Natural Language Processing - Exercise 3

Due: 22.12.2024 23:59

Please submit **two** files – a pdf file for the answers to the questions and a zip archive. The zip file should contain your code files and a README txt. Please do not put the pdf in the zip.

1. (10 pts) Consider this (toy) biological setup:
   A cell can be in one of two states - H, for high GC-content, and L for low GC. On each time step the cell produces one nucleotide, A,C,T or G, and might also change its state. The probability of changing from state H to L is 0.5, and from state L to H is 0.4.
   In state H the probabilities for producing nucleotides are 0.2 for A, 0.3 for C, 0.3 for G and 0.2 for T. In L the probabilities are 0.3 for A, 0.2 for C, 0.2 for G and 0.3 for T.
   Consider the nucleotide sequence $S = ACCGTGCA$. Use the Viterbi algorithm to find the best state-sequence and calculate the probability of S given this state-sequence. Assume the previous state before S was H.

2. (10 pts) In class we saw the trigram HMM model and the corresponding Viterbi algorithm. We will now make two main changes. First, we will consider a four-gram tagger, where p takes the form:

$$p(x_1 \cdots x_n, y_1 \cdots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i|y_{i-3}, y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i) \tag{1}$$

   We assume in this definition that $y_0 = y_{-1} = y_{-2} = *$, where $*$ is the START symbol, $y_{n+1} = STOP$, and $y_i \in \mathcal{K}$ for $i = 1 \cdots n$, where $\mathcal{K}$ is the set of possible tags in the HMM.
   Second, we consider a version of the Viterbi algorithm that takes as input **an integer** n (and not a sentence $x_1 \cdots x_n$ as we saw in class) and finds

$$\max_{y_1 \cdots y_{n+1}, x_1 \cdots x_n} p(x_1 \cdots x_n, y_1 \cdots y_{n+1})$$

   for a four-gram tagger, as defined in Equation 1. $x_1 \cdots x_n$ may range over the values of some fixed vocabulary $\mathcal{V}$. Complete the following pseudo-code of this version of the Viterbi algorithm for this model . The pseudo-code must be efficient.

   Input: An integer n, parameters $q(w|t, u, v)$ and $e(x|s)$.
   Definitions: Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-2} = \mathcal{K}_{-1} = \mathcal{K}_0 = \{*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \cdots n$. Define $\mathcal{V}$ to be the set of possible words.
   Initialization: $\cdots$
   Algorithm: $\cdots$
   Return: $\cdots$

3. (80 pts) In this programming exercise with Python, we will implement several versions of an HMM POS tagger.

NLP - תרגיל 3

דרור פרונטר ואורי גובר

1. (10 pts) Consider this (toy) biological setup:
A cell can be in one of two states - H, for high GC-content, and L for low GC. On each time step the cell produces one nucleotide, A,C,T or G, and might also change its state. The probability of changing from state H to L is 0.5, and from state L to H is 0.4.
In state H the probabilities for producing nucleotides are 0.2 for A, 0.3 for C, 0.3 for G and 0.2 for T.
In L the probabilities are 0.3 for A, 0.2 for C, 0.2 for G and 0.3 for T.
Consider the nucleotide sequence $S = ACCGTGCA$. Use the Viterbi algorithm to find the best state-sequence and calculate the probability of S given this state-sequence. Assume the previous state before S was H.

1) נשתמש באלגוריתם ויטרבי שלמדנו בכיתה:

נגדיר $n = 8$ אורך הרצף

נגדיר S_k עבור $k = 0, \ldots, 8$ הסט של המצבים האפשריים

ב- $\{H, L\}$ בפוזיציה k כאשר $S_0 = \{H\}$ (נתון).

נגדיר פונ' r:

$$r(y_0, \ldots, y_k) = \prod_{i=1}^{k} q(y_i \mid y_{i-1}) \cdot \prod_{i=1}^{k} e(x_i \mid y_i)$$

כאשר e פונקציית emmission כפי שלמדנו ו-q ההסתברות לעבור

ל state y_i בהינתן ה-state הקודם y_{i-1}.

