ELL884 A1
NLP with Deep Learning
POS Tagging with HMM & MEMM

Siddhartha Parupudi
2021EE10135

February 10, 2024

1 Hidden Markov Model

- Created Vocabulary of words from training data.
- Created dictionaries to map distinct tags to distinct indices and vice-versa.
- Calculated prior probabilities of each state with smoothing.
- Calculate the ML estimates of the Transition probabilities and Observation likelihoods with smoothing utilising the counts from the training data and normalised the rows to give a valid probability.
- Implemented a bigram viterbi decoder to predict the most likely sequence of states (POS Tags) given a sentence using a dynamic-programming approach.
- Initially, I handled the OOV words by assigning them a small constant probability. Then, I used the spaCy lemmatizer to stem the OOV word (converts original word to a stemmed word which increases it's chance of being found in the vocabulary). If even the stemmed word is not in the vocabulary, I assigned them a probability which is similar to the most infrequent words in the training corpus (which is around 10^{-6}).
- Obtained Accuracy of 76.6%

2 Maximum Entropy Markov Model

- I 1^{st} created a special tag to handle storage of previous tags when there were no previous tags in the sentence.
- I created a function to return the history given the sentence and the current position.
- I then manually curated 50 features for the model to train on. The feature set spans context features, morphological features and syntactic and semantic features.
- I then calculated the probability of the occurence of each sentence and the Likelihood of the data.
- I then calculated the gradient of the Likelihood function which we have to maximize with respect to the parameters with ℓ_2-regularisation.
- I then implemented the gradient ascent function to train the parameters which were initialised as normally distributed random numbers.
- This turned out to be very slow, so I implemented mini-batch gradient ascent by splitting the training data into 1479 batches of size 32/33.
- I then implemented the Viterbi backtracking algorithm to decode the best possible sequence of states (POS Tags) given a sentence.
- Since I used a trigram model, the time taken to decode one sentence itself is 1 minute. So, it would not be possible to test the algorithm on the entire training dataset of 4000 sentences.
- But it is expected that the MEMM model will perform better than the HMM model as it can capture a vast variety of features related to the structure of the English language not limited to transition probabilities and observation likelihoods.

3 Notebook

- Colab Notebook

4 References

- HMM reference: Jurafsky and Martin Appendix A
- MEMM reference:
  - Log-linear taggers, Columbia University
  - Ratnaparkhi's original paper