NLP

Exercise 2

Names : Waseem Abu Leil
Muaz Abdeen

Question 1

First of all let's write the transition and emission tables

transition

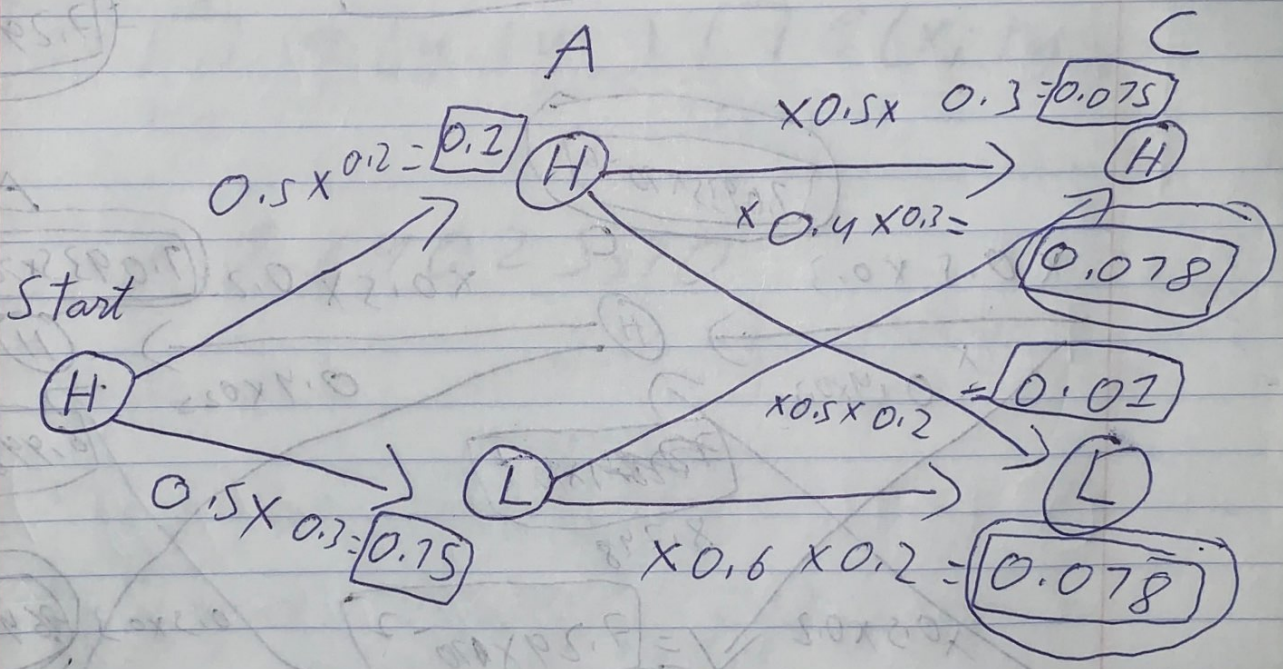