באופן, נגדיר טבלת תכנון דינאמי דו-מימדית

כל כניסה תסומן ב $\pi(k,t)$. וגם נשמור

את ההסתברות המירבית לרצף באורך k שנגמר במצב t.

כלומר,

$$\pi(k,t) = \max_{\langle y_0, y_1, \ldots, y_k \rangle : y_0 = H, y_k = t} \left(r(y_0, \ldots, y_k) \right)$$

נתחיל למלא את הטבלה:

		A	C	C	G	T	G	C	A
	0	1	2	3	4	5	6	7	8
H	1	0.1	0.018	$\frac{2.7}{10^3}$	$\frac{4.05}{10^4}$	$\frac{4.05}{10^5}$	$\frac{7.29}{10^6}$	$\frac{1.0935}{10^6}$	$\frac{1.0935}{10^7}$
L	0	0.15	0.018	$\frac{2.16}{10^3}$	$\frac{2.7}{10^4}$	$\frac{6.075}{10^5}$	$\frac{7.29}{10^6}$	$\frac{8.748}{10^7}$	$\frac{1.64025}{10^7}$

בכל שלב אנחנו נשמור פוינטר למקום ממנו הגענו כך נוכל למלא את S_8.

בסוף נקבל לרצף $S = ACCGTGCA$

את רצף המצבים $S_T = (H, L, H, H, H, L, H, H, L)$ בהסתברות $\frac{1.64025}{10^7}$

2. (10 pts) In class we saw the trigram HMM model and the corresponding Viterbi algorithm. We will now make two main changes. First, we will consider a four-gram tagger, where p takes the form:

$$p(x_1 \cdots x_n, y_1 \cdots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i|y_{i-3}, y_{i-2}, y_{i-1}) \prod_{i=1}^{n} e(x_i|y_i) \quad (1)$$

We assume in this definition that $y_0 = y_{-1} = y_{-2} = *$, where $*$ is the START symbol, $y_{n+1} = STOP$, and $y_i \in \mathcal{K}$ for $i = 1 \cdots n$, where $\mathcal{K}$ is the set of possible tags in the HMM.
Second, we consider a version of the Viterbi algorithm that takes as input **an integer** n (and not a sentence $x_1 \cdots x_n$ as we saw in class) and finds

$$\max_{y_1 \cdots y_{n+1}, x_1 \cdots x_n} p(x_1 \cdots x_n, y_1 \cdots y_{n+1})$$

for a four-gram tagger, as defined in Equation 1. $x_1 \cdots x_n$ may range over the values of some fixed vocabulary $\mathcal{V}$. Complete the following pseudo-code of this version of the Viterbi algorithm for this model . The pseudo-code must be efficient.

Input: An integer n, parameters $q(w|t, u, v)$ and $e(x|s)$.
Definitions: Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-2} = \mathcal{K}_{-1} = \mathcal{K}_0 = \{*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \cdots n$. Define $\mathcal{V}$ to be the set of possible words.
Initialization: $\cdots$
Algorithm: $\cdots$
Return: $\cdots$

(2)

<u>אתחול</u>: נאתחל טבלה תלת-מימדית בגודל $n \times |V| \times |K|$.

כל כניסה בטבלה תיוצג על ידי $\pi(i, y, x)$ ותייצג את

ההסתברות המירבית של רצף מילים כלשהו באורך i שנגמר

במילה x ובתגית y. בנוסף נגדיר $\pi(0, y, x) = 1$

לכל $x \in V$ ו-$y \in K$.

