LING572 Hw6: Beam search
Due: 11pm on Feb 16, 2021

The example files are under /dropbox/20-21/572/hw6/examples/.

Q1 (75 points): Write a script, **beamsearch_maxent.sh**, that implements the beam search for POS tagging.

- The format is: beamsearch_maxent.sh test_data boundary_file model_file sys_output beam_size topN topK
- test_data has the following format (e.g., **ex/test.txt**): "instanceName goldClass f1 v1 f2 v2 ...", where an instance corresponds to a word and goldClass is the word's POS tag according to the gold standard. Note this format is slightly different from the format used in the previous assignments, which is "goldClass f1:v1 f2:v2 ...".
- boundary_file: the format of boundary_file is one number per line, which is the length of a sentence (e.g., **ex/boundary.txt**); for instance, if the first line is 46, it means the first sentence in test_data has 46 words.
- model_file is a MaxEnt model in text format (e.g., **m1.txt**).
- sys_output (e.g., **ex/sys**) has the following format: "instanceName goldClass sysClass prob", where *instanceName* and *goldClass* are copied from the test_data, *sysClass* is the tag y for the word x according to the best tag sequence found by the beam search, and *prob* is $P(y \mid x)$. Note *prob* is NOT the probability of the whole tag sequence given the word sentence. It is the probability of the tag y given the word x.
- topN: When expanding a node in the beam search tree, choose only the *topN* POS tags for the given word based on $P(y \mid x)$.
- beam_size is the max gap between the lg-prob of the best path and the lg-prob of kept path: that is, a kept path should satisfy $lg(prob) + beam\_size \geq lg(max\_prob)$, where max_prob is the prob of the best path for the current position. lg is base-10 log.
- topK is the max number of paths kept alive at each position after pruning.

Note:

- A *path* in the beam search is the path from the root to a node in the beam search tree. And for more info about how beam search works and the meaning of beam_size, topN and topK, see the hw6 slides.
- Remember that the feature vectors in the test_data do not include features t_{i-1}=tag_{i-1} (e.g., **prevT=NN**) and t_{i-2} t_{i-1}=$tag_{i-2} + tag_{i-1}$ (e.g., **prevTwoTags=JJ+NN**), because the tags of the previous words are not available for the test data before the decoding starts. **You need to add those features to the feature vectors before calling the model to classify the current instance based on the current path.**

- Ex1: suppose the current instance is "instanceName goldTag f1 v1 f2 v2 ...", and in the current path the system tags the previous word as NN and the word before the previous word as JJ. You need to add "prevT=NN 1" and "prevTwoTags=JJ+NN 1" to the feature vector in order to determine the top tags of the current instance according to the current path.
- Ex2: If the current word is the first word in the sentence, you need to add these feature pairs: "prevT=BOS 1" and "prevTwoTags=BOS+BOS 1".
- Ex3: If the current word is the 2nd word in the sentence and the POS tag of the previous word in the current path is X, you need to add these pairs: "prevT=X 1" and "prevTwoTags=BOS+X 1".
- Ex4: Nothing special needed to be done for last two words in the sentence, as the added features look at previous tags, which are available when you are dealing with any word after the second words in the sentence.
- When you add these two types of features, only add the ones that appear in the model file. If a feature (e.g., prevTwoTags=NN+RB) does not appear in the model file, that means that the tag bigram does not appear in the training data. In that case, you do not need to add the feature to the feature vector, as the model does not contain the weights for the corresponding feature functions. Another way to look at this is that if a (feature, class) pair does not appear in the model file, it means the weight of the feature function is zero.
- For your convenience, the list of these two types of features in the **m1.txt** is stored in **feats_to_add**. Your code should NOT read in a file like **feats_to_add** because this info should come from the model file. This file is there just to show you what these features look like.
- To summarize, you need to add prevT=xx and prevTwoTags=yy+xx features on the fly. If such a feature does not appear in the model file, simply ignore the feature (i.e., assuming its weight is 0).

Run beamsearch_maxent.sh with **sec19_21.txt** as the test data, **m1.txt** as model_file, **sec19_21.boundary** as the boundary file.

- Before running your code on the whole test set, you should test your code on smaller data sets. For instance, you can use ex/test.txt as the test file, ex/boundary.txt as boundary file, m1.txt as the model file. After that, you can run your code on the real data set with the (0, 1, 1) setting, and record the time it takes. The running time for other settings could be much longer. In that case, plesae use condor_submit if possible.
- Fill out Table 1.
- Submit the sys_output file for the third row in Table 1 (i.e., the row when beam_size=2, topN=5 and topK=10).

beam_size	topN	topK	Test accuracy	Running time
0	1	1		
1	3	5		
2	5	10		
3	10	100		

Table 1: Beam search results

Submission: Submit the following to Canvas:

- Your note file *readme.(txt | pdf)* that includes Table 1 and any notes that you want the TA to read.
- hw.tar.gz that includes all the files specified in dropbox/20-21/572/hw6/submit-file-list, plus any source code (and binary code) used by the shell scripts.
- Make sure that you run **check_hw6.sh** before submitting your hw.tar.gz.