Assignment-Discussion POS tagging using (a) EnCo-DeCo, (b) FFNN-BP

Praneeth,200050028
Janaki Ram,200050112
Varun,200050073
Samarth,19d180029
10/03/2023

Problem Statement: **part 1**

- Given a sequence of words, produce the POS tag sequence
- Technique to be used: Encoder-decoder (use standard word vectors); anything other than transformer
- Use Universal Tag Set (12 in number)
  “ADP”, “PRT”, “ADV” , ”CONJ”, ”NUM”, ”X”, “.”, ”VERB”, ”ADJ”, ”PRON”, ”DET”, ”NOUN”
- 5-fold cross validation

Problem Statement: **part 2**

- Given a sequence of words, produce the POS tag sequence
- Technique to be used: word2vec vectors, FFNN and BP (use libraries)
- Use Universal Tag Set (12 in number)
  “ADP”, “PRT”, “ADV” , ”CONJ”, ”NUM”, ”X”, “.”, ”VERB”, ”ADJ”, ”PRON”, ”DET”, ”NOUN”
- 5-fold cross validation
- Compare with EnCoder-DeCoder

TagSet

ADJ:	Adjective
ADP:	Adposition (preposition or postposition)
ADV:	Adverb
CONJ:	Coordinating conjunction
DET:	Determiner (article, demonstrative, possessive)
NOUN:	Noun (common or proper)
NUM:	Numeral
PRT:	Particle (words that function as a unit with a verb)
PRON:	Pronoun (personal, possessive, relative, etc.)
VERB:	Verb (auxiliary, modal, or main verb)
. :	Punctuation marks such as period, comma, etc.
X:	Other (foreign words, abbreviations, etc.)

Experimental Setup (give details)

- Encoder Decoder Setup

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 40, 300)           14945100

 simple_rnn (SimpleRNN)      (None, 40, 64)            23360

 time_distributed_1 (TimeDis  (None, 40, 13)           845
 tributed)

=================================================================
Total params: 14,969,305
Trainable params: 14,969,305
Non-trainable params: 0
```

Experimental Setup (give details)

- word2vec vectors, FFNN and BP

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 40, 300)           14945100

 dense (Dense)               (None, 40, 10)            3010

 dense_1 (Dense)             (None, 40, 13)            143

=================================================================
Total params: 14,948,253
Trainable params: 14,948,253
Non-trainable params: 0
_________________________________________________________________
None
```

Experimental Setup (give details)

```
[ ]  EMBEDDING_SIZE  = 300
     MAX_SEQ_LENGTH = 40
     VALID_SIZE = 0.2
```

Embedding size is length of embedding vector of a word

Max seq length is maximum length of input sentence.

Valid size is fraction of validation data in total data.

Overall performance (EnCo-DeCo)

- Precision : 0.951
- Recall : 0.954
- F-score (3 values)
  - F1-score : 0.951
  - F0.5-score : 0.949
  - F2-score : 0.953

Overall performance (FFNN-BP)

- Precision : 0.940
- Recall : 0.938
- F-score (3 values)
  - F1-score : 0.938
  - F0.5-score : 0.939
  - F2-score : 0.937

