Huggingface flair model.

Task 1 - Part 1

Below is my model's performance. Its tagging output is the same as the Perl script *viterbi.pl*.

Bigram HMM and English:

```
[(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python viterbi.py my.hmm < data/ptb.22.txt > my_bigram.out
[(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/ptb.22.tgs my_bigram.out
 error rate by word:       0.05409178153899843  (2170 errors out of 40117)
 error rate by sentence:   0.6558823529411765  (1115 errors out of 1700)
```

Task 1 - Part 2

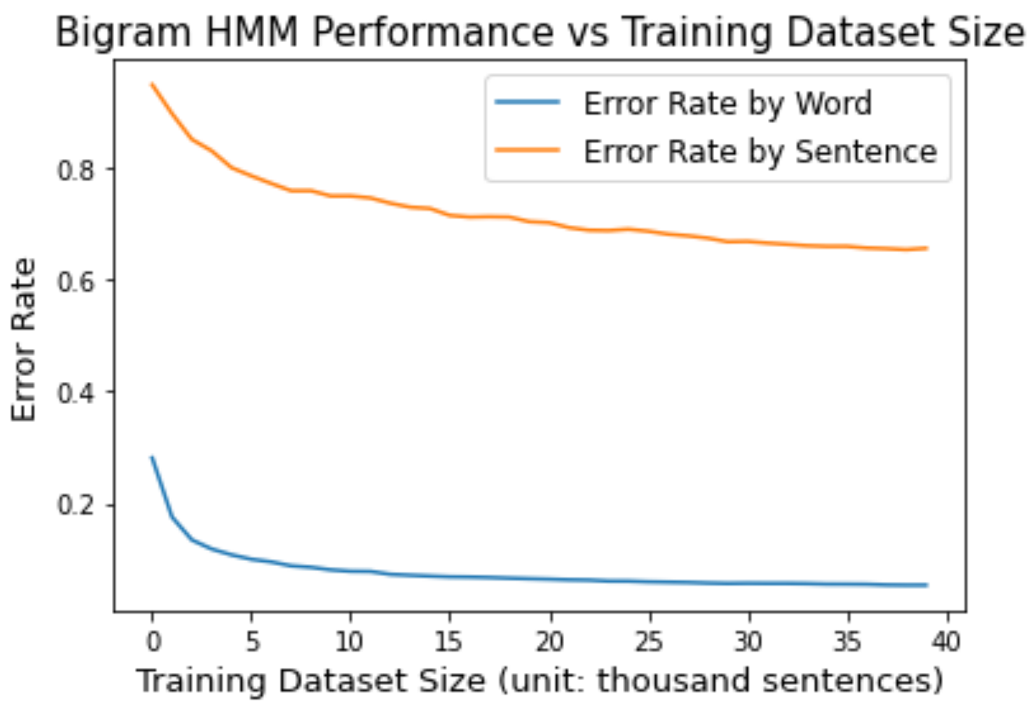
Below is my model's performance on the development data (*ptb.22.txt*).

Trigram HMM and English:

```
[(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python train_trigram_hmm.py data/ptb.2-21.tgs data/ptb.2-21.txt > my_trigram.hmm
[(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python trigram_viterbi.py my_trigram.hmm < data/ptb.22.txt > my_trigram_22.out
[(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/ptb.22.tgs my_trigram_22.out
 error rate by word:       0.04751103023655807  (1906 errors out of 40117)
 error rate by sentence:   0.6170588235294118  (1049 errors out of 1700)
```

Task 2

Based on the learning curve below, the more POS-tagged training data is fed into the bigram HMM, the lower the error rates by word and by sentence will be. However, the change in accuracy slows down when the amount of data provided reaches a certain level. For instance, in this set of provided data, when the dataset size reaches 35,000 lines, the decreases in error rates are not as drastic as when the size is 2,000. Therefore, when implementing a POS-tagging system, one has to make a tradeoff between getting higher, but almost negligible change in accuracy and saving on training time and cost to acquire more tagged data.



Task 3

1)

Both of bigram and add-1 smoothing trigram HMMs work the best on English; the models' error rates are the lowest on English among the three languages. And both models perform the worst on Bulgarian.

