Information Theory

Lab 1: Approximating natural languages

Description of the tasks

All tasks realized during classes are `.pdf` files formatted similarly as this document. The tasks will be of different kinds. Every task will be appropriately marked:

- Tasks to be realized during classes are marked with $\square$ – you won't get points for them, but you still need to do them.
- Pointed tasks to be realized during classes are marked with $\diamond$ – you need to do them during class and show to your teacher, and in the case you don't manage to do so (or are absent) they become your homework ($\star$).
- Homeworks are marked with $\star$ – they also have assigned a number of points, and you need to deliver them to your teacher before a deadline (usually before the next class).
- You may use any programming language you like for the programming tasks, but you are limited to only the standard library of that language and libraries widely accepted as standard.

Introduction

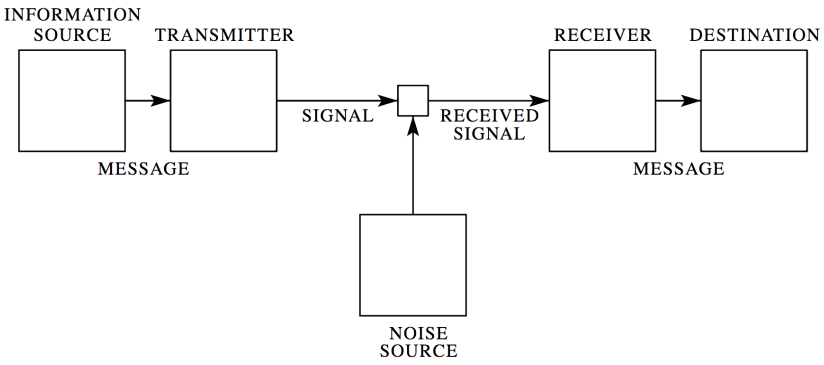

Figure 1: The diagram of a general communication system proposed by Claude Shannon

Objective

During this lesson, we want to focus on a discrete information source. We can think about it as a generator of messages, which generates them sequentially symbol after symbol. The source selects a new symbol with a certain probability, which usually depends on the previously provided symbols.

We can thus treat the discrete information source as a stochastic process. And vice versa, we can treat every stochastic process with a discrete sequence of events from a finite set, as an information source.

Natural languages (such as English) also can be treated as an information source. The same applies to sound, picture, or video, as long as their representation is discrete.

The exercises in this document are focused on the creation of artificial information sources, which with the growing complexity will provide better and better approximations of the English language. We will also show, how statistical knowledge can be used to reduce the time needed to send a message or the size of a message.

1 Preparation □

Task

- Download text corpus from the eKursy website.
- Files in the corpus contain 26 lower case letters of the English alphabet, digits, and a space (in total 37 characters).
- Write a function which will load the files into memory.

2 Zeroth-order approximation □

Task

In this task, we will generate a zeroth-order approximation of the text in the English language, by which we mean an approximation in which each character (26 lower case letters + space) has the same probability of occurrence (1/27).

Hint: in next tasks you will too generate some approximations of text. Thus, it may save you some work of writing redundant code if the generator accepts the information source as an argument.

What is the average length of a word in this approximation? By 'word' we mean, as usual, a sequence of letters without a space (and we assume that two spaces doesn't generate a word of length 0 between them).

3 Frequency of letters □

Task

Are different characters equally likely in the real text? Compute the frequency of characters in the English texts in the corpus. Hint: you don't have to count all characters in the corpus, a "big enough" sample would suffice.

Which characters are the most probable, and which are the least probable? Compare the frequency of characters with the codes assigned to the letters in the Morse code – do you see some similarities?

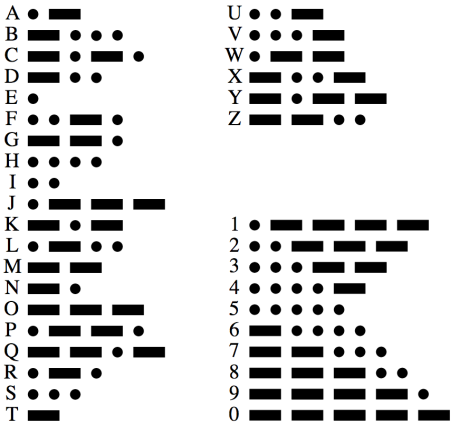

Figure 2: International Morse code

4 First-order approximation □

Task

Using the probabilities computed in the previous task, generate a new sequence of characters – a first-order approximation, in which characters are generated with the probability of their occurrence in text.

What is the average length of words in this approximation?

5 Conditional probability of letters □

Task

Are all characters equally likely to occur after each other in texts? For the two most frequent characters in the corpus, compute a conditional probability of all other characters occurring after them in text.

Example

Let's consider the following corpus: "beekeepers keep bees in a beehive".

The letter "e" is the most frequent in this sentence. In the first-order approximation of that corpus, "e" would have the probability 12/32. However, there is no word in this corpus which begins with "e" – after a space, which there are 5 in the text, the letter "e" never occurs.

This suggests that we can achieve even better approximation if we take this more complicated structure into account, instead of choosing characters at random independently. We can create a first-order Markov information source, which generates characters with the probability:

$$P(j|i) = P(i,j)/P(i),$$

where $P(i)$ is the probability of the letter i in the text, $P(i,j)$ is the probability of the bigram (a pair of consecutive letters) (i,j), and $P(j|i)$ is the conditional probability of the occurrence of the letter j after the letter i.

Let's assume that we are given the text `"eaebe"`. Then:

$$P(\texttt{"e"}) = 3/5$$
$$P(\texttt{"ea"}) = 1/4$$

For convenience, let's assume that we denote a conditional probability that 'a' appears after 'e' as: $P(\texttt{"e|a"}) = P(\texttt{"a"}|\texttt{"e"})$. When computing $P(\texttt{"e|a"})$, we need to have the shared space of probabilistic events, and thus the computations would look like this:

$$P(\texttt{"e|a"}) = \frac{P(\texttt{"ea"})}{P(\texttt{"e*"})} = \frac{1/4}{2/4} = 1/2$$

This also works for conditional probabilities of higher order:

$$P(\texttt{"ea|e"}) = \frac{P(\texttt{"eae"})}{P(\texttt{"ea*"})} = \frac{1/3}{1/3} = 1$$

In 1948 Shannon could not easily use a computer to automatically compute these probabilities for a large corpus of text, so he used a kind of Monte Carlo method in order to approximate the Markov information source. He

would take a book, open it on a random page, choose a random line and a position within it, and then he was moving along the text as long as he found a selected earlier letter i and noted the letter j which occurred directly after it. He repeated this process many times.

6 Approximations based on Markov sources *10pt*◇

Task

Generate an approximation of the English language text using the first-order Markov source (probability of the next character depends on the 1 previous character). ***(3pt)***

Then do the same, but for the third-order Markov source (probability of the next character depends on the 3 previous characters). Use first and second order Markov sources to generate the starting characters. ***(3pt)***

Finally, do the same for the fifth-order Markov source. Begin with a sequence of characters starting with "probability". ***(4pt)***

What is the average length of the word in those approximations?

If your computer does not have enough memory, then make certain simplifications in order to minimize its usage. For example, you may ignore sequences of characters which occur in the corpus less often than some assumed threshold k. You can also use fourth-order Markov source instead of fifth-order.

Example expected outputs:

first-order: `e t tewed olond iancchenooni weg`
third-order: `uken first tole sovincludes of m`
fifth-order: `probability the and for wisner dream henry`