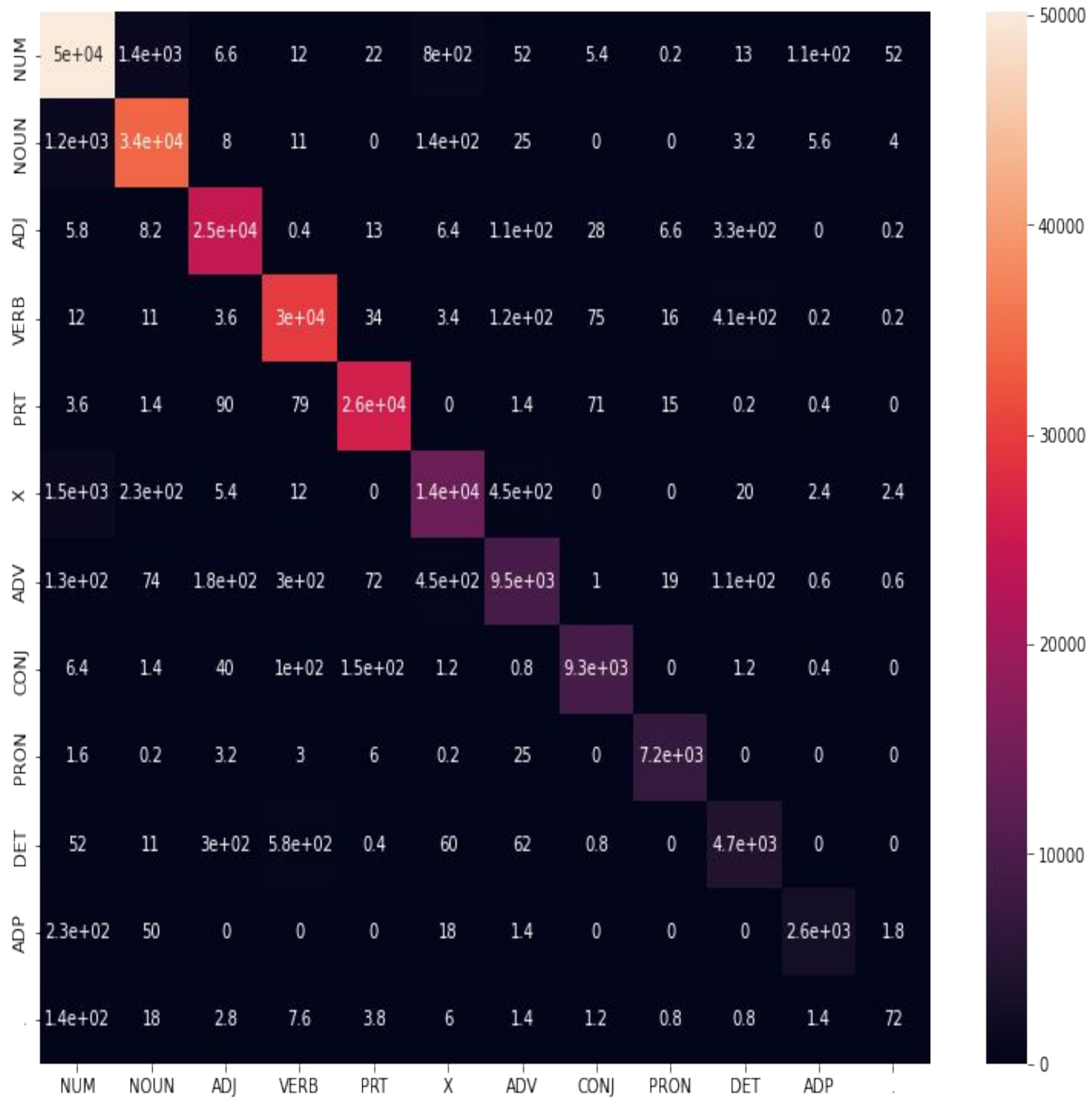
Per POS performance (EnCo-DeCo)

Tags	Precision	Recall	F1-Score
“ADP”	0.943	0.910	0.924
“PRT”	0.987	0.989	0.988
“ADV”	0.917	0.876	0.896
“CONJ”	0.980	0.968	0.974
“NUM”	0.938	0.953	0.946
“X”	0.899	0.861	0.880
“.”	0.553	0.267	0.352
“VERB”	0.966	0.979	0.972
“ADJ”	0.969	0.975	0.972
“PRON”	0.992	0.995	0.993
“DET”	0.842	0.815	0.827

