CS 626 - Assignment 1, PoS Tagging

Manoj Bhadu, Tushar Dhawal Baranwal, Naman Verma

14th September 2020

1 Introduction

PoS tagging - Part-Of-Speech tagging is the process of attaching one of the parts of speeches to the given word. It is an important starting step (before chunking) for many high level complex NLP tasks like sentiment analysis, machine translation and so on. PoS tagging can be formulated as a classification problem where the labels are the PoS tags and the features are words themselves. In this assignment, we use probabilistic(HMM), classical machine learning(SVM) and advanced machine learning(LSTMs) approaches for this classification problem. We observe that because the formulation of PoS tagging as a HMM is fairly natural, we achieve a good enough accuracy (95 %), but the decoding speed is relatively slow. To use SVMs, we need to generate a good feature set for the algorithm to work well. Adding language specific morphological features or using more sophisticated word embeddings can improve the results to a large degree, but the results(72 %) are still not as good as HMMs or Bi-LSTMs. Bi-LSTMs is the most modern approach, and is end-to-end. We find that this approach gives the best accuracy (96 %).

2 PoS Tagging Using Hidden Markov Models

Average Accuracy : 95 %
Overall average precision : 0.92
Overall average recall : 0.89
Overall average F1 score : 0.90

2.1 Approach

An HMM based PoS tagger using the approach taught in class (Viterbi decoder). A detail that has been arbitrarily assumed is that when the lexical probability is zero, the log of that would be -*inf*. To resolve that, it is set to -500. This is a kind of smoothing where instead of setting the probability to be zero, we assume that every word can be emitted by every tag with a very minuscule probability (10^{-500}) . We also show the heat-map for the Transition Probability Matrix.

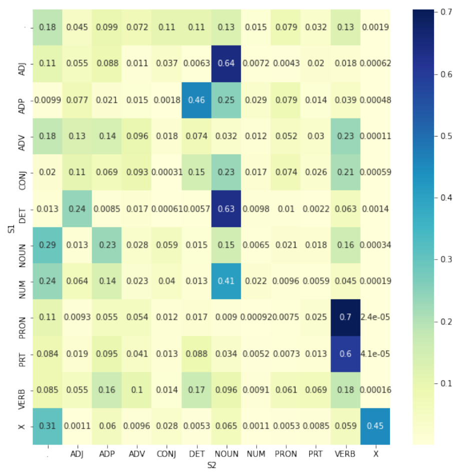

Figure 1: Transtion Probability Matrix; Previous tag : S1, Next tag : S2

2.2 Error Description

	accuracy	count	tag
0	0.99	136206.0	DET
1	0.91	273974.0	NOUN
2	0.95	181780.0	VERB
3	0.97	143873.0	ADP
4	0.89	83132.0	ADJ
5	1.00	146764.0	.
6	0.99	37873.0	CONJ
7	0.98	49050.0	PRON
8	0.90	29706.0	PRT
9	0.89	55878.0	ADV
10	0.89	14824.0	NUM
11	0.34	1376.0	X

Table 1: Per PoS Accuracy

actual	.	ADJ	ADP	ADV	CONJ	DET	NOUN	NUM	PRON	PRT	VERB	X
pred												
.	146734	277	25	124	3	0	4351	171	4	11	475	108
ADJ	0	74127	74	2204	0	1	4956	78	2	39	826	68
ADP	1	250	139002	1739	15	646	1763	53	413	2310	1199	62
ADV	0	2423	2060	49747	115	88	203	0	3	234	132	5
CONJ	0	0	146	107	37661	26	1	0	0	0	0	4
DET	1	2017	176	280	75	134412	4237	397	301	23	748	122
NOUN	2	3099	57	593	2	6	250275	852	41	268	6439	461
NUM	0	4	1	0	0	2	413	13177	0	0	0	5
PRON	0	62	352	114	0	1018	2977	59	48269	28	102	23
PRT	0	201	1846	726	0	1	146	1	7	26743	13	5
VERB	0	664	131	244	1	4	4505	30	9	50	171841	49
X	26	8	3	0	1	2	147	6	1	0	5	464

