CENG 461 Artificial Intelligence
Homework 2

Due date: 10.12.2022

Using a corpus of 58110 English words, our aim is to generate synthetic word samples using a Markov chain, which has a total number of 27 states: all lower case letters in the English alphabet (**26 letters**) and an end-of-word symbol (*). The Markov Chain has an order of 1, which implies the following conditional independence: $P(L_N \mid L_{N-1}, L_{N-2}, \ldots, L_{N-k}) = P(L_N \mid L_{N-1})$.

Using Python and the packages **numpy**, **matplotlib.pyplot** and **random** if you need so:

1. Estimate $P(L_0)$ and $P(L_N \mid L_{N-1})$ and print it.
2. Calculate the average length of a word using the given list of words and print it.
3. Implement a function (**calcPriorProb1**) which takes the given list of words andN as input and returns $P(L_N)$. Plot the distributions for N=1,2,3,4,5 using bar plots.
4. Implement a function (**calcPriorProb2**) which takes $P(L_0)$,$P(L_N \mid L_{N-1})$ (estimated at **Step 1**) and N as input and returns $P(L_N)$. Plot the distributions for N=1,2,3,4,5 using bar plots.
5. Implement a function (**calcWordProb**) which takes $P(L_0)$,$P(L_N \mid L_{N-1})$ (estimated at **Step 1**) and a word as input and returns its probability, e.g. $P(L_0=w, L_1=o, L_2=r, L_3=d)$, assuming the Calculate and print the probabilities for the following words:

   sad*, exchange*, antidisestablishmentarianism*, qwerty*, zzzz*, ae*
6. Implement a function (**generateWords**) which takes $P(L_0)$, $P(L_N \mid L_{N-1})$ (estimated at **Step 1**) and M as input and returns randomly sampled M English words using the given probabilities. Print 10 of them.
7. By generating a synthetic dataset of size 100000, estimate the average length of a word and print it.
8. **BONUS** Generate words after Increasing the order of the Markov chain by making each letter dependent on not only one ($P(L_N \mid L_{N-1})$) but two ($P(L_N \mid L_{N-1}, L_{N-2})$) or more previous letters ($P(L_N \mid L_{N-1}, L_{N-2}, \ldots, L_{N-k})$), in order to generate better samples. Keep in mind that the size of the conditional probability table will increase exponentially as the dependency depth increases. When generating a word, at the beginning, you will either need to use $P(L_0, L_1, \ldots, L_k)$ (which means you cannot generate words shorter than k and words can only start with k-tuples that are in the dataset), or $P(L_0)$, $P(L_N \mid L_{N-1})$, …, $P(L_N \mid L_{N-1}, L_{N-2}, \ldots, L_{N-k-1})$ (which means word of any length can be generated.)