Natural Language Processing

Semester 2020s

Students:

Alex Zaikin ID: 321128258

Matan Sudry ID: 203495411

Date 29/01/2020

HW1

1. Implementation details that differ from the mandatory implementation details
   a. We implemented all mandatory features, in both the final models we had 33 feature groups, final model 1 had 42,772 individual features, final model 2 had 4,548 features.
   b. We used a unique threshold for each feature group.
   c. We implemented mini-batch training with reshuffling every epoch.
   d. We implemented accuracy approximation during training.
   e. For the avoidance of doubt, model1 and model2 were only trained on the train1 and train2 datasets respectably.
   f. We implemented beam Viterbi.
   g. In order to establish a baseline for the competition datasets (comp1 and comp2), we used a 3rd party POS-tagger (https://aihaiyang.com/software/stanford-pos-tagger/single/) to tag the datasets.
   We tested the accuracy of the 3rd party POS-tagger by retagging some of the available tagged data test1 and train2, and got 98-99% accuracy.

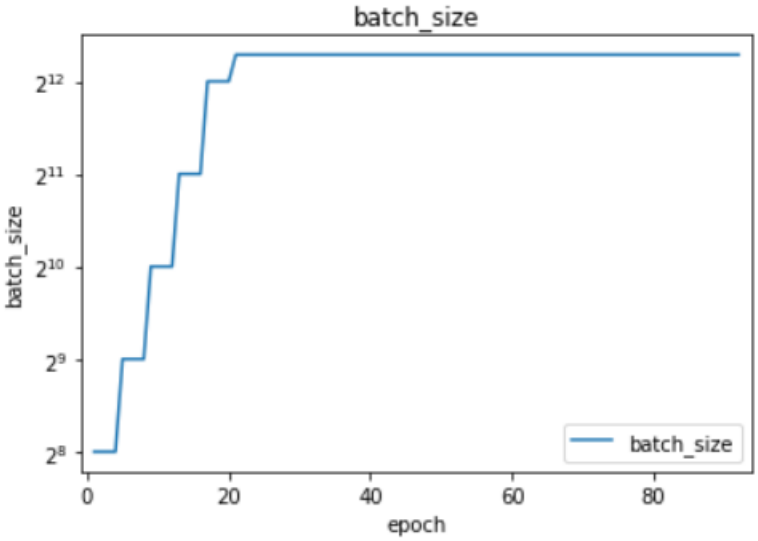
2. Worth-Noting Experimentation
   a. We tried different lambda parameters for the L2 regularization (0, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100), but we found all non-zero lambdas to yield significantly worse results. in addition during most of the training process the training and validation losses were “close” and seemed to go down together
   b. We experimented with a few initialization techniques: uniform 0-1, centered uniform 0-1, normalized uniform 0-1, Xavier (X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks., in: Aistats, Vol. 9, 2010, pp. 249–256) initialization and all-0 initialization. We found the uniform 0-1 initialization technique to both significantly improve the initial loss and convergence speed.
   c. We experimented with different feature groups and different thresholds for each group, the guiding intuition was to increase the threshold for features that seemed to have more noise in their relation to the tag or their interpretation. But in the end the thresholds were chosen through cross-validation.
   d. We tried weights averaging of the last 5 epochs (inspired by Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In UAI - Conference on Uncertainty in Artificial Intelligence, 2018) but we did not find it to improve our results.

e. During training we experimented with different Viterbi beam sizes (1, 2, 3, 5, 10) however we did not find the non-greedy (beam>1) variants to be consistently better than the greedy (beam=1) variant, in some cases the models yielded +0.0001 accuracy for beam=2, but in other cases we have seen as much as -0.0004 accuracy for beam=5, so we only used beam > 1 during the final predictions of Model1 and Model2.

3. Training process and Model selection
   a. Batch size

      For Model2 we used the whole dataset (250 samples) for the whole training process.

      For Model1 we used 256 mini-batch training for the early epochs which significantly reduced early epoch time and doubled it every 5 epochs until maxing out at 5000 batch-size (inspired by Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. arXiv preprint arXiv:1711.00489 [cs.LG], 2017), batch_size over-time illustration:



   b. We used the tagged comp1 and comp2 datasets as validation sets and maximized the accuracy on each one (a technique that maximizes the expected competition score and not overall model generalization). We evaluated our model loss and accuracy (for Model1 using a constant sample of 50 sentences from each dataset and beam size of 1, for Model2 we evaluated the whole dataset each epoch due to the shorter evaluation time) with the comp1_tagged and comp2_tagged sets respectably every epoch.
   c. We trained both Model1 and Model2 for 150 epochs, while saving a checkpoint each epoch.
   d. In the end we sampled the models of the best epochs and evaluated the models on the whole comp1 and comp2 datasets using beam size of 1 to choose the best models.
   e. Finally, we evaluated the best model1 (epoch 92) and model2 (epoch 43) using a beam sizes of 1,5,10, 15, 20 and kept the best results as the final results.
   f. Pure Train time (Excluding: evaluation, failed experiments and trimmed epochs) for the Final Model1 was 16:23 hours and Final Model2 was 8:32 minutes (all training done on the Azure servers).