Table 2: Confusion Matrix

	F1 score	precision	recall	tag
0	0.98	0.96	1.00	.
1	0.90	0.90	0.89	ADJ
2	0.95	0.94	0.97	ADP
3	0.90	0.90	0.89	ADV
4	0.99	0.99	0.99	CONJ
5	0.96	0.94	0.99	DET
6	0.93	0.95	0.91	NOUN
7	0.93	0.97	0.89	NUM
8	0.95	0.91	0.98	PRON
9	0.90	0.90	0.90	PRT
10	0.96	0.97	0.95	VERB
11	0.46	0.70	0.34	X

Table 3: Precision/Recall values

2.3 Error Analysis

The HMM PoS tagger performs fairly well for almost all PoS tags except X - unknown. This is because when a word in the testing data-set is not present in the training data-set, the tag which is most likely given the previous PoS tag is assigned to it. Thus, some genuinely non-dictionary words may be assigned proper tags. Hence the recall for X is low. This also slightly lowers the precision for highly likely tags. It would be interesting to see whether we get a slightly better accuracy if we tag unseen words to be X. Errors for . and DET are very low because there is less ambiguity in their tagging. Moreover, we see in a separate analysis that even when we set all transition probabilities to be equal, we get a respectable accuracy of 85 percent (analysis not presented in detail). Thus, lexical probabilities are good enough to get to a certain level of accuracy.

2.4 Learning

- PoS tagging using HMM is fairly accurate, even with a bigram assumption.
- The Viterbi algorithm implementation using dynamic programming makes the algorithm sufficiently efficient. However, this method is still the slowest because of the sequential manner in which decoding is done and may suffer when we need extremely fast decoding.
- Lexical probabilities can get us to respectable degree of accuracy, but using the transition probabilities help us in resolving ambiguities for ADJ, ADV and NOUN type tags.

3 PoS Tagging Using Support Vector Machines

Average Accuracy : 72 %
Overall average precision : 0.76
Overall average recall : 0.56
Overall average F1 score : 0.55

3.1 Approach

We first implement a general 2-class soft margin SVM, with a regularization parameter. The loss function optimizer implemented is stochastic gradient descent where we shuffle the input variables and use the first N samples for calculating the gradient. Then, we use a one-vs-all approach for multi-class classification. Initially we used simple features like length of the word, capitalization characteristics, occurrence of common 2 character sets and so on, but they returned abysmal results (40 percent accuracy). To improve the accuracy of the model, we used word2vec and converted every word into a vector of 16 entries. We observed that increasing the number of entries did not improve the results by a large amount. So, to avoid over-fitting, we settle for lower number of features. One thing to be noted here is that we did not consider the features of the next or previous word. As it turns out, this is quite significant.

3.2 Error Description

	accuracy	count	tag
0	0.09	29828.0	PRT
1	1.00	147530.0	.
2	0.04	56232.0	ADV
3	0.02	83695.0	ADJ
4	0.84	144725.0	ADP
5	0.96	136975.0	DET
6	0.96	275475.0	NOUN
7	0.47	182712.0	VERB
8	0.88	49326.0	PRON
9	0.80	38139.0	CONJ
10	0.04	14873.0	NUM
11	0.00	1386.0	X

Table 4: Per PoS Accuracy

actual_tag pred_tag	.	ADJ	ADP	ADV	CONJ	DET	NOUN	NUM	PRON	PRT	VERB	X
.	147526	137	3391	3564	1120	1131	4235	2022	26	2927	2701	112
ADJ	0	2035	0	1755	0	0	340	303	0	105	330	0
ADP	0	2284	121986	5401	3489	494	383	0	224	15119	1031	20
ADV	0	720	368	2371	0	0	25	0	0	0	0	0
CONJ	0	0	525	11	30608	0	0	0	0	0	0	4
DET	0	1862	5679	1161	12	131896	402	1750	2864	745	65	20
NOUN	4	73898	7957	27875	449	2399	263709	9341	2008	3485	92930	1212
NUM	0	0	0	0	0	0	0	598	0	0	0	0
PRON	0	56	1080	6024	0	577	563	859	43327	3865	5	6
PRT	0	0	1914	0	0	0	0	0	0	2707	0	1
VERB	0	2703	1825	8070	2461	478	5818	0	877	875	85650	11