בנוסף נגדיר

$$V_{-3} = V_{-2} = V_{-1} = V_0 = \{*\}, \quad V_i = V \text{ for } i \in \{1, .., n\}$$

$$K_{-3} = K_{-2} = K_{-1} = K_0 = \{\#\}, \quad K_i = K \text{ for } i \in \{1, ..., n\}$$

empty tag for $*$

אלגוריתם:

for $i = 1, \dots, n$

  for $y \in K$

    for $x \in V$

$$\pi(i, y, x) = \max_{\substack{y \in K_i,\ y''' \in K_{i-3}, \\ y'' \in K_{i-2},\ y' \in K_{i-1}, \\ x \in V_i}} \pi(i-1, y, x) \cdot q(y \mid y''', y'', y') \cdot e(x_k \mid y)$$

Return $\max_{x \in V_n,\ y \in K_n} \left(\pi(n, x, y)\right)$

הסבר קצר: בשלב הלולאות אנחנו ממלאים את הטבלה. לבסוף

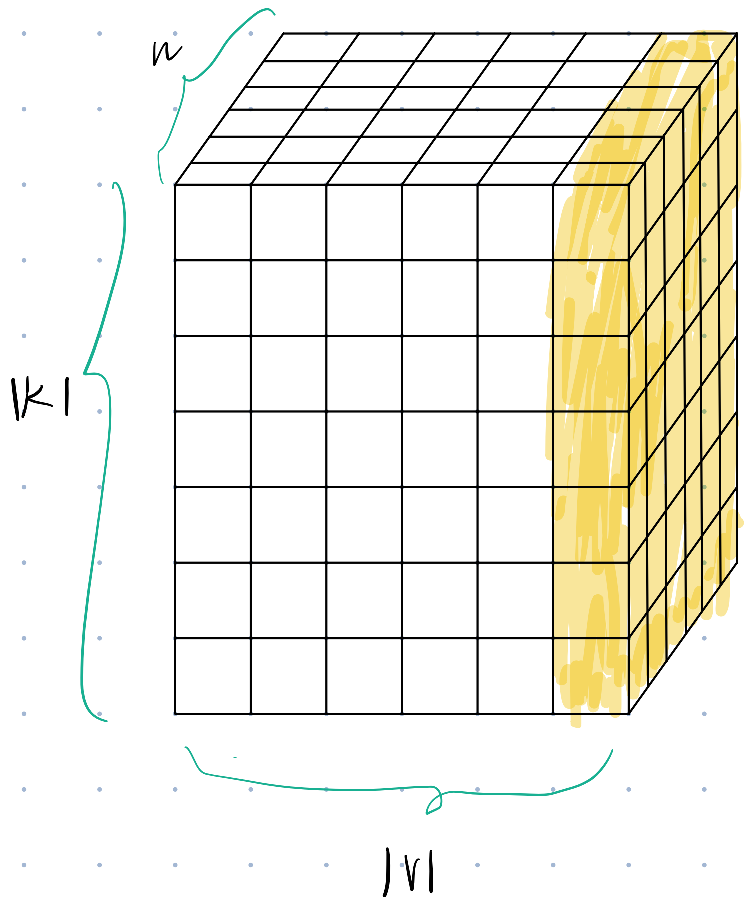
נקח טבלה תלת מימדית שכל "פרוסה" בה ייצג טבלה דו מימדית

"הפרוסה" האחרונה היא טבלה של כל ההסתברויות המקסימליות לכל

ערכי X השונים. לכן נצטרך לעבור על הטבלה הדו מימדית

הזו ולהחזיר מה את הערך המקסימלי.

אוליטרציה למח שנה:



הפרוסה המסומנת בצהוב היא כל ערכי $T(n,x,y)$

המתאימים עבור $x \in V$ כלשהו ו- $y \in K$ כלשהו:

לכן נצטרך לעבור רק על החלק הזה בגלל הנוסחה.

זמן הריצה יהיה $O(n \cdot |V| \cdot |K|^3)$ למילוי הטבלה

ועוד $O(|V| \cdot |K|)$ לחילוץ המקסימום.