4. Conclusions

a. The main challenge, that made the task different then other ML/DL tasks, was our inability to efficiently evaluate our models due to both the extremely long inference time, and the uncertainty of the return on investing more time on a larger beam size.

b. Our best and final results of Model1 on the test1 dataset are 0.9532 accuracy with beam=1 (tagging duration 33.22 minutes). As mentioned earlier, we used test1 as a test set and only evaluated the model once on it (since we were only required to pass the 0.9 accuracy and additional improvement on test1 were not a priority).

```
[42]: 1 tic = time.time()
      2 pred_tags, true_tags = best_model.predict(test1_dataset.sentences, beam=1)
      3 test1_score = best_model.score_func(pred_tags, true_tags)
      4 print(test1_score)
      5 print('time:', time.time() - tic)

      0.9531976007434316
      time: 33.222426414489746
```

c. Individual Tags accuracy (1 vs. all):

Accuracy	Tag
1.0000	LS
1.0000	:
1.0000	.
1.0000	``
1.0000	-RRB-
1.0000	#
1.0000	$
1.0000	WRB
1.0000	-LRB-
1.0000	,
1.0000	EX
1.0000	TO
1.0000	WP$
1.0000	SYM
1.0000	WP

Accuracy	Tag
0.9988	RP
0.9989	DT
0.9992	CD
0.9993	JJS
0.9995	WDT
0.9996	RBS
0.9998	PDT
0.9998	POS
0.9998	UH
0.9998	PRP
0.9998	FW
0.9998	CC
0.9999	MD
0.9999	''
0.9999	PRP$

Accuracy	Tag
0.9813	NN
0.9839	JJ
0.9929	NNP
0.9937	RB
0.9942	VBN
0.9947	IN
0.9947	NNS
0.9957	VBD
0.9967	VB
0.9968	NNPS
0.9970	VBP
0.9970	VBG
0.9974	JJR
0.9981	VBZ
0.9983	RBR

d. 10 worst tags confusion matrix:

	NN	JJ	NNP	RB	VBN	IN	NNS	VBD	VB	NNPS
NN	3137	101	24	5	0	1	39	4	15	0
JJ	78	1323	23	16	29	0	3	7	4	0
NNP	25	25	1903	1	0	0	9	1	0	18
RB	11	21	0	721	0	15	0	0	2	0
VBN	6	29	0	0	431	0	0	32	0	0
IN	4	0	2	54	1	2477	0	0	0	0
NNS	2	1	3	0	0	0	1433	0	0	2
VBD	5	2	0	1	37	0	0	781	1	0
VB	18	6	1	0	0	1	0	1	544	0
NNPS	0	4	28	0	0	0	22	0	0	29

i. We could (perhaps) improve the result by implementing a boosting variant of MEMM to dynamically increase the weight of samples (in the loss function) the model did not classify correctly

ii. Or by increasing the weights of tags (in the loss function) the model did not classify correctly.

iii. Or by adding more features related to the wrongly classified tags.

e. Our best and final results of Model1 on the comp1 dataset are 0.9275 accuracy (against the 3rd party tags) with beam=5 (tagging duration 2:56 hours), and considering the expected error of the tags, we would expect our accuracy against the true tags to be between 0.925-0.93.

f. Our best and final results of Model2 on the comp2 dataset are 0.927 accuracy (against the 3rd party tags) with beam=1 (tagging duration 19:23 minutes), and considering the expected error of the tags, we would expect our accuracy against the true tags to be between 0.925-0.929.

g. In general, one of the partners implemented the training stage, the other one implemented the inference tests. Training + Experiments + hyperparameter tuning was done in parallel by both of us, after which we compared results and picked the most promising courses of action.

5. Final Features for Model1 and Model2:

Feature Group	mandatory	threshold	Model1 features	Model2 features
Lowercase word at pos i, tag	F100	0	14,719	1,736
Lowercase word at pos i suffix of length 1, tag	F101	5	2,289	170
Lowercase word at pos i suffix of length 2, tag	F101	5	1,757	165
Lowercase word at pos i suffix of length 3, tag	F101	5	955	145
Lowercase word at pos i suffix of length 4, tag	F101	5	248	96
Lowercase word at pos i prefix of length 1, tag	F102	5	2,540	158
Lowercase word at pos i prefix of length 2, tag	F102	5	2,431	170
Lowercase word at pos i prefix of length 3, tag	F102	5	1,432	200
Lowercase word at pos i prefix of length 4, tag	F102	5	392	133
Tag at pos i-2, Tag at pos i-1, Tag at pos i	F103	1	5,192	598
Tag at pos i-1, Tag at pos i	F104	1	908	216
Tag at pos i	F105	1	44	31
IND word at pos i is lowercase, tag	has_uppercase	1	75	43
IND word at pos i contains digits, tag	has_digits	1	49	35
Lowercase word at pos i-1, tag		20	616	28
Lowercase word at pos i+1, tag		20	614	23
Lowercase word at pos i+1 prefix of length 3, tag		20	869	29
Lowercase word at pos i-1 prefix of length 3, tag		20	938	35
Lowercase word at pos i+1 prefix of length 2, tag		20	1,081	44
Lowercase word at pos i-1 prefix of length 2, tag		20	1,108	42
Lowercase word at pos i+1 suffix of length 3, tag		20	962	30
Lowercase word at pos i-1 suffix of length 3, tag		20	971	41
Lowercase word at pos i+1 suffix of length 2, tag		20	975	49
Lowercase word at pos i-1 suffix of length 2, tag		20	996	61

IND word at pos i is alpha-numeric, tag		10	52	24
IND word at pos i is alphabetic, tag		10	52	24
IND word at pos i is ascii, tag		10	42	21
IND word at pos i is decimal, tag		10	43	22
IND word at pos i is digits, tag		10	43	22
IND word at pos i is numeric, tag		10	43	22
IND first char of word at pos i is uppercase and all other chars are lowercase, tag		10	64	27
IND word at pos i is uppercase, tag		10	49	22
word at pos i length, tag		10	223	86