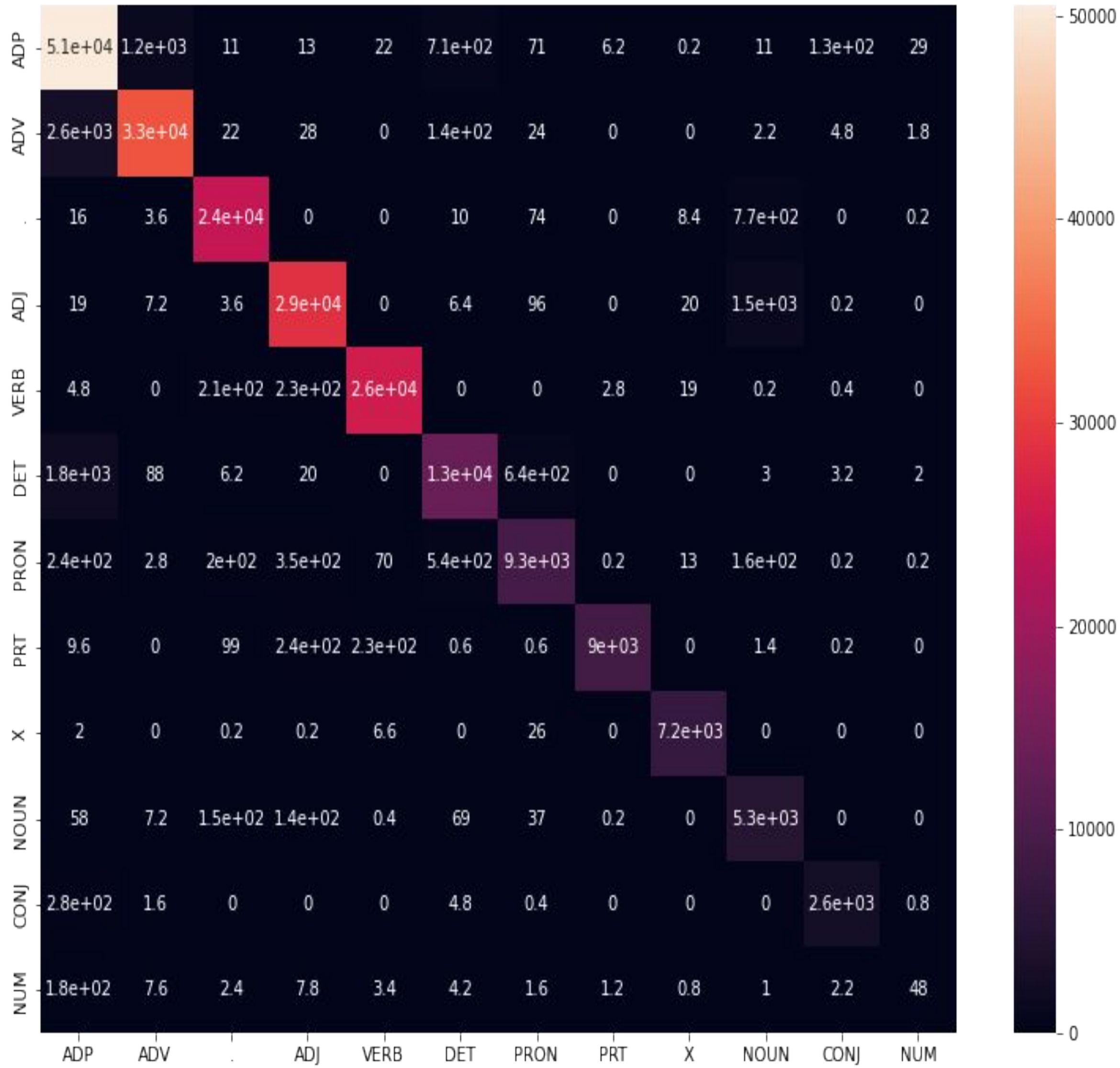
Per POS performance (FFNN-BP)

Tags	Precision	Recall	F1-Score
“ADP”	0.905	0.957	0.930
“PRT”	0.999	0.939	0.968
“ADV”	0.960	0.920	0.940
“CONJ”	0.936	0.913	0.923
“NUM”	0.567	0.174	0.262
“X”	0.992	0.995	0.993
“.”	0.964	0.957	0.961
“VERB”	0.986	0.981	0.984
“ADJ”	0.967	0.948	0.957
“PRON”	0.906	0.853	0.878
“DET”	0.896	0.842	0.868