	H	L
H	0.5	0.5
L	0.4	0.6

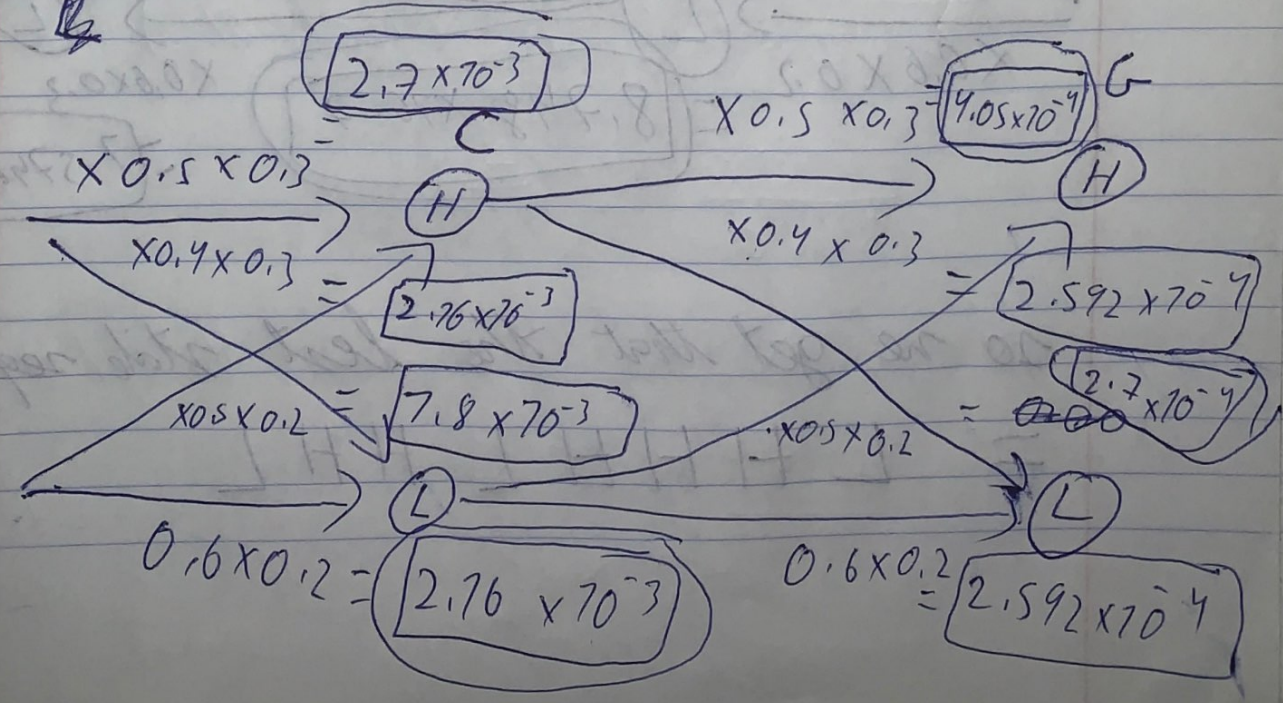
emission

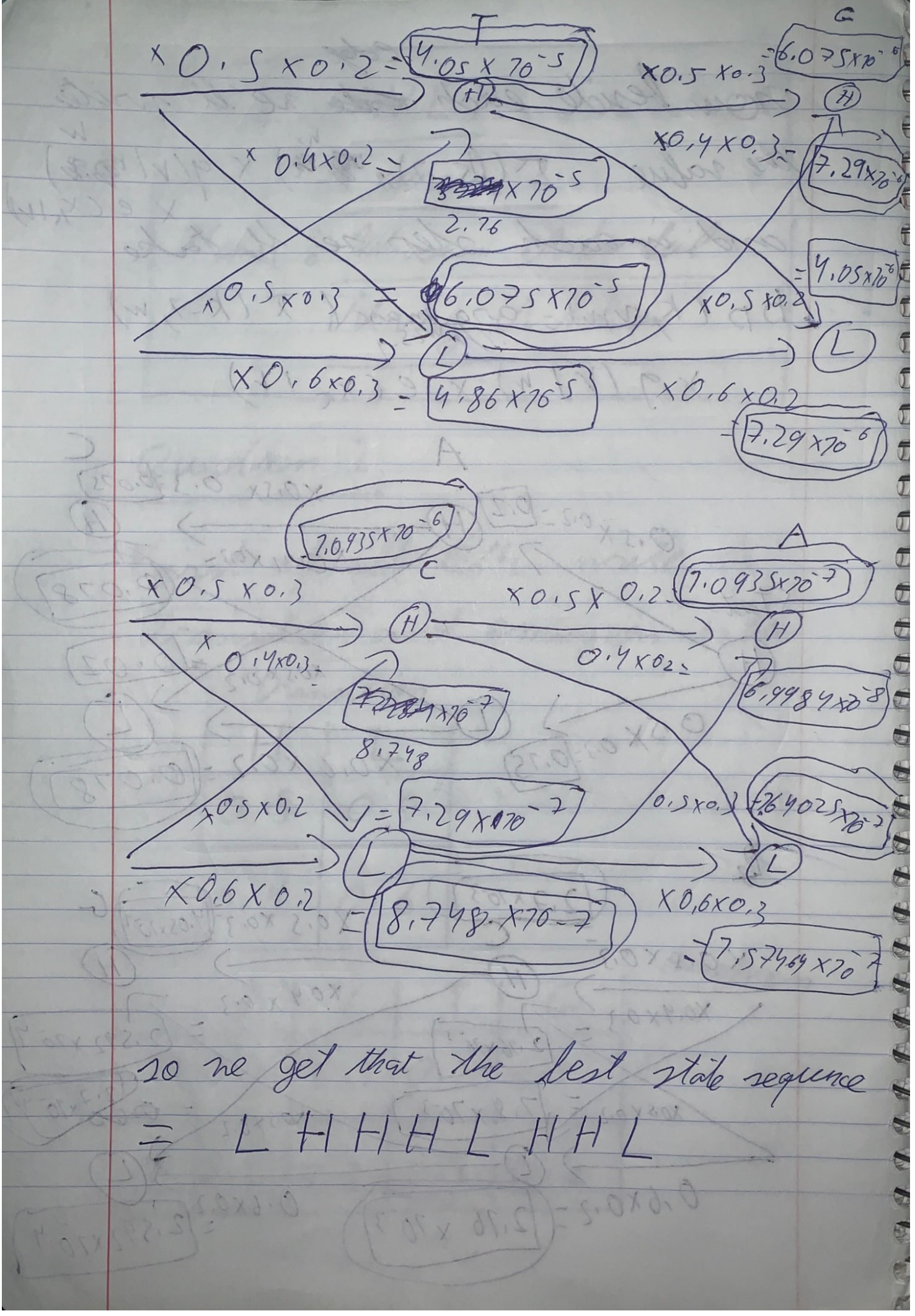
	A	C	T	G
H	0.2	0.3	0.2	0.3
L	0.3	0.2	0.3	0.2

