Natural Language Processing - Ex1

Due: Sunday 13.11.2022 23:55

1 Theoretical (60 points)

1. (10 pts) Given a bigram language model for sentences of the form START w_1 w_2 w_3 $\cdots$ w_n STOP (where w_i for $1 \leq i \leq n$ is a word), show that if the transition probabilities are well-defined (i.e., sum up to 1) and each word has some non-zero probability for generating STOP ($\forall w, p(\text{STOP}|w) > 0$), then the sum of the probabilities over all finite sequences is 1.

   Hint: prove that the complement probability (i.e., the probability to never generate STOP, which is the same as the sum of all the sequences that don't have STOP) is 0.

2. (15 pts) We want to build a spelling corrector, focusing on the distinction between "where" and "were". Given a sentence as input, the corrector should predict the true spelling for each instance of "where" or "were" and correct the spelling in the case of mistake.

   For example, given the sentence "He went where there where more opportunities", the corrector should predict "where" for the first instance and "were" for the second one. It should also correct the word in the second case.

   Suppose we use a language model for this task. Given a language model $p(w_1, w_2, \cdots, w_n)$ where n is the length of the sentence, the corrector returns the spelling that gives the highest probability.

   In our example, the spelling corrector will output "were" for the second instance if:

   p(He went where there **were** more opportunities) $>$ p(He went where there **where** more opportunities)

   (a) Describe formally a unigram language model for the spelling corrector. Assume that the probability of a word is given by its proportion in the corpus (the training set) and that the number of instances in the corpus of each word in the vocabulary is strictly bigger than 0. Given the sentence "He went where there where more opportunities", under which conditions will the spelling corrector give a right answer for the first instance of "where"? for the second instance of "where"? for both instances?

   (b) Describe formally a bigram language model for the spelling corrector. Assume again that we estimate the parameters of the model using relative frequency and that the number of instances in the corpus of each word in the vocabulary is strictly bigger than 0. Why might this model be better that the model in (a)? Can a sentence in this model get a zero-probability? Would it be a problem for the model?

3. (15 pts) Consider the advanced smoothing method called Good-Turing smoothing. Let N_c be the number of word types (unique words) which appeared exactly c times in the training corpus (e.g., N_1 is the number of unique words that appeared one time in the training corpus). N denotes the total number of word instances in the training corpus. An estimate of the total probability of all unseen words (i.e., words that do not appear in the training corpus) is given by $p_{unseen} = \frac{N_1}{N}$.

The smoothed Good-Turing estimate of a frequency of a word that appears c times in the training corpus is $\frac{(c+1)N_{c+1}}{N_c \cdot N}$.
Note: Assume that $N_c > 0$ for all values of c up to a certain maximum value c_{max} and $N_c = 0$ for all $c > c_{max}$.

(a) Show that the sum of smoothed Good-Turing frequency estimates over all word types in the training corpus is $1 - p_{unseen}$

(b) Write down the equation for the smoothed Add-One estimate of a frequency of a word that appears c times in the training corpus. Show that there is a threshold μ, such that for all words of frequency less than μ, their smoothed estimate is higher than the MLE, and for all words of frequency more than μ, their smoothed estimate is lower than the MLE.

(c) Show that the property in (b) does not necessarily hold for the smoothed Good-Turing estimate.

4. (15 pts)

(a) Write down the equation for a trigram language model (without detailing the probability estimations). Which (conditional) independence assumption is made in the model?

(b) Give an example of an English sentence and a Hebrew sentence where the phenomenon of verb-subject agreement (see below) is captured by the model in (a). That is, give an example where the model in (a) is likely to predict the correct inflection of the verb, given the subject.

(c) Give an example of an English sentence and a Hebrew sentence where subject-verb agreement is not captured. Which n (for an n-gram model) is necessary for capturing this phenomenon in your example?

Note: Subject-verb agreement is the correspondence between the morphological inflection of the verb and the type of the subject. For instance, in English where the subject is singular, verbs in present tense end with an 's', while the base form is used for plural subjects. (e.g., "a dog bark**s**", "dogs bark")

5. (5 pts) Give an example of a sentence (in Hebrew or in English), where each consecutive pair of words is grammatically valid, but the sentence is not grammatically valid. Do the same with consecutive triplets and consecutive 4-tuples. You will note that doing this exercises for 4-tuples is considerably more difficult than for pairs. What does that indicate, in terms of the suitability of Markov models (of various orders) to be language models?
Note: A sequence of words is said to be *grammatically valid* in a language, if there is a grammatical sentence in that language that contains the sequence as a sub-string.

2 Practical (40 points)

In this part you will implement simple unigram and bigram language models and an interpolation between then. You will train the models with text from Wikidata.

Required Packages: In order to carry out the exercise, you will need to install the *spacy* package for language processing and Huggingface's *datasets* package for acquiring the data.

- You can find more details on the *spacy* website, where the two most important data structures are *doc* (for a document) and *token*:

  `https://spacy.io/api/doc`
  `https://spacy.io/api/token`

- You will also need the default English model of SpaCy called *"en_core_web_sm"*. See instructions here:

  `https://spacy.io/usage/models`

- For additional information on the datasets package see -
  `https://huggingface.co/docs/datasets/index`
- You will use the wikitext-2-raw-v1 version and use the train set only (see code example).
- Code example:

```
import spacy
from datasets import load_dataset

nlp = spacy.load("en_core_web_sm")
dataset = load_dataset('wikitext', 'wikitext-2-raw-v1', split="train")
for text in dataset['text']:
    doc = nlp(text)
```

Principles:

- You will use lemmas instead of exact words. See the .lemma_in the *spacy token* documentation.
- You will use only words and not punctuation or numbers. This should be done with the .is_alpha property.
- We regard each line in the data as a separate document. The n-grams can continue across sentences (ignoring the period) but not across documents.
- For the bigram model, you will add a "START" token at the beginning of each document. This token will be used for the probability of the first actual word and not the "START" token itself. For both models, you will not add "STOP" tokens.
- All probabilities should be calculated and reported in log-space.
- The same principles hold for predictions. That is, the probability for a given sentence is the n-gram probability of its word's lemmas.

Tasks:

1. Train maximum-likelihood unigram and bigram language models based on the above training data.
2. Using the bigram model, continue the following sentence with the most probable word predicted by the model: " I have a house in ...".
3. Using the bigram model:
   (a) compute the probability of the following two sentences (for each sentence separately).
   (b) compute the perplexity of **both** the following two sentences (treating them as a single test set with 2 sentences).

   Brad Pitt was born in Oklahoma
   The actor was born in USA
4. Now we use linear interpolation smoothing between the bigram model and unigram model with $\lambda_{bigram} = 2/3$ and $\lambda_{unigram} = 1/3$, using the same training data. Given this new model, compute the probability and the perplexity of the same sentences such as in the previous question.

Submission: Submit a .pdf file with your answers to the questions, as well a .py or .ipynb file with your code.