Confusion Matrix (12 X 12)(heatmap)(EnCo-DeCo)



Confusion Matrix (12 X 12) (heat map) (FFNN-BP)



Interpretation of confusion (error analysis)

- In FFNN-BP NUM got confused largely with ADP
- In FFNN-BP the reason for the confusion between NUM and ADP can be attributed to the fact that some words can function as both NUMs and ADPs depending on their context. For instance, the word "two" can be a NUM when used to express the quantity of something, such as "I have two books." However, it can also function as an ADP when used to indicate a spatial relationship, such as "I'm sitting two feet away from you."

Interpretation of confusion (error analysis)

- In encoder decoder . got confused with NUM
- . can be used to represent decimal points in numbers, such as "3.14," which can be mistaken for a NUM.

Interpretation of confusion (error analysis)

Tags	Most confused tags	Analysis
"PRT"	"ADP"	both can function as small, function words that modify the meaning of a verb or link nouns
"ADV"	"ADJ"	Adverbs are generally confused with adjectives
"CONJ"	"ADV"	Conjunctions are rarely confused
"NUM"	"NOUN"	Numerals can also be used as nouns
"X"	"NOUN"	X is a catch-all category for words that do not belong to any established part of speech
"VERB"	"NOUN"	Many Verbs can also used as nouns
"ADJ"	"NOUN"	Adjectives are generally confused with nouns
"PRON"	"ADP"	Prepositional pronouns
"DET"	"PRON"	Determiners are rarely confused With any other tags

Data Processing Info (Pre-processing)

- We’ve used gensim Word2Vec model from python library.
- We’ve used brown corpus from nltk and universal tagset as tags list.
- In preprocessing first we’ve done lower casing all words in the corpus.
- Then we used a word tokenizer to tokenize the words and it gives 1 as token for out of vocabulary words.

Data Processing Info (Pre-processing)

- We have split the data into 5 batches and we have trained the model 5 times using 4 of the 5 batches at a time and used the remaining batch as test set.
- We've truncated a sentence to max of 40 words and padded it if it has less than 40 words in it.
- This as the input and truncated/padded tags as the output we've trained the models.

Data Processing Info (Pre-processing)

```
word_tokenizer = Tokenizer(oov_token=True)
word_tokenizer.fit_on_texts(sent_words)
VOCABULARY_SIZE = len(word_tokenizer.word_index) + 1
embedding_matrix = np.zeros((VOCABULARY_SIZE,
EMBEDDING_SIZE))
wvmodel = Word2Vec(sent_words, size=300, window=1,
min_count=1)
```

Marking Scheme

- 1. Demo working- 8/8 + 7/7; two problems (if not, 0)
- 2. Implemented EnCo-DeCo and Clarity on the approach- 5/5
- 3. Implemented FFNN-BP and Clarity on the approach- 5/5
- 4. Confusion matrix drawn and error analysed- 5/5 + 5/5 (both approaches)
- 5. Overall F-score > 90- 10/10, >80 & <=90- 8/10, else 6/10
- 6. Unknown word handling- done (5/5; else 0)

Assignment-Presentation POS Tagging Using HMM

Problem Statement

- Given a sequence of words, produce the POS tag sequence

- Technique to be used: HMM-Viterbi

- Use Universal Tag Set (12 in number)

- 5-fold cross validation

- <List the tags here>
  “ADP”, “PRT”, “ADV” , ”CONJ”, ”NUM”, ”X”,
  “.”, ”VERB”, ”ADJ”, ”PRON”, ”DET”, ”NOUN”

Overall performance

- Precision

  0.9247
- Recall

  0.9231
- F-score (3 values)
  - F1-score

    0.9222
  - F0.5-score

    0.9231
  - F2-score

    0.9224

Overall Performance

```
Overall Precision Value:  0.924715949646662
Overall Recall Value:  0.9231040373651866
Overall F1 Score: 0.922225735766357
Overall F2 Score: 0.9224119051493851
Overall F 0.5 Score: 0.9231479227240549
Overall Tag Precision:
```

