UNIVERSITÄT
DES
SAARLANDES

Statistical Natural Language Processing

Summer Semester 2019
Prof. Dr. Dietrich Klakow

Exercise Sheet III

Submission Deadline: May 17th, 23:59

1 Information Theory

Exercise 1.1: Entropy and Probability Distributions (8 points)

Consider two discrete random variables X and Y with the following joint distribution

	$X = 1$	$X = 2$	$X = 3$	$X = 4$
$Y = 1$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{32}$
$Y = 2$	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{32}$	$\frac{1}{32}$
$Y = 3$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
$Y = 4$	$\frac{1}{4}$	0	0	0

(a) Calculate the marginal distribution of X and the marginal distribution of Y.

(b) What is the entropy $H(X)$ and $H(Y)$ in bits?

(c) What is $H(X|Y)$, $H(Y|X)$, and $H(X,Y)$ in bits?

(d) Calculate the mutual information $I(X;Y)$ and verify its symmetry property with your findings in (b) and (c).

You are expected to write all intermediate steps in clear notations and highlight the final answers.

2 N-gram Language Models

A word n-gram is a sequence of n adjacent words. In this exercise, you will train and evaluate n-gram language models on text corpora of contemporary English. Download the file `exercise3_corpora.zip` from the course materials and unpack it in a separate directory. The text in the file has been processed to contain only one sentence per line. For this exercise, you are required to use the tokenizer provided in the supplementary code. The code also contains the class `ngram_LM` which is meant as blueprint for a language model and you will implement a few methods and extend the code gradually starting from this exercise.

Exercise 2.1: Language Model Training (6 points)

Training an n-gram language model (hereafter LM) is simply collecting n-grams occurrence statistics from a large text corpus. In this exercise you will build a unigram LM and a bigram LM as follows

(a) Implement a Python function `word_ngrams` that takes as an input a sentence and an integer n and returns a list of n-grams in the sentence as tuples. The function should behave as follows

```
> word_ngrams('hello world', 1)
[('hello',), ('world',)]
```

and ...

```
> word_ngrams('hello world', 2)
[('<s>', 'hello'), ('hello', 'world'), ('world', '</s>')]
```

You should use the provided tokenization method and you can integrate `nltk.ngrams` functionality within `word_ngrams` function. However, primitive Python solutions are highly recommended as they are computationally much more efficient.

(b) With the funciton `word_ngrams`, collect unigram and bigram counts from the training text corpus `corpus.sent.en.train` and use a suitable data structure to store these counts, preferably `collections.Counter` objects. Generate a table of the top 15 frequent unigrams and bigrams with their relative frequencies. Can you tell the genre of this corpus from the table?

(c) Construct the vocabulary $\mathcal{V}$ of the LM from the words that occur at least once in the corpus. For a bigram LM, the vocabulary should include sentence padding tokens `'<s>'` and `'</s>'`. What is the type-token ratio of this corpus?

(d) In a unigram LM, the probability of a word is estimated as

$$\mathbf{P}(w) = \frac{N(w)}{\sum_{w' \in \mathcal{V}} N(w')} = \frac{N(w)}{N(\cdot)}$$

and for a bigram LM, the probability of a word conditioned on a predecessor word w_{-1} is estimated as

$$\mathbf{P}(w|w_{-1}) = \frac{N(w_{-1}, w)}{\sum_{w' \in \mathcal{V}} N(w_{-1}, w')} = \frac{N(w_{-1}, w)}{N(w_{-1})}$$

Implement the `estimate_prob` function in the provided code. The function takes as an input an n-gram history and a word, and returns as an output the probability of the word given the history. For a unigram LM, consider the empty string as the n-gram history.

(e) Using the class `ngram_LM`, create a unigram LM and a bigram LM. Examine the `test_LM` function and understand what it is supposed to do. Then test the correctness of your implementation by calling `test_LM` to check whether the probability mass for each LM sums up to 1.

