# Maman 12:

Some running examples:

## python ./decode.py 2 heb-pos.test exps/hmm-smooth-n.lex exps/hmm-smooth-n.gram

## Question 1: descriptive statistics

1. Number of unigram instances: Train: 11,282 Gold: 127,884
2. Number of unigram types: Train: 3171 Gold: 15986
3. Number of segment-tags: Train: 11,282 Gold: 127,884
4. Number of types of segment-tags: Train: 3,424 Gold: 18,143
5. Ambiguity of POS tagging:

Train: 3,424 / 3,171= 1.079

Gold: 18,143 /15,986 = 1.1349

1. A discussion about the differences between the Gold and the Train files:
2. the train file is ~5 times smaller than the Gold file with respect to segment tag , the number of types of unigrams and segment-tags types.
   1. Number of unigram types: the difference of types of can mainly affect on the **emission probabilities**, and cause to a lack of them which cause to a poor tagging.
   2. Number of types of segment-tag: the difference of types of can mainly affects on the **emission probabilities**, each word may have a few different correct tagging and the relatively small train corpus will miss a variety of tagging possibilities for a certain word which eventually will cause to a poor tagging.
   3. Number of unigram instances/segment-tag instances: the relatively small number of lines in the train corpus can damage the **transition probability** of the tagging model, as we can assume that in the gold file we may see more permutations of the words, this can help getting much more accurate transition probabilities .
   4. Ambiguity of POS tagging: the ambiguity is bigger in the Gold file and this result is quite expectable due to the large number of lines and sentences. Naturally a word may have a various tags and as the bigger the corpus the bigger the ambiguity

## Question 2: the basic tagger

1. Parametric scheme:  
   The conditional probability of tag given word
2. Estimation formula for the parameters:

Maximum likelihood estimate(MLE): Relative Frequency in corpus

1. Model run-time complexity:  
   s- sentence length

t- tags set  
Training complexity:

* 1. Iterating the file line by line and counting for each segment the number of instances for each tag. , we go over each sentence once o(s)
  2. For each segment we iterate over each tag count and take the maximum and calculating the probability to get it,

Training complexity =

Tagging complexity:

Assuming we have different words

1. Iterating over each sentence in the gold file and using the dictionary created in the training tagging each word –

Tagging complexity – ) ~ o(s)

1. Macro-average  
   sentence accuracy for the test corpus:  
   All: 0.106   
   word accuracy for the test corpus:  
   A: 0.830969686226

# Question 3: first order tagger

1. Target function:
2. Parameters scheme:

– Transition probabilities

1. Estimation formula for the parameters:

Maximum likelihood estimate(MLE): Relative Frequency in corpus

🡪

🡪

1. Training run time complexity

Iterating over all words in sentence and counting tag instances of each type of tag and type instances of a certain segment -

Iterating over each sentence words and calculating the conditional probability  *for each tag -*

creating bigram and calculating transition probabilities -creation calculating probabilities , for each tag count bigrams with all other tags

Training run time =

1. !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2. Macro-Avg No smoothing:

sentence accuracy for the test corpus:

All: 0.158

word accuracy for the test corpus:  
 A: 0.85206523666

1. After unknown words smoothing the results improved:

All: 0.222 A: 0.878922176919

1. After add\_1 with small delta, delta =0.00001, the results are slightly improved:

All: 0.222 A: 0.889469952136