Table 5: Confusion Matrix

	F1 score	precision	recall	tag
0	0.93	0.87	1.00	.
1	0.05	0.42	0.02	ADJ
2	0.83	0.81	0.84	ADP
3	0.08	0.68	0.04	ADV
4	0.88	0.98	0.80	CONJ
5	0.93	0.90	0.96	DET
6	0.69	0.54	0.96	NOUN
7	0.08	1.00	0.04	NUM
8	0.82	0.77	0.88	PRON
9	0.16	0.59	0.09	PRT
10	0.59	0.79	0.47	VERB

Table 6: Precision/Recall values

3.3 Error Analysis

We see that when we use one-vs-all multi-class SVM for PoS tagging using the features of just the word in question, we get very poor results in terms of accuracy for PRT, ADV, ADJ, VERB, NUM and X. We see from the heat-map shown in the previous section that these are tags which depend on the next PoS tag. Since this information is lost when we use the features only for the current word, the poor performance for these tags is explained. We also observe that the tag most frequently assigned is NOUN, and consequently the precision for NOUN is very low. This is because of the type of SVM we have used. NOUNs occupy the maximum 'space' in the word2vec vector space since they are the most frequent, and the one-vs-all SVM gives the class which minimises the loss given the feature set. Hence, *whenever in doubt*, the class returned is NOUN. For example class X is overwhelmingly classified as NOUN. This method overall performs well for punctuation, DET and in general for words which do not need much context to be tagged well.

3.4 Learning

- Simple features like word length and capitalization characteristics are not enough for even basic NLP tasks and we need morphological features or word embeddings to even reach a respectable score.
- Increasing the size of the feature space in word2vec did not better the results after a point.
- This implementation shows the elegance of the HMM paradigm and shows how important the next or previous PoS tag is, for effective classification especially for tags like ADJ, ADV, VERB and so on.
- This implementation works well for tags which can be determined without much context.
- The features for the previous and the next word can be added to improve the results. Also, a one-vs-one instead of one-vs-all SVM can be used to see if this solves the NOUN precision problem.

4 PoS Tagging Using Bi-LSTMs

Average Accuracy : 96 %
Overall average precision : 0.93
Overall average recall : 0.89
Overall average F1 score : 0.91

4.1 Approach

We implemented the POS tagger using the Bidirectional LSTM. We have used Keras library for implementation. The major difference between an HMM and a Bidirectional LSTM is that the latter is capable of reading the sequence in reverse and using the next words in the sequence for predicting tags. We have converted the words and tags into integers and added padding at the end of each sentence and tag sequence for equalizing the length. For OOV word we defined the "-UNK-" word. For tags we used one hot encoding and for words we used the Keras embedding layer and made the parameters trainable. The network architecture is as in the figure displayed below. We trained the model for 10 epochs and got more than 0.99 accuracy on training data and 0.96 on test data.