For both of English and Bulgarian, trigram HMM improves accuracy. But for Japanese, trigram worsens accuracy.

Bigram HMM and English:

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python train_hmm.py data/ptb.2-21.tgs data/ptb.2-21.txt > my.hmm
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python viterbi.py my.hmm < data/ptb.22.txt > my_bigram.out
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/ptb.22.tgs my_bigram.out
error rate by word:       0.05409178153899843  (2170 errors out of 40117)
error rate by sentence:   0.6558823529411765  (1115 errors out of 1700)
```

Trigram HMM and English:

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python train_trigram_hmm.py data/ptb.2-21.tgs data/ptb.2-21.txt > my_trigram.hmm
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python trigram_viterbi.py my_trigram.hmm < data/ptb.22.txt > my_trigram_22.out
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/ptb.22.tgs my_trigram_22.out
error rate by word:       0.04751103023655807  (1906 errors out of 40117)
error rate by sentence:   0.6170588235294118  (1049 errors out of 1700)
```

Bigram HMM and Japanese:

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python train_hmm.py data/jv.train.tgs data/jv.train.txt > jv.hmm
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python viterbi.py jv.hmm < data/jv.test.txt > jv.out
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/jv.test.tgs jv.out
error rate by word:       0.06286114515846612  (359 errors out of 5711)
error rate by sentence:   0.1368124118476728  (97 errors out of 709)
```

Trigram HMM and Japanese:

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python train_trigram_hmm.py data/jv.train.tgs data/jv.train.txt > jv_tri.hmm
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python trigram_viterbi.py jv_tri.hmm < data/jv.test.txt > jv_tri.out
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/jv.test.tgs jv_tri.out
error rate by word:       0.06478725267028541  (370 errors out of 5711)
error rate by sentence:   0.15091678420310295  (107 errors out of 709)
```

Bigram HMM and Bulgarian:

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python train_hmm.py data/btb.train.tgs data/btb.train.txt > btb.hmm
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python viterbi.py btb.hmm < data/btb.test.txt > btb.out
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/btb.test.tgs btb.out
error rate by word:       0.11594202898550725  (688 errors out of 5934)
error rate by sentence:   0.7512562814070352  (299 errors out of 398)
```

Trigram HMM and Bulgarian

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python train_trigram_hmm.py data/btb.train.tgs data/btb.train.txt > btb_tri.hmm
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python trigram_viterbi.py btb_tri.hmm < data/btb.test.txt > btb_tri.out
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/btb.test.tgs btb_tri.out
error rate by word:       0.10195483653522076  (605 errors out of 5934)
error rate by sentence:   0.6733668341708543  (268 errors out of 398)
```

2)

One factor that accounts for the performance variance is the difference in the size of training datasets. The figure below shows that Bulgarian training dataset contains the least amount of data and English dataset contains the largest number of lines.

