PART OF SPEECH TAGGING

HUC722

RAGHAV GOYAL
2010MT50612

Introduction

Part-of-speech (POS) tagging is the task of assigning part of speech tags to words in a text. It is indispensable to numerous NLP tasks such as syntactic parsers, information extraction, speech synthesis etc.
There are many ways to approach this problem but here I have implemented tagger based on unigram, bigram and Hidden Markov Models using MATLAB 2012a.

Pre-Processing

Given Brown corpus dataset which contained tagged sentences with sequences of <word/tag>, the following operations were performed,

- Converted all words to lower case, to avoid ambiguity between same words.
- Took Test Data: 1-2000 sentences [Total=2000]
- Training Data: 2001 – 57340 sentences [Total=55340]
- Created Hash table for words and tags for quick access.
- No. of different words are found to be 48398
- Total no. of tags are 30

I've also considered '*' as a tag which only occurs with word 'not' in the dataset.

There were instances such as <word/word/tag> for eg. '`1-1/2-story/N`'. I have taken '`1-1/2-story`' as a single word.

Unigram Tagger

This tagger finds the most frequent tag for every word in the training set and uses that information to assign tags to new words.
I've performed the similar approach but since, in the case of unknown words, we do not know which tag to assign. Therefore, I assigned the most frequent tag that occurred in the dataset. Tag frequency is maximum for 'N' i.e. 220256 instances. Therefore, I assigned 'N' to unknown words.

Total no. of unknown words: 1893
In which most frequent tag is 'NP' with 1078 instances. 'N' with 447 and 'ADJ' with 141 and many others.
Therefore, tagging unknown words with 'NP' will give more accuracy but since, we selected most frequent tag 'N',

Accuracy for Unigram Tagger is 88.9041%
Accuracy for Unknown words is 23.613% [477 correct out of 1893]

Confusion Matrix for Test Data

	CNJ	N	P	V	DET	VG	ADV	ADJ	WH	VN	TO	NP	VD	NUM	MOD	PRO	VBZ	*	EX	FW	UH	VB	PPO	NIL	IN	VBG	VBN	RP	JJ	AT
CNJ	1629	0	42	0	10	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	2	9070	2	145	14	42	8	141	0	7	0	35	2	1	1	2	59	0	0	1	0	0	0	0	0	0	0	0	0	0
P	76	5	4040	0	1	1	42	1	0	0	356	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	65	248	0	2771	2	0	4	11	0	3	0	2	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DET	30	8	0	0	4453	0	58	15	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VG	0	64	0	0	0	564	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADV	13	5	89	0	32	0	878	59	0	0	0	0	0	7	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0
ADJ	0	213	5	5	6	2	26	1820	0	8	0	5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
WH	46	0	0	0	0	0	15	0	405	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VN	0	15	0	16	0	0	0	0	0	744	0	0	184	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TO	0	0	0	0	0	0	0	0	0	0	548	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NP	0	1248	3	6	0	8	0	46	0	0	0	2786	4	0	20	1	0	0	0	9	0	0	0	0	0	0	0	0	0	0
VD	0	36	0	7	0	0	0	0	0	347	0	0	733	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NUM	0	108	0	0	0	0	0	0	0	0	0	0	0	995	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MOD	0	1	0	0	0	0	0	0	0	0	0	0	0	0	578	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRO	0	0	0	0	0	0	1	0	0	0	0	0	0	3	0	1446	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VBZ	0	73	0	0	0	0	0	0	0	0	0	1	0	0	0	0	163	0	0	0	0	0	0	0	0	0	0	0	0	0
*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	101	0	0	0	0	0	0	0	0	0	0	0	0
EX	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	67	0	0	0	0	0	0	0	0	0	0	0
FW	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0
UH	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
VB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PPO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NIL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VBG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VBN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 1: Confusion matrix for Unigram Tagger

There are many tags that do not occur in test data but are present in train data hence, there are ‘zero’ rows & columns in confusion matrix.
We observe that highest count for misclassification is NP for N.