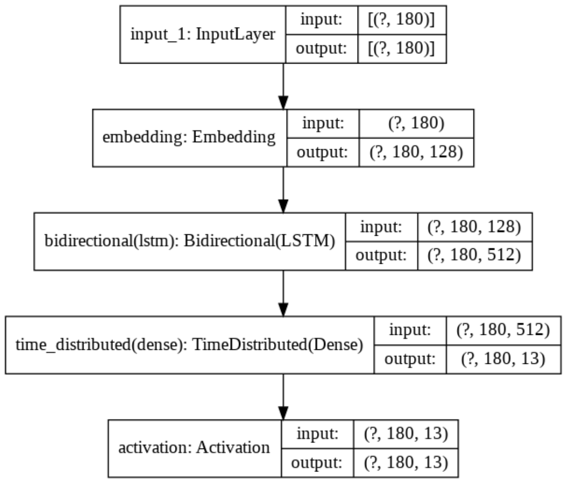

Figure 2: Bi-LSTM architecture

4.2 Error Description

	accuracy	count	tag
0	0.99	137019.0	DET
1	0.96	275558.0	NOUN
2	0.89	83721.0	ADJ
3	0.96	182750.0	VERB
4	0.98	144766.0	ADP
5	1.00	147565.0	.
6	0.91	56239.0	ADV
7	1.00	38151.0	CONJ
8	0.92	29829.0	PRT
9	0.98	49334.0	PRON
10	0.90	14874.0	NUM
11	0.21	1386.0	X

Table 7: Per PoS Accuracy

actual pred	.	ADJ	ADP	ADV	CONJ	DET	NOUN	NUM	PRON	PRT	VERB	X
.	147550	1	22	0	0	2	6	1	0	0	0	24
ADJ	3	74307	48	1801	3	7	4375	136	3	48	811	64
ADP	4	89	141872	1155	31	482	123	14	266	1775	135	29
ADV	0	1557	708	51307	46	93	225	1	3	241	139	6
CONJ	1	1	140	134	38001	34	5	0	0	0	1	4
DET	1	2	341	177	57	136104	116	0	504	5	5	19
NOUN	2	6579	92	980	10	29	264147	1172	45	296	6002	762
NUM	2	4	1	2	0	2	260	13339	0	0	0	7
PRON	0	0	90	3	0	249	27	0	48491	1	0	7
PRT	0	75	1332	346	0	1	58	1	7	27371	9	3
VERB	2	1072	119	328	3	15	5989	204	13	89	175630	167
X	0	16	1	4	0	1	126	0	2	3	16	291

Table 8: Confusion Matrix

	F1 score	precision	recall	tag
0	1.00	1.00	1.00	.
1	0.90	0.91	0.89	ADJ
2	0.98	0.97	0.98	ADP
3	0.93	0.94	0.91	ADV
4	0.99	0.99	1.00	CONJ
5	0.99	0.99	0.99	DET
6	0.95	0.94	0.96	NOUN
7	0.94	0.98	0.90	NUM
8	0.99	0.99	0.98	PRON
9	0.93	0.94	0.92	PRT
10	0.96	0.96	0.96	VERB
11	0.32	0.63	0.21	X

Table 9: Precision/Recall values

4.3 Error Analysis

The errors obtained using Bi-LSTM model are the best, and it has no visible weakness with respect to any particular tag. However, we also see that the accuracy for tags where there is more room for ambiguity is more or less unchanged - i.e. ADJ and ADV, with respect to the HMM implementation. To increase the accuracy further, we can use the word2vec or Glove pre-trained embedding. Majority of the tags are '-PAD-' tags so we have excluded them during calculating the accuracy because they are easily predictable. Use of different network architectures and training for more epochs can also improve the accuracy but may lead to over-fitting.

4.4 Learning

- Use of state of the art machine learning techniques improve the results further from classical models, but the increase in accuracy is not very large. This is again to the credit of the elegance of the formulation of the PoS tagging problem as a HMM.
- However, since these calculations can be vectorized, execution time is much lesser than an HMM. Hence, a major drawback of the HMM formulation is sequential decoding.

5 Citations

- https://www.cse.iitb.ac.in/ cs626/cs626-sem1-2012/
- https://github.com/adityajn105/SVM-From-Scratch
- https://towardsdatascience.com/svm-implementation-from-scratch-python-2db2fc52e5c2

- https://nlpforhackers.io/word-embeddings
- http://www.nltk.org/book/ch05.html
- https://keras.io/api/layers/core_layers/embedding
- https://towardsdatascience.com/pos-tagging-using-rnn-7f08a522f849
- https://www.tensorflow.org/guide/keras/masking_and_padding
- https://keras.io/api/layers/recurrent_layers/time_distributed