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ wc -l data/btb.train.txt data/jv.train.txt data/ptb.2-21.txt
   12823 data/btb.train.txt
   17044 data/jv.train.txt
   39832 data/ptb.2-21.txt
   69699 total
```

3)

For both of English and Bulgarian, trigram HMM is a relatively better model. But for Japanese, trigram is relatively worse. This difference is likely due to the high level of granularity in the Japanese tagset. The ratio of the number of trigrams to the number of unique vocabularies is 13074:3232 in the Japanese training set, 11542:32440 in the Bulgarian training set, and 16225:44390 in the English training set. In other words, there are more trigram transition probabilities than there are unique words in the Japanese training datasets. The increase in details in trigram HMM might cause the model to perform worse, especially when there is a high number of homographs in the Japanese language.

Japanese:

Name	Type	Size	Value
A	dict	2185	{('init', 'init'):{'NAMEper':-3.3887207944580484, 'VN':-3.639047080366 ...
B	dict	81	{'NAMEper':{'kasahara':-6.081762185088762, 'arisa':-7.285734989414698, ...
f	TextIOWrapper	1	TextIOWrapper object of _io module
hmmfile	str	10	jv_tri.hmm
line	str	3	81
line_split	list	5	['trans', 'P', 'ADVdgr', 'Ntmp', '0.013157894736842105']
matrix	str	4	emit
p	str	21	0.0001547029702970297
prev_state	tuple	2	('P', 'ADVdgr')
state	str	4	VAUX
states	dict	13074	{('init', 'init', 'NAMEper'):1, ('init', 'NAMEper', 'NAMEper'):1, ('NA ...
states_i	dict	2185	{('init', 'init'):56, ('init', 'NAMEper'):5, ('NAMEper', 'NAMEper'):5, ...
states_j	dict	81	{'NAMEper':1, 'PVfin':1, '.':1, 'PNsf':1, 'CD':1, '--':1, 'VN':1, 'VSf ...
textfile	str	16	data/jv.test.txt
tri_state	tuple	3	('P', 'ADVdgr', 'Ntmp')
vocab	dict	3232	{'kasahara':1, 'arisa':1, 'desu':1, '.':1, 'kadowaki':1, 'masakazu':1, ...
word	str	3	OOV

Bulgarian

Name	Type	Size	Value
A	dict	1377	{('init', 'init'):{'Nc':-1.713091545840292, 'An':-4.369525507291367, ' ...
B	dict	55	{'Nc':{'Глава':-8.891690371598097, 'СЪБРАНИЕ':-10.501128284032198, 'съ ...
f	TextIOWrapper	1	TextIOWrapper object of _io module
hmmfile	str	11	btb_tri.hmm
line	str	3	55
line_split	list	5	['trans', 'Vxi', 'Pn', 'An', '0.03389830508474576']
matrix	str	4	emit
p	str	22	1.5413307849997687e-05
prev_state	tuple	2	('Vxi', 'Pn')
state	str	1	P
states	dict	11542	{('init', 'init', 'Nc'):1, ('init', 'Nc', 'Mo'):1, ('Nc', 'Mo', 'final ...
states_i	dict	1377	{('init', 'init'):48, ('init', 'Nc'):34, ('Nc', 'Mo'):10, ('init', 'An ...
states_j	dict	55	{'Nc':1, 'Mo':1, 'An':1, 'Vpi':1, 'Af':1, 'Cp':1, 'Am':1, 'Punct':1, ' ...
textfile	str	17	data/btb.test.txt
tri_state	tuple	3	('Vxi', 'Pn', 'An')
vocab	dict	32440	{'Глава':1, 'трета':1, 'НАРОДНО':1, 'СЪБРАНИЕ':1, 'Народното':1, 'събр ...
word	str	3	OOV

English

Name	Type	Size	Value
A	dict	1398	{('init', 'init'):{'IN':-2.066238549898792, 'NNP':-1.6017689141541263, ...
B	dict	46	{'IN':{'In':-4.410862333160494, 'of':-1.8301743117058338, 'at':-3.4895 ...
f	TextIOWrapper	1	TextIOWrapper object of _io module
hmmfile	str	14	my_trigram.hmm
line	str	3	46
line_split	list	5	['trans', 'NN', 'WRB', 'CD', '0.005479452054794521']
matrix	str	4	emit
p	str	22	1.1242776516088413e-05
prev_state	tuple	2	('NN', 'WRB')
state	str	3	WP$
states	dict	16225	{('init', 'init', 'IN'):1, ('init', 'IN', 'DT'):1, ('IN', 'DT', 'NNP') ...
states_i	dict	1398	{('init', 'init'):41, ('init', 'IN'):32, ('IN', 'DT'):38, ('DT', 'NNP' ...
states_j	dict	46	{'IN':1, 'RB':1, 'NNP':1, 'RBR':1, 'DT':1, ',':1, 'NN':1, 'CD':1, 'VB' ...
textfile	str	15	data/ptb.22.txt
tri_state	tuple	3	('NN', 'WRB', 'CD')
vocab	dict	44390	{'In':1, 'an':1, 'Oct.':1, '19':1, 'review':1, 'of':1, '``': 1, 'The':1 ...
word	str	3	OOV

Bonus Task 1

The neural-network-based model I used is Hugging Face Flair (https://flairnlp.github.io/docs/tutorial-training/how-to-load-custom-dataset). For each language, Flair was trained using the language's training datasets, 1 epoch, and a mini batch size of 8. Due to limitation in RAM, I was able to use only 14,000 sentences in the training of the English model and Japanese model. I used the full sets of training data for Bulgarian.

The three models' performances are as below. They are much worse than the results of the HMMs. I believe the reasons for the difference in the performance between English Flair and HMM is due to 1) much smaller training data was fed to Flair, and 2) hyperparameters were not fine-tuned to optimize the model. As for Japanese and Bulgarian, the lack of accuracy in the Flair models is that Flair is not pre-trained on these two languages and therefore is not pre-exposed to the linguistic characteristics of the two languages.

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/ptb.22.tgs bonus1.out
error rate by word:       0.25485455043996313  (10224 errors out of 40117)
error rate by sentence:   0.7023529411764706  (1194 errors out of 1700)
```