Bigram Tagger

This tagger finds the most likely tag for each word based on the previous tag. When run on new data, it works through the sentence from left to right, and uses the tag that it just generated for the preceding word.
From training, I created a data structure which stored the count for every <current/previous> tag combination for each different word.
For testing, I found the most frequent tag for a word given its previous tag in test data and allot that tag. If the word is in start of the sentence, I allot tag on the basis of frequency similar to unigram.
If word is unknown then allot the tag on the basis of previous tag. If unknown word is in start of the sentence, then allot the most frequent tag i.e. N.

Accuracy for Bigram Tagger is 91.2947%
Accuracy for Unknown words is 39.883% [755 correct out of 1893]

I observed that there were total 2665 words that were differently tagged by Unigram and Bigram tagger. Out of these there were 1476 words which were incorrectly tagged by unigram tagger and correctly by bigram tagger.

Confusion Matrix for Test Data

	CNJ	N	P	V	DET	VG	ADV	ADJ	WH	VN	TO	NP	VD	NUM	MOD	PRO	VBZ	*	EX	FW	UH	VB	PPO	NIL	IN	VBG	VBN	RP	JJ	AT
CNJ	1616	1	50	7	5	1	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	8931	76	93	115	33	7	134	0	9	0	60	2	1	1	2	67	0	0	1	0	0	0	0	0	0	0	0	0	0
P	43	1	4133	0	2	1	50	0	0	0	293	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	39	98	0	2953	6	0	3	4	0	4	0	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DET	31	8	5	0	4456	0	48	16	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VG	0	43	6	1	14	560	0	5	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADV	15	3	82	0	32	0	911	42	0	0	0	1	0	2	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0
ADJ	0	157	17	14	32	1	16	1836	0	2	0	13	2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
WH	46	0	0	0	0	0	15	0	405	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VN	0	6	3	3	8	0	0	0	0	823	0	0	116	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TO	0	0	24	0	0	0	0	0	0	0	524	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NP	0	362	153	33	170	5	0	45	0	0	0	3345	4	0	7	1	1	0	0	5	0	0	0	0	0	0	0	0	0	0
VD	0	6	9	13	2	0	0	0	0	217	0	6	870	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NUM	0	34	14	1	56	0	7	0	0	0	0	3	0	988	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MOD	0	7	0	1	0	0	0	0	0	0	0	0	0	0	571	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRO	0	0	1	0	0	0	0	0	0	0	0	0	0	3	0	1446	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VBZ	0	43	10	3	2	0	0	0	0	0	0	4	0	0	0	0	175	0	0	0	0	0	0	0	0	0	0	0	0	0
*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	101	0	0	0	0	0	0	0	0	0	0	0	0
EX	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	54	0	0	0	0	0	0	0	0	0	0	0
FW	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
UH	0	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
VB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PPO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NIL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VBG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VBN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 2: Confusion matrix for Bigram Tagger

In this, we observe that NP correct classifications have been increased from 2786 to 3345. Also, the highest misclassification count is reduced form previous case.
Hence, we exploited the context-sensitive nature of language through bigram tagger.

HMM Tagger

The HMM tagger uses Hidden Markov Model (HMM) to find the most likely tag sequence for each sentence. Basically it has two assumptions:

- Probability of a word appearing is dependent only on its own part of speech tag,

$$P({w_1}^n|{t_1}^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

- Probability of a tag appearing is dependent only on the previous tag,

$$P({t_1}^n) \approx P(t_i|t_{i-1})$$

Final form of the model,

$$\hat{t}_1^n \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)$$

I have implemented this model by finding Transition Probability Matrix representing probability of moving from one tag to other, Emission Probability describing probability of each word being generated from each tag.
Initial probabilities were taken to be proportion of each tag in the training dataset.
Finally, Viterbi Search Algorithm was used to find the most likely sequence of tags.