now beside each ~~node~~ edge we'll write the value $\pi((k-1), w) \times q(v|w) \times e(x_k|v)$ and in each step we'll take

$$bp(k, v, v) = \arg\max_{w \in S_{k-1}} (\pi(k-1, w) \times q(v|w) \times e(x_k|v))$$







so we get that the best state sequence

= L H H H L H H L

we denot $(y_1, \ldots, y_8) = LHHHLHHL$

and $(x_1, \ldots, x_8) = S$

to calculat the probability of S

$$P(y_1, \ldots, y_8, x_1, \ldots, x_8)$$

$$= \prod_{i=1}^{8} q(y_i | y_{i-1}) \prod_{i=1}^{8} e(x_i | y_i)$$

$$= 1.64025 \times 10^{-7}$$

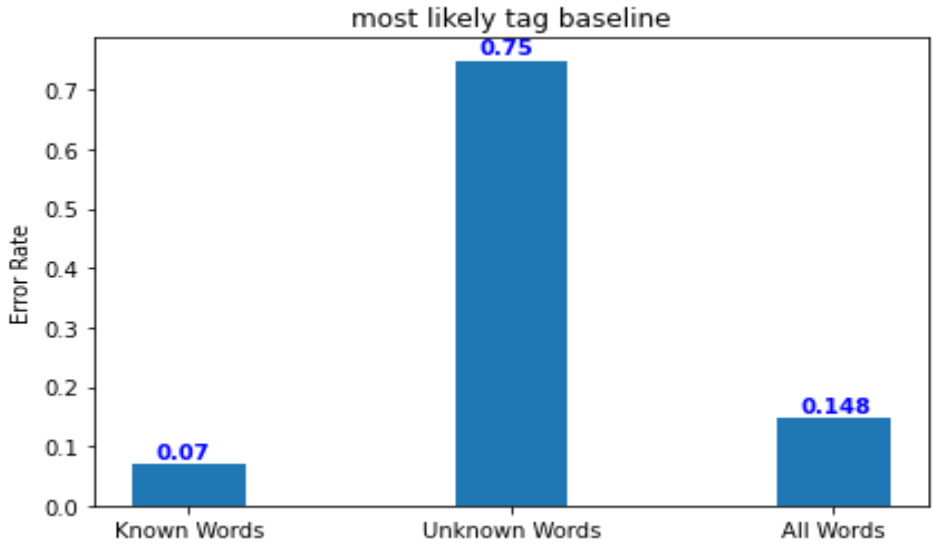
Question 2:

	Algorithm: Modified Viterbi
	Input: An integer n, parameters $q(w\|t,u,v)$ and $e(x,w)$.
	Output: The highest scoring pair of sequences $x_1 \cdots x_n, y_1 \cdots y_n$.
	Definitions: Define K to be the set of possible tags. Define $K_{-2} = K_{-1} = K_0 = \{*\}$, and $K_k = K$ for $k = 1 \cdots n$. Define V to be the set of possible words.
1 -	**Initialization:** Set π to be a table with $n\|K\|^3$ entries, and $\pi(0,*,*,*) = 1$.
2 -	for $k = 1 \cdots n$:
3 -	for $t \in K_{k-2}, u \in K_{k-1}, v \in K_k$:
4 -	$\pi(k,t,u,v) = \max_{w \in K_{k-3},\, x \in V} (\pi(k-1,w,u,t) \times q(v\|w,u,t) \times e(x,v))$
5 -	**Return:** $\max_{t \in K_{n-2},\, u \in K_{n-1},\, v \in K_n,} (\pi(n,t,u,v) \times q(STOP\|t,u,v))$

In step (4) we loop over all possible pairs (w,x), instead of looping over w only, so time complexity to fill one entry in the table is $O(|V| \cdot |K|)$.
The correctness of this change derived from the assumption that the emitted value $x \in V$ is independent of the rest of the variables given some $v \in K$, so we maximizing the emission function over all $x \in V$ given some $v \in K$.

