Statistical Natural Language Processing
Assignment 3

Sven Stauden 2549696
Janis Landwehr 2547715
Carsten Klaus 2554140

Exercise 1

Proof

$$
\begin{aligned}
H(X,Y) &= -\sum_x \sum_y P(x,y) log_2(P(x,y)) \quad \texttt{Definition} \\
&= -\sum_x \sum_y P(x,y) log_2(P(y|x)P(x)) \quad \texttt{see Ch.2 slide 29} \\
&= -\sum_x \sum_y P(x,y) log_2(P(x)) - \sum_x \sum_y P(x,y) log_2(P(y|x)) \quad \texttt{logarithm calculus} \\
&= -\sum_x P(x) log_2(P(x)) - \sum_x \sum_y P(x,y) log_2(P(y|x)) \quad \texttt{since } \sum_y P(x,y) = P(x) \\
&= H(X) + H(Y|X) \quad \texttt{Definition + Definition}
\end{aligned}
$$

Exercise 2

$A = \{\text{a,b, ... , z}\}$ i.e. $\forall \alpha \in A : P(A = \alpha) = \frac{1}{26}$

ä, ö, ü, ß
we would have 30 letters
4.9 bits

$$
\begin{aligned}
H(A) &= -\sum_{\alpha \in A} P(\alpha) log_2 P(\alpha) \\
&= -(26 \cdot \frac{1}{26} \cdot log_2(P(\alpha))) \\
&= -log_2 \frac{1}{26} \approx 4.7
\end{aligned}
$$

Log base = 2 i.e. unit = (average) bits

Exercise 3

The entropy $H(X)$ of a random variable X distributed on a density function p denotes how many bits are required to encode information that may appear for X. The probability values of the information tokens can be observed from a train set/corpus so we can setup a distribution estimation for p (e.g. as a language model). To test, whether distribution estimation is representative for the language, one can take another corpus as test set which might underlies a different distribution q. The cross entropy $H(p, q)$ denotes the amount of bits needed to encode tokens from the test distribution but using the probabilities of the language model created with the train set.

If the train set is very representative, the distributions p and q are very common, which results a cross entropy $H(p, q)$ that is very close to the general entropy $H(X)$. The more different the distributions q and p are, the larger is their cross entropy.

Computation in practice

First we consider two text corpora or divide one large corpus into a train and test set. For both sets, the probability for each word w_i of the considered vocabulary V gets estimated by counting and dividing by the text size. This results one estimated probability value for each corpus set. The cross entropy can than be computed with:

$$\sum_{w_i \in V} p_p(w_i) \cdot log(\frac{1}{p_q(w_i)})$$

where $p_p(x)$ and $p_q(x)$ denote the probabilities resulting from the different corpora considerations.

Reference: `https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/`

Exercise 4

What is smoothing and why do we need it?

Smoothing is a method to prevent unseen words or word sequences in the training corpus to be assigned a probability of zero. This is crucial, since a zero probability will prevent a model from choosing the word, even with a

definite tendency towards it. Furthermore, $log(0)$ is undefined so we should not take zero-probabilities into consideration.

Comment briefly on how to interpret the numbers from the results (entropy).

- War and Peace (en): 9.581291832304837
- Anna Karenina (en): 9.192985285349534
- Schuld und Suehne (de): 10.0884674455979

less predictable than english texts

As one can see from the values above the English texts have similar entropies while the German text shows a larger value. Schuld und Suehne takes on average about 10 bits/word to encode, while the English ones only need between 9.19 and 9.58. The difference can be explained with the morphological richness of the German language. To explain it with information density, the German language contains a higher viarity of words and therefore more information. There are more bits necessary to compress the German text without loss.

Comment on any difference in the results (Kullback-Leibler Divergence).

- War and Peace (en) + Anna Karenina (en): 0.9775732784878245
- War and Peace (en) + Schuld und Suehne (de): 10.124614683870634

The KullbackLeibler divergence measures how one probability distribution diverges from another one. Similar distributions will result in a divergence close to zero, while differing ones let increase the size of the KL-divergence. This can be seen in the difference between the two English and the English-German corpora. The two English ones have quite a low Relative Entropy of 0.9775732784878245 , while comparing two different language corpora lead to a value of 10.124614683870634. This can be explained with the big difference and the little overlapping within the different languages.

Exercise 5

The Lagrange multiplier allows us to find optimal (minimum/maximum) vales in a function respecting a certain condition. In the lecture, we used the Lagrange multiplier to find the minimal codeword length for a language with different word occurrence probabilities which is also prefix free. The optimization technique assigns a vector to lambda. It can be used to get the optimal values for the given function.

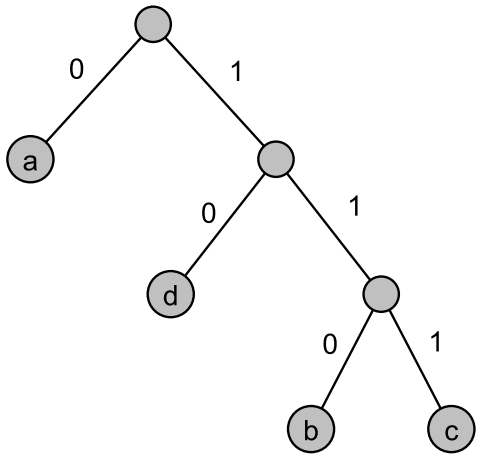
Exercise 6

a)

For the Huffman encoding, we first count the character occurrences in our sample text:

word	frequency
a	6
b	3
c	2
d	4

We sort the characters according their frequencies and setup the following Huffman tree.



Out of this we receive the following encodings:

word	frequency
a	0
b	110
c	111
d	10

which we can use to encode the input string to...

00011001111010110011001111010

b)

The optimal encoding length is

$$\begin{aligned}
&\sum_{w_i \in \{a,a,..,c,d,d\}} -log_2(p(w_i)) \cdot p(w_i) \\
&= \left(-log_2(\frac{6}{15}) \cdot \frac{6}{15}\right) \cdot 6 \\
&+ \left(-log_2(\frac{4}{15}) \cdot \frac{4}{15}\right) \cdot 4 \\
&+ \left(-log_2(\frac{3}{15}) \cdot \frac{3}{15}\right) \cdot 3 \\
&+ \left(-log_2(\frac{2}{15}) \cdot \frac{2}{15}\right) \cdot 2 \\
&= 7.37
\end{aligned}$$

optimal length $l_i = -log_2(p(w_i))$ of a symbol i

encoding should be prefix free!

$\Rightarrow$ Huffman encoding is optimal but not unique