	DET	NOUN	PRON	NUM	ADJ	X	PRT	CONJ	ADV	ADP	.	VERB
0	0.901308	0.923977	0.857833	0.982722	0.853458	0.800142	0.91374	0.985997	0.901162	0.903061	0.948915	0.96578

```
Overall Tag Recall:
```

	DET	NOUN	PRON	NUM	ADJ	X	PRT	CONJ	ADV	ADP	.	VERB
0	0.959304	0.907494	0.95306	0.781681	0.851793	0.113315	0.841409	0.991875	0.857917	0.970148	0.999205	0.900923

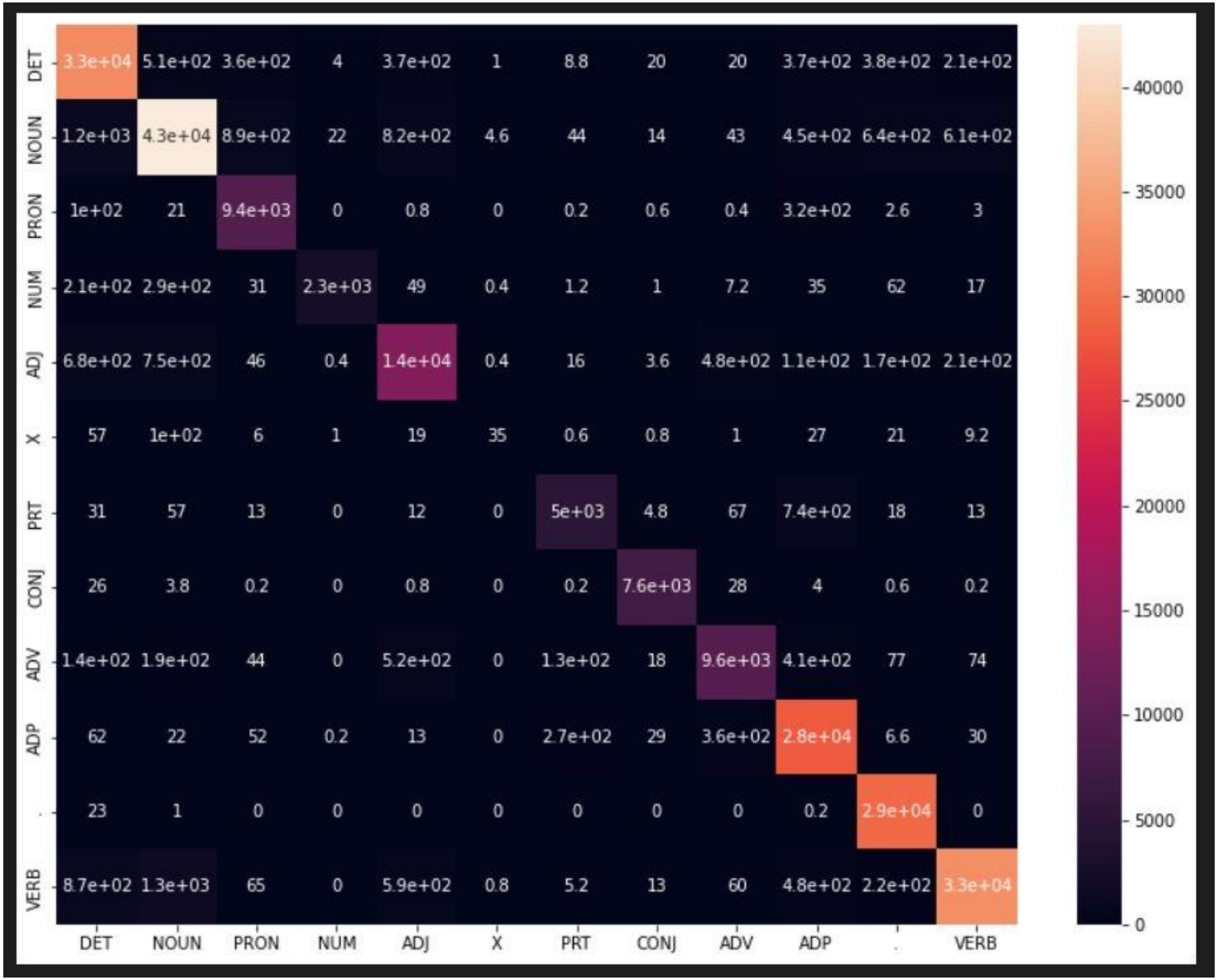
```
Overall Tag F1:
```

	DET	NOUN	PRON	NUM	ADJ	X	PRT	CONJ	ADV	ADP	.	VERB
0	0.927776	0.91435	0.902827	0.868938	0.852546	0.196805	0.876021	0.988919	0.879001	0.935354	0.973354	0.932185

Per POS performance

Tags	Precision	Recall	F1-Score
“ADP”	0.903	0.970	0.935
“PRT”	0.913	0.841	0.876
“ADV”	0.901	0.857	0.879
“CONJ”	0.985	0.991	0.988
“NUM”	0.982	0.781	0.868
“X”	0.800	0.113	0.196
“.”	0.948	0.999	0.973
“VERB”	0.965	0.900	0.932
“ADJ”	0.853	0.851	0.852
“PRON”	0.857	0.953	0.902
“DET”	0.901	0.959	0.927
“NOUN”	0.923	0.907	0.914

Confusion Matrix (12 X 12)



Interpretation of confusion (error analysis)

Tags	Most confused tags	Analysis
“PRT”	**“ADP”**	Many adpositions cannot be declined further
“ADV”	**“ADJ”**	Adverbs are generally confused with adjectives
“CONJ”	**“ADV”**	Conjunctions are rarely confused
“NUM”	**“NOUN”**	Numerals can also be used as nouns
“X”	**“NOUN”**	Due to transition probabilities
“VERB”	**“NOUN”**	Many Verbs can also used as nouns
“ADJ”	**“NOUN”**	Adjectives are generally confused with nouns
“PRON”	**“ADP”**	Prepositional pronouns
“DET”	**“PRON”**	Determiners are rarely confused With any other tags
“NOUN”	**“DET”,”ADJ”**	Many Nouns can also be used as adjectives

Data Processing Info (Pre-processing)

- We have split the downloaded data into 5 batches and we have trained the model 5 times using 4 of the 5 batches at a time and used the remaining batch as test set.
