PEE

Exercise 1

Jaime Ferrando Huertas

December 2020

1 Problem 1

For the first problem we are required to justify the **initialization** and **final result** expressions of the *Outside* algorithm.

- Initialization: $\forall A \in N$

$$f(A, 0, T) = \delta(A, S) \tag{1}$$

  Where δ is a binary function that returns 1 if $A == S$ or 0 if $A \neq S$. The reasoning for this is behind the trivial case of the outside algorithm, where the sub-state A is equal to the initial state S and therefore the probability must be equal to 1.

- Final result:

$$P_\theta(x) = \sum_{A \in N} f(A, i, i+1) * p(A \rightarrow x_{i+1}) \tag{2}$$

  We can understand this final result as the sum of every right and left production multiplied by the probability of producing A. The outside algorithm computes all the outside possible productions so we can determine the probability of a given sentence, this is done recursively.

2 Problem 2

In this problem we need to modify the *Viterbi* algorithm for Conditional random Fields to allow for the recovery of the optimal sequence.

We have seen in lectures how to use Viterbi to calculate the score of an optimal sequence $x_1, x_2, x_3, ..x_n$ ending at $y_t = S \in \mathcal{Y}$. This means that we were already visiting the optimal path of states and we only need to modify out algorithm to be able to retrieve the optimal sequence. We do so by adding an extra term b_t that will work as a back pointer, every optimal sequence step $\forall t$ will be stored. Our new Viterbi algorithm is as follows:

- Definition

$$\gamma_t(s) \stackrel{\text{def}}{=} \max_{y_1^t; y_t = s} \prod_{i=1}^{t} \Psi_i\left(y_{i-1}, y_i, x_i\right) \tag{3}$$

$$b_t(s) \stackrel{\text{def}}{=} \arg \max_{y_1^t; y_t = s} \prod_{i=1}^{t} \Psi_i\left(y_{i-1}, y_i, x_i\right) \tag{4}$$

- Initialization: $\forall s \in \mathcal{Y}$

$$\gamma_1(s) = \Psi_1\left(y_0 = \text{ null}, y_1 = s, x_1\right) \tag{5}$$

$$b_t(s) = 0 \tag{6}$$

- Recursion: $\forall t = T...2;\ \forall s \in \mathcal{Y}$

$$\gamma_t(s) = \max_{s' \in \mathcal{Y}} \left\{\gamma_{t-1}\left(s'\right) \cdot \Psi_t\left(t_{t-1} = s', y_t = s, x_t\right)\right\} \tag{7}$$

$$b_t(s) = \arg\max_{s' \in \mathcal{Y}} \left\{\gamma_{t-1}\left(s'\right) \cdot \Psi_t\left(t_{t-1} = s', y_t = s, x_t\right)\right\} \tag{8}$$

- Final Result: The optimal score for x is

$$\max_s \ \gamma_T(s) \tag{9}$$

  Optimal sequence $\hat{y}$ for x:

$$y_t = \arg\max \gamma_T(s) \tag{10}$$

$$\forall t = T...2;\ y_{t-1} = b_t(y_{t)} \tag{11}$$

$$\hat{y} = y_1, y_2, ..., y_t \tag{12}$$

With the introduction of b_t we can look back to the most optimal choice for every step of our sequence and recover the optimal path.

3 Problem 4

For problem number four we are required to modify CKY algorithm to work with a *left*-branching grammar. Meaning we have to modify the recursive algorithm to not explore recursively the right children node as it will always be a terminal one. We use the same CKY definition and initialization as the one seen in slides and modify the recursion step. Our modification to the recursion step is as follows:

$$\forall A \in N, l : 2...T, \forall i : 0...T - l$$

$$e(A, i, i + l) = \sum_{B,C \in N} p(A \to BC) \ \cdot \ e(B, i, i + l - 1) \ \cdot \ e(C, i + l - 1, i + l) \tag{13}$$

Finally we will return:

$$e(S, 0, T) \tag{14}$$

With the new modification the second part of the recursion step becomes $e(C < i+l-1, i+l)$ and the right most node is always terminal. If we know that the last node will always be terminal we can also remove the sum over K. The final cost of our algorithm will be $\mathcal{O}n^2$ for n being the length of the sequence.

4 Practical exercise

For the practical exercise we are require to use the SCFG toolkit for a triangle recognition problem. The objective is to recognize if a triangle is equilateral, isosceles or scalene.

