Exercise Sheet IV

Submission Deadline: May 24th, 23:59

1 Kullback–Leibler Divergence

Exercise 1.1: Phonemes and Language Similarity (5 points)

Words in natural languages can be represented as a sequence of phonemes. For example, the German word {`Schrank`} is a sequence of five phonemes /ʃraŋk/. In this exercise, you will work on a small multilingual parallel corpus of phonetic transcriptions and investigate how Kullback–Leibler divergence can be used to measure similarity between languages based on their phoneme distributions. Each text file in the corpus has been transcribed into IPA phonetic symbols. The languages are British English, French, Spanish, and Italian. You will find this data in the `ipa_corpus` directory within the file `exercise4_corpora`.

(a) From all the languages in the multilingual corpus, construct a unified phoneme set Φ. Based on the phoneme counts of each language, create a probability distribution for each language over the phoneme set Φ, preferably as different `collections.defaultdict` objects. Apply Lidstone smoothing with $\alpha = 1$ to deal with unseen phonemes and write test cases to check whether your code is correct.

(b) Plot each phoneme distribution as a bar chart where the x-axis represents the phonemes and y-axis represents the probability (or relative frequency in this case). What do you observe?

(c) Write a function `KL_divergence` that takes two probability distributions and the sample space as inputs and returns the value of the KL-divergence in bits as an output. The function should implement the following formula

$$D(\mathbf{P}||\mathbf{Q}) = \sum_{x \in \mathcal{X}} \mathbf{P}(X = x) \log \frac{\mathbf{P}(X = x)}{\mathbf{Q}(X = x)}$$

(d) Using the phoneme distributions of part (a), compute the KL-divergence for each possible language pair and generate a 4 × 4 table to show the results. Your table should reflect the asymmetric property of the KL-divergence metric. Briefly explain your findings.

You solution should include the plots for part (b), a table for part (d), as well as the source code to reproduce your results.

2 Language Models Evaluation

In this exercise, you will evaluate the unigram and bigram LMs developed in the previous exercise sheet based on the corpus `corpus.sent.en.train` and will extend the functionality of the Python's class `ngram_LM`. The text files required for this exercise are in the `lm_eval` directory within `exercise4_corpora` in the course material. You will use the evaluation corpora to evaluate the LMs in the following exercises.

Exercise 2.1: LM Evaluation - Perplexity (10 points)

Language models are used to estimate the probability of a sequence of word tokens. Thus, an intrinsic evaluation metric is the likelihood that the LM assigns to held-out corpus which was not used in training. For an n-gram LM, the likelihood of a held-out test corpus $\mathbf{T}$ is

$$\mathcal{L}(\mathbf{T}) = \sum_{i=1}^{M} log\mathbf{P}(w_i|w_{i-n+1}, \dots, w_{i-1})$$

The perplexity of the held-out corpus is the following quantity

$$\text{perplexity}(\mathbf{T}) = 2^{-\frac{\mathcal{L}(\mathbf{T})}{M}}$$

(a) Assume that words of a vocabulary $\mathcal{V}$ are drawn from a uniform distribution. Given a test corpus of length M where each word token is in $\mathcal{V}$, what is the perplexity of this corpus?

(b) Implement a function `perplexity` that takes a test corpus and smoothing factor α as inputs and returns the perplexity. The function should expect the test corpus to be represented as a sequence of sentences and each sentence is a sequence of n-grams (with the padding symbols {`<s>`, `</s>`} when necessary).

(c) With the `perplexity` function in (b), compute the perplexity of the two test corpora for this exercise: `simple.test` and `wiki.test`. For each test corpus, report the perplexity quantities obtained by the unigram LM (with both unsmoothed and smoothed probabilities) and the bigram LM (with both unsmoothed and smoothed probabilities). Use Lidstone smoothing with $\alpha = 0.2$ for the smoothed probabilities. Discuss your observation.

(d) Examine the two test corpora and closely observe the sentence structure of a few example sentences. Calculate the percentage of unseen unigrams and bigrams in both corpora and relate your findings to the perplexity values computed in part (c).

(e) Examine the impact of the smoothing factor α on the perplexity of the test corpora with the unigram LM and the bigram LM. For each $\alpha \in \{0.1, 0.2, \dots, 1.0\}$, compute the perplexity values and plot a linear line chart for each test corpus where the x-axis represents α and the y-axis represents the perplexity. Discuss your observations.

(f) In the textbook of Manning and Schütze (sections 2.2.7 and 2.2.8), the perplexity metric is described from an information-theoretic point of view. Briefly explain the difference between the perplexity definition in the textbook and the definition provided in this exercise sheet.

Your solution should include a table of the perplexity values of part (c), a table of the unseen n-grams of part (d), a figure of part (e), a sufficient, but brief discussion for parts (c), (d), (e), and (f), and the source code to reproduce your results.

Exercise 2.2: LM Evaluation - Grammatically Assessment (5 points)

Although n-gram LMs are not trained to model the internal syntactic structure of natural languages, the sequential order of words encodes some structural aspects of the language which LMs can capture. In this exercise, you will conduct a qualitative analysis and investigate whether LMs can detect unconventional English sentence structure. To this end, you are given a small data set that consists of popular quotations from *Star Wars* iconic character; *Yoda*. For example, consider the following *Yoda-ish* sentence

" Powerful you have become, the dark side I sense in you. "

Contrary to standard English, which follows a subject-verb-object (SVO) order, Yoda's speech follows a non-standard word order, either OSV or XSV, where X is the phrase that complements the verb of the sentence. In this data set, each Yoda-ish sentence is aligned to its standard English counterpart. You will find this data in the `lm_eval` directory within the file `exercise4_corpora`.

(a) With the bigram LM and the `score_sentence` function you developed for the previous exercise sheet, compute the score of each Yoda-ish sentence and its English equivalent. Generate a table of the scores for each pair. Write your observations.

(b) Based on your observations from the table, select two pairs; one where the LM score difference was the largest, and the other where the LM score difference was the smallest. For each sentence in each pair, plot a line chart where the x-axis represents the words (in their original order with `<s>` at $x = 0$) and the y-axis represents the negative log probability of the word (with base 2) assigned by the bigram LM. Briefly explain your findings.

(c) Did the bigram LM do well on this task? Justify why or why not.

(d) Can the unigram LM be used for this task? Explain why or why not in a few sentences.

(e) If your mother tongue is not English and you were asked to re-write the *Start Wars* series in your own language, explain in a few sentences how would you structure Yoda's speech and provide a few examples if necessary. (*Note: this is an optional question*)

Your solution should include a table for part (a), four line charts for part (b), a convincing argument for part (c), as well as the source code to reproduce your results.

Submission Instructions

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

- You have to submit the solutions of this exercise sheet as a team of 2-3 students.
- 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 (`<2MB`) 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

  `exercise04_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.