(f) With the bigram model, investigate the properties of the training text corpus by observing the 10 most likely words to occur after each of the following word contexts

w_{-1} = `'blue'`	w_{-1} = `'green'`	w_{-1} = `'white'`	w_{-1} = `'black'`
w_{-1} = `'natural'`	w_{-1} = `'artificial'`	w_{-1} = `'global'`	w_{-1} = `'domestic'`

Your solution should contain the source code, a single table for top frequent unigrams and bigrams in part (b), a table of part (f), and any observations you want to include.

Exercise 2.2: Language Model Smoothing and Evaluation (6 points)

Zero-valued probabilities given by LMs due to unseen n-grams in the training corpus can be problematic for many NLP applications. Thus, the probability distributions estimated by LMs are 'smoothed' in order to prevent zero-valued probabilities. A simple smoothing method is the so-called Lidstone smoothing (a.k.a additive smoothing) which is used as follows

$$\mathbf{P}(w) = \frac{\alpha + N(w)}{\alpha V + \sum_{w' \in \mathcal{V}} N(w')} = \frac{\alpha + N(w)}{\alpha V + N(\cdot)}$$

$$\mathbf{P}(w|w_{-1}) = \frac{\alpha + N(w_{-1}, w)}{\alpha V + \sum_{w' \in \mathcal{V}} N(w_{-1}, w')} = \frac{\alpha + N(w_{-1}, w)}{\alpha V + N(w_{-1})}$$

where V is the size of the vocabulary and $0 < \alpha \leq 1$.

(a) Implement this smoothing technique in the function `estimate_smoothed_prob` which also takes the value of α as an input.

(b) Write a `test_smoothed_prob` function to check whether the probability mass of the distribution sums up to 1 when Lidstone smoothing is applied with $\alpha = 0.5$.

(c) Language models are used in statistical machine translation (MT) to assess the fluency of each hypothesised translation. For example, an MT system could propose three translations for the German sentence below

DE sentence	Gestern war ich zu Hause.
EN hypothesis 1	Yesterday was I at home.
EN hypothesis 2	Yesterday I was at home.
EN hypothesis 3	I was at home yesterday.

In this scenario, a bigram LM can be used to score each hypothesised sentence $w_0, w_1, \ldots, w_M$, where $w_0 =$ `<s>` and $w_M =$ `</s>`, as follows

$$\text{score}(w_0, w_1, \ldots, w_M) = -\frac{1}{M} \sum_{i=1}^{M} log\mathbf{P}(w_i|w_{i-1})$$

This score is sometimes referred to as average word surprisal, and word sequences with lower average surprisal should be preferred. In your code, implement the scoring functionality (with the smoothed probabilities) and assess the fluency of each hypothesised translation above. Make sure to apply the same preprocessing for the training corpus on these three sentences. Use the bigram counts and vocabulary in Exercise 2.1 to train the language model. Discuss your findings and highlight the limitations of bigram language models to assess the fluency of a given sentence.

Your solution should include the source code and an adequate discussion for part (c).

Submission Instructions

The following instructions are mandatory. Please read them carefully. If you do not follow these instructions, the tutors can decide not to grade your exercise solutions.

- You have to submit the solutions of this exercise sheet as a team of 2-3 students.
- Except for the module `nltk.ngrams`, NLTK modules are not allowed, and not necessary, for this assignment.
- You do not need to include the distributed corpora within your submission.
- Make a single `ZIP` archive file of your solution with the following structure

  - A source_code directory that contains your well-documented source code and a README file with instructions to run the code and reproduce the results.
  - A PDF report with your solutions, figures, and discussions on the questions that you would like to include.
  - A README file with group member names, matriculation numbers and emails.

- Rename your ZIP submission file in the format

  `exercise03_id#1_id#2_id#3.zip`

  where id#n is the matriculation number of every member in the team.

- Your exercise solution must be uploaded by only one of your team members to the course management system (CMS). Once the grading is done, your assignment grade will be distributed to each team member by one of the tutors.

- If you have any problems with the submission, contact babdullah@lsv.uni-saarland.de before the deadline.