Three different models were learned in several ways (G1, G2, G3) for each of the three classes of triangles. We also have a test set for every type of triangle with 1000 samples in each one of them. The chosen metric to evaluate our models is the set perplexity, defined as:

$$2^{\frac{-1}{N}\sum_x \log_2 P_M(x)} \tag{15}$$

Where $P_M(x)$ is the probability assigned to x by our model M. A simple way to understood the perplexity can also be how surprised is our model when we provide it with new samples.

Now we provide the perplexity results of testing each set of models (G1, G2, G3) with the mentioned test sets. All results are using the inside algorithm in the decode step.

Sub-model	Test set	Perplexity
G1-EQ	TS-EQ	99095.202
G1-EQ	TS-IS	191.884
G1-EQ	TS-SC	245.913
G1-IS	TS-EQ	47334.820
G1-IS	TS-IS	26581.814
G1-IS	TS-SC	21621.971
G1-SC	TS-EQ	48369.445
G1-SC	TS-IS	27508.970
G1-SC	TS-SC	33618.351

Table 1: G1 models perplexity results

For the G1 models we can see similar performance for G1-IS and G1-SC across all test sets, is the model G1-EQ that obtains a incredibly high score for test sets TS-IS,TS-SC. This is something that surprises us as we expected G1-EQ to perform better on the EQ test set and we found the opposite in this case.

For the G2 models we can see similar performance for G2-IS and G2-SC across all test sets with G2-IS being worse in TS-EQ. G2-EQ performs almost

Sub-model	Test set	Perplexity
G2-EQ	TS-EQ	635136.427
G2-EQ	TS-IS	389521.316
G2-EQ	TS-SC	520221.737
G2-IS	TS-EQ	102498.669
G2-IS	TS-IS	52217.595
G2-IS	TS-SC	49786.006
G2-SC	TS-EQ	68152.556
G2-SC	TS-IS	42723.115
G2-SC	TS-SC	43543.300

Table 2: G3 models perplexity results

one degree of magnitude worse in all test sets than the other G2 models. Is also worth mentioning the poor performance of G2-EQ on the TS-EQ, something we were not expecting as will probably mean this grammar is not capable of generalizing EQ triangles. We have seen a similar behaviour with G1 models where the EQ model will perform different to IS,SC but in this case is worse performance while with G1 it was performing better.

Sub-model	Test set	Perplexity
G3-EQ	TS-EQ	986.206
G3-EQ	TS-IS	550208.126
G3-EQ	TS-SC	1081980.366
G3-IS	TS-EQ	1283.379
G3-IS	TS-IS	1254.897
G3-IS	TS-SC	1218.742
G3-SC	TS-EQ	1272.729
G3-SC	TS-IS	1258.354
G3-SC	TS-SC	1218.039

Table 3: G2 models perplexity results

For the G3 model we can see similar performance for G3-IS and G3-SC across all test sets. Again we encounter the G3-EQ model performing different than IS,SC. This time results are worse than the other models in IS,SC test sets while the EQ test set obtains an all time best score, this is something we expect to happen as we are with the EQ version of G3.

Overall we can say that the EQ versions of G1,G2,G3 have different behaviour than the IS, SC counterparts with worse results in G2 and G3 but better results in G3. There were two exceptions, first in G1 where the G1-EQ model performance was very low with TS-EQ while the others were very high, second in G3 where G3-EQ performance was an all time best TS-EQ. This leads us to think that the grammar G3 will be a better choice to represent EQ triangles than the G1.

We also created a confusion matrix for each set of models with the provided confus script. Results below:

	EQ	IS	SC	Err	Err%
EQ	597	285	118	403	40.3
IS	88	471	441	529	52.9
SC	71	406	523	477	47.7

Table 4: G1 confusion matrix

	EQ	IS	SC	Err	Err%
EQ	281	211	508	719	71.9
IS	71	215	714	785	78.5
SC	81	190	729	271	27.1

Table 5: G2 confusion matrix

	EQ	IS	SC	Err	Err%
EQ	789	102	109	211	21.1
IS	178	512	310	488	48.8
SC	106	421	473	527	52.7

Table 6: G3 confusion matrix

With the confusion matrix of each grammar we get a different view at our models performance, being able to confirm some of out theories. G3 has the lower error for EQ triangles than G1,G2.