**Assignment-Discussion**

**Vector Based POS Tagging**

Butti Vaishnavi, 200050026

Budati Yuva Dhatri, 200050024

Nannapaneni Yasaswi,200050085

…

09-10-2022

# Problem Statement: **part 1**

* Given a sequence of words, produce the POS tag sequence
* Technique to be used: HMM-Viterbi-vector (vector based; the whole corpus is corpus of word vectors which replace words)
* Use Universal Tag Set (12 in number); <list the tags>
* 5-fold cross validation
* Compare with HMM-Viterbi-symbolic

# Problem Statement: part 2

* Given a sequence of words, produce the POS tag sequence
* Technique to be used: word2vec vectors, FFNN and BP (a slide on FFNN-BP architecture is a must)
* Use Universal Tag Set (12 in number); <list the tags>
* 5-fold cross validation
* Compare with HMM-Viterbi-symbolic

# Overall performance

* Precision – 0.994, 0.8715
* Recall-0.992, 0.8616
* F-score (3 values)
  + F1-score-0.9930, 0.8649
  + F0.5-score-0.9939, 0.8683
  + F2-score-0.9922, 0.8627
* For both part 1 and 2 and compare
* Also Compare with HMM-Viterbi-vectoric

Per POS performance

DET: 0.9895 0.9753 0.9823

NOUN: 0.9182 0.9364 0.9272

ADJ: 0.8651 0.8577 0.8614 VERB: 0.9403 0.9366 0.9385 ADP: 0.9347 0.9193 0.9269 Tag-‘.’: 0.9998 0.9835 0.9916 ADV: 0.8481 0.8927 0.8698 CONJ: 0.9949 0.9911 0.993 PRT: 0.7855 0.7191 0.7508 PRON 0.94 0.9937 0.9661 NUM: 0.9025 0.8936 0.898 Tag-‘X’: 0.2215 0.3595 0.2741

* Compare all three models

Confusion Matrix (12 X 12) (can give heat map) (compare all 3 models)

Chart, waterfall chart

Description automatically generatedChart, waterfall chart

Description automatically generated

HMM\_Viterbi\_Vector HMM\_Viterbi

# Interpretation of confusion (error analysis)

For all tags, the corresponding tags that they are most confused with remained same

After unknown word handling by word\_similiarity, in HMM-Viterbi-Vector unknown words are not confused with nouns.

Most of the words which were of another tag and confused with noun are reduced.

Data Processing and Data Sparsity

* <Describe how you obtained the word vectors>

We used genism library which contains Word2Vec (). It takes words as an input and gives the output where each word will get mapped to a vector with specified EMD\_SIZE

* For solving the problem of unseen words use cosine similarity of vectors

1. We used Word2Vec () , in which we trained the whole data corpus, and then whenever we encounter unseen words we used word2vec.similar(), which calculates the cosine similarity and gives the topmost similar vectors.