- We have defined a function which takes train sentences as input and returns tag set and word set.
- We also defined a function which takes train sentences as input and calculates p(tag|tag),p(word|tag) values which were stored in matrices and were returned.
- We have used laplacian smoothing for unknown words which make sures that p(word|tag) cannot be zero.

Inferencing/Decoding Info

- When the number of tags in a tagset is N then at each word
- We analysed N*N paths and went to the next word with choosing best N paths among them.

- We have used a numpy array for forward propagation of shape (N, length(sentence)) for storing the maximum probability of tag with every word and also used another numpy array of same shape for backtracking, here for each word,tag we have stored the tag index of previous word for which the probability of current tag is maximised.

- We have found out the maximum value in the last column of forward propagation matrix and then with the help of backtracking matrix we build the tag sequence of a particular sentence.

Marking Scheme

- 1. Demo working- 10/10 (if not, 0)
- 2. Implemented Viterbi and Clarity on Viterbi- 5/5
- 3. Transition and Lexical tables clearly described- 5/5
- 4. Confusion matrix drawn and error analysed- 5/5
- 5. Overall F-score > 90- 10/10, >80 & <=90- 8/10, else 6/10
- 6. Unknown word handling- done (5/5; else 0)

Any thoughts on generative vs. discriminative POS tagging

- In general, a discriminative model models the decision boundary between the classes, and a generative model explicitly models the actual distribution of each class.

- A generative model learns the joint probability distribution $p(x,y)$. It predicts the conditional probability with the help of the Bayes Theorem. A discriminative model learns the conditional probability distribution $p(y|x)$.