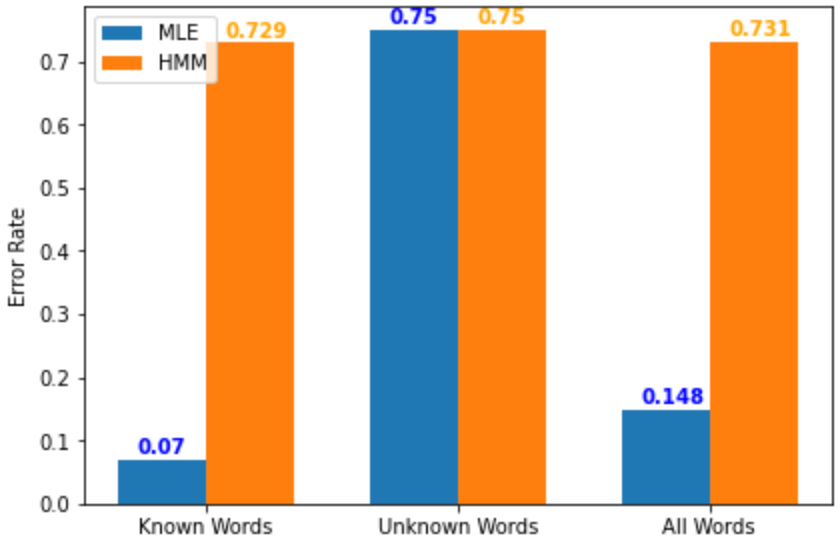
Question 3:

(b)



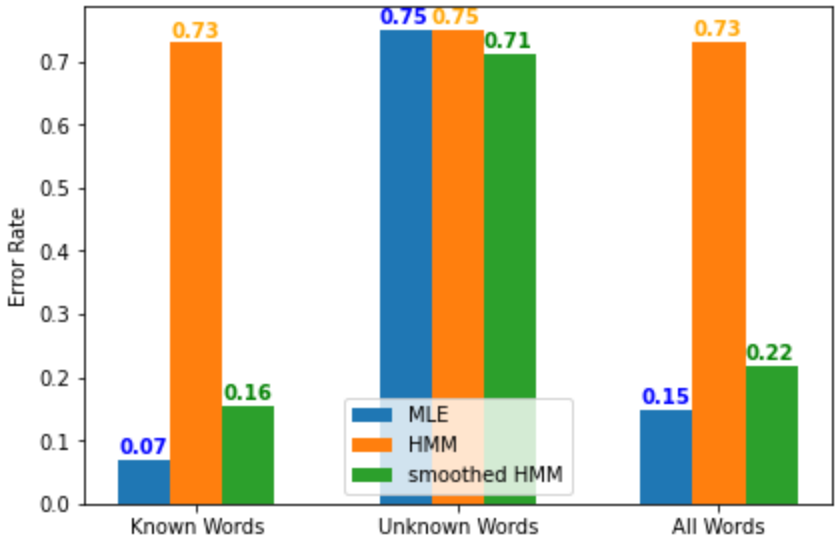
We can see that for the most likely tag baseline it struggles to tag unknown words since we chose an arbitrary tag for unseen words but it's pretty good for the known ones

(c)



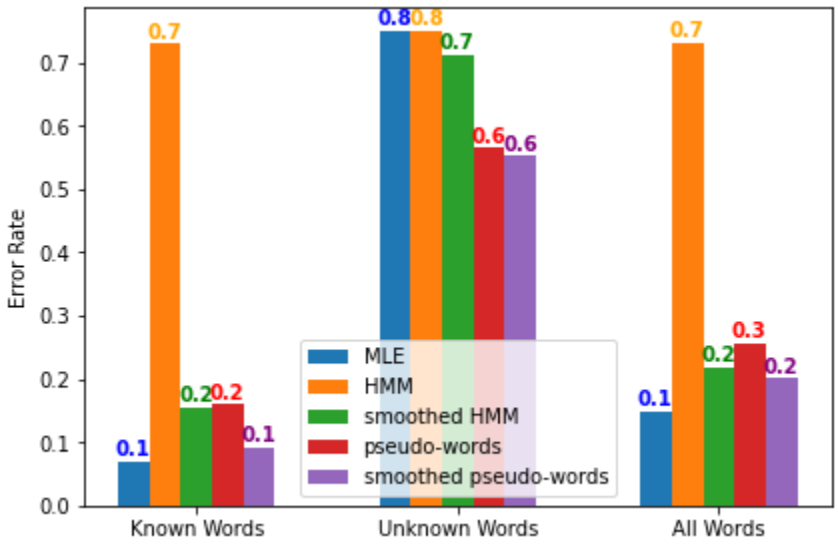
We can see that for known words the most likely tag baseline works much better than the bigram HMM tagger but both perform the same on unknown words

(d)



As expected, the HMM tagger with Laplace smoothing improved tremendously for known words and have a slight improvement on unknown words but is still worse than MLE baseline for known words

(e)



We can see that after using pseudo words the performance on unknown words improved so much and that as expected using smoothing will improve the results

- Regarding the confusion matrices, they can be found on the google colab notebook, I couldn’t put them here because it’s too large