Accuracy for HMM Tagger is 92.1784%
Accuracy for Unknown words is 35.710% [676 correct out of 1893]

Confusion Matrix for Test Data

	CNJ	N	P	V	DET	VG	ADV	ADJ	WH	VN	TO	NP	VD	NUM	MOD	PRO	VBZ	*	EX	FW	UH	VB	PPO	NIL	IN	VBG	VBN	RP	JJ	AT
CNJ	1601	1	52	6	9	1	11	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	8	8844	37	95	57	35	15	235	4	21	8	75	16	3	3	20	55	0	0	1	0	0	0	0	0	0	0	0	0	0
P	52	2	4402	0	0	0	47	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	40	76	2	2965	4	0	4	5	5	7	0	1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DET	29	10	1	1	4429	0	84	10	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VG	2	31	8	0	6	566	0	7	0	5	0	0	4	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ADV	14	2	76	2	28	0	931	35	0	0	0	0	0	4	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0
ADJ	2	58	12	6	39	1	20	1921	0	2	0	22	3	3	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WH	27	0	0	3	0	0	12	0	424	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VN	0	3	2	4	0	0	0	0	0	871	0	2	76	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TO	0	0	32	0	0	0	0	0	0	0	516	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NP	20	207	73	47	93	9	16	114	9	13	38	3273	114	0	6	83	0	0	0	16	0	0	0	0	0	0	0	0	0	0
VD	2	6	5	8	0	0	0	1	0	170	0	2	926	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
NUM	0	11	7	1	36	0	4	28	0	7	2	7	0	991	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MOD	0	0	0	1	0	0	0	0	0	0	0	0	0	0	578	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRO	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	1447	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VBZ	0	38	9	0	2	0	0	0	0	0	0	2	4	0	0	0	182	0	0	0	0	0	0	0	0	0	0	0	0	0
*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	101	0	0	0	0	0	0	0	0	0	0	0	0
EX	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	66	0	0	0	0	0	0	0	0	0	0	0
FW	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
UH	0	1	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
VB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PPO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NIL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VBG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VBN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 3: Confusion matrix for HMM Tagger

We observe that the highest misclassification count is reduced further and misclassifications for NP and N are more widely spread.

Perceptron Algorithm

The algorithm discussed in Collins et al paper[1],

Inputs: A training set of tagged sentences, $(w^i_{[1:n_i]}, t^i_{[1:n_i]})$ for $i = 1 \ldots n$. A parameter T specifying number of iterations over the training set. A "local representation" ϕ which is a function that maps history/tag pairs to d-dimensional feature vectors. The global representation Φ is defined through ϕ as in Eq. 1.
Initialization: Set parameter vector $\bar{\alpha} = 0$.
Algorithm:

For $t = 1 \ldots T, i = 1 \ldots n$

- Use the Viterbi algorithm to find the output of the model on the i'th training sentence with the current parameter settings, i.e.,

$$z_{[1:n_i]} = \arg\max_{u_{[1:n_i]} \in \mathcal{T}^{n_i}} \sum_s \alpha_s \Phi_s(w^i_{[1:n_i]}, u_{[1:n_i]})$$

where $\mathcal{T}^{n_i}$ is the set of all tag sequences of length n_i.

- If $z_{[1:n_i]} \neq t^i_{[1:n_i]}$ then update the parameters

$$\alpha_s = \alpha_s + \Phi_s(w^i_{[1:n_i]}, t^i_{[1:n_i]}) - \Phi_s(w^i_{[1:n_i]}, z_{[1:n_i]})$$

Output: Parameter vector $\bar{\alpha}$.

This algorithm is dependent on choice of feature vector representation of history-tag pairs. It has an intuitive feeling of up weighing parameter values for ‘missing’ features and down weighing parameter values for ‘incorrect’ features.

Conclusion

Though there are many taggers which outperform the taggers described above but we gained insight that HMM tagger is better than Bigram Tagger, also Bigram is better than Unigram tagger in terms of accuracy and that each method has its own set of advantages and disadvantages.

Bibliography

[1] Collins, Michael. "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms." *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002.