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/jv.test.tgs jvbonus1.out
error rate by word:       0.8748030117317458  (4996 errors out of 5711)
error rate by sentence:   0.997179125528914  (707 errors out of 709)
```

```
(base) MacBook-Pro-4:COMS4705_hw2 kaimihuang$ python tag_acc.py data/btb.test.tgs btbbonus1.out
error rate by word:       1.0  (5934 errors out of 5934)
error rate by sentence:   1.0  (398 errors out of 398)
```

Bonus Task 2

1)
In the prompt, I would give ChatGPT a few examples (few-shot learning) such as three pairs of sentences and their corresponding tags and ask ChatGPT to provide a tag sequence for a new sentence.

2)
The potential advantages include saving time on acquiring training datasets and on writing hmm-training and Viterbi algorithms. ChatGPT is also easy to use; both of users with and without coding background can get POS tags for a sentence quickly. The potential disadvantage is because ChatGPT is a pretrained model, it may not be as accurate and optimized for the POS task compared to traditional POS tagging methods.

3)
Below is an example of a prompt given to ChatGPT. The expected response is a tag sequence of at least the same length as the input sentence if not with high accuracy.

However, the output tag sequence is longer than the sentence. This means the few-shot learning method does not work well in the POS-tagging task.

K

Example 1:
input: "Influential members of the House Ways and Means Committee introduced legislation that would restrict how the new savings-and-loan bailout agency can raise capital , creating another potential obstacle to the government 's sale of sick thrifts ."
output: "JJ NNS IN DT NNP NNP CC NNP NNP VBD NN WDT MD VB WRB DT JJ NN NN NN MD VB NN , VBG DT JJ NN TO DT NN POS NN IN JJ NNS ."

Example 2:
input: "The bill , whose backers include Chairman Dan Rostenkowski (D. , Ill.) , would prevent the Resolution Trust Corp. from raising temporary working capital by having an RTC-owned bank or thrift issue debt that would n't be counted on the federal budget ."
output: "DT NN , WP$ NNS VBP NNP NNP NNP -LRB- NNP , NNP -RRB- , MD VB DT NNP NNP NNP IN VBG JJ VBG NN IN VBG DT JJ NN CC NN NN NN WDT MD RB VB VBN IN DT JJ NN ."

Example 3:
input: " ` ` Such agency ` self-help ' borrowing is unauthorized and expensive , far more expensive than direct Treasury borrowing , " said Rep. Fortney Stark (D. , Calif.) , the bill 's chief sponsor . "
output: " ` ` JJ NN ` ` NN " NN VBZ JJ CC JJ , RB RBR JJ IN JJ NNP NN , " VBD NNP NNP NNP -LRB- NNP , NNP -RRB- , DT NN POS JJ NN ."

Task:
input: "ChatGPT is primarily designed for generating human-like text based on the given prompts."
prompt: "Generate a POS tag sequence for the task input."

Here is the part-of-speech (POS) tag sequence for the input sentence: "ChatGPT is primarily designed for generating human-like text based on the given prompts."

"NNP VBZ RB JJ VBN IN VBG JJ NN VBN IN VBG JJ NN VBN IN VBG JJ NN VBN NN VBN IN VBG JJ NNS ."

Please note that this tag sequence is a simplified representation of POS tagging based on common English POS tags. Actual POS tagging may vary depending on the tag set and language used.

Regenerate