Exercise Sheet VII

Submission Deadline: June26th, 23:59

Pruning Language Models

1) **Probability Threshold Pruning (5 points)**
In this exercise, you will train a language model using absolute discount smoothing and pruning. Backoff n-gram models are typically trained on very large text corpora. An uncompressed LM is usually too large for practical use since all realistic applications have memory constraints. Therefore, LM pruning techniques are used to produce the smallest model while keeping the performance loss as small as possible. You can implement this by extending your code from assignment 6.

   (a) Pre-process the corpus English_train.txt by applying white-space tokenization and removing punctuation. You can use the `tokenize` function provided in the script.

   (b) Implement a method `prune`, which takes epsilon as an argument, and creates and returns a new absolute discounting model with only the bigrams and unigrams with a probability of at least epsilon in the original model. You can use your code from assignment 6.

   (c) Compute the perplexity on the English_test.txt file for the original model without pruning from assignment 6 and the pruned models for $\epsilon = 10^{-n}$ for $n = 3, 4, 5, 6$. Use $d = 0.9$ for all models. Report the number of parameters for all models (the number of bigrams and unigrams) in a table along with the perplexity values.

   (d) What do you observe when looking at the perplexity and model sizes?

   Your solution should contain a table for part (c), a sufficient explanation of the findings in this exercise, as well as the source code to reproduce the results.

Class-Based Language Models

1) **Incorporating POS-Tagging for Language Models (5 points)**
In this exercise, you will explore the use of word classes for language models. Words can be grouped into classes, and these classes can be used as a feature for estimating the ngram probabilities rather than the word identities. In this exercise, the classes will be POS-tags. The bigram probability is estimated by summing over all POS possibilities as shown below:

$$P(w|w_{-1}) = \sum_{P_{1,i}} P(w|POS_i, w_{-1})P(POS_i|w_{-1}) \tag{1}$$

Furthermore, the following two assumptions are made to simplify the context

$$P(w|POS_i, w_{-1}) = P(w|POS_i) \tag{2}$$

$$P(POS_i|w_{-1}) = P(POS_i|POS_{-1}) \tag{3}$$

The POS-tag probabilities are estimated as follows:

$$P(w|POS) = \frac{N(w, POS)}{N(POS)} \tag{4}$$

$$P(POS|POS_{-1}) = \frac{N(POS_{-1}, POS)}{N(POS_{-1})} \tag{5}$$

(a) Download the Penn Treebank Corpus from nltk with `nltk.download('treebank')`. You may use the corpus reader functions such as `words(), tagged_words()`. The `tagged_words()` words function returns the corpus with tagged words in the format (word, POS-tag). Pre-process the corpus by lower-casing the words and deleting the words that belong to the following POS-tag classes: ("CD", "LS", "SENT", "SYM", "#", "$", "“", "“", "-LRB-", "-RRB-", ",", ":", ".", "-NONE-", "”"). Split the corpus into 80% for training and 20% for testing.

(b) Estimate the bigram probabilities of your training corpus using the above formulas with absolute discounting smoothing and $d = 0.9$. For $P(w|POS)$:

$$P_{abs}(w|POS) = \frac{max\{N(w, POS) - d, 0\}}{N(POS)} + \lambda(POS)P_{abs}(w) \tag{6}$$

$$P_{abs}(w) = \frac{max\{N(w) - d, 0\}}{N} + \lambda(.)P_{unif}(w) \tag{7}$$

with

$$\lambda(POS) = \frac{d}{N(POS))}N_{1+}(POS\bullet) \tag{8}$$

where

$$N_{1+}(POS\bullet) = |\{v : N(v, POS) > 0\}| \tag{9}$$

and

$$\lambda(.) = \frac{d}{N}N_{1+} \tag{10}$$

where

$$N_{1+} = |\{v : N(v) > 0\}| \tag{11}$$

When applying absolute discounting smoothing on $P(POS|POS_{-1})$, consider the following:

$$P_{abs}(POS|POS_{-1}) = \frac{max\{N(POS_{-1}, POS) - d, 0\}}{N(POS_{-1})} + \lambda(POS_{-1})P_{abs}(POS) \tag{12}$$

$$P_{abs}(POS) = \frac{max\{N(POS) - d, 0\}}{N} + \lambda(.)P_{unif}(POS) \tag{13}$$

with

$$\lambda(POS_{-1}) = \frac{d}{N(POS_{-1})}N_{1+}(POS_{-1}\bullet) \tag{14}$$

where

$$N_{1+}(POS_{-1}\bullet) = |\{POS_i : N(POS_{-1}, POS_i) > 0\}| \tag{15}$$

and

$$\lambda(.) = \frac{d}{N} N_{1+} \tag{16}$$

where

$$N_{1+} = |\{POS_i : N(POS_i) > 0\}| \tag{17}$$

(c) Compute the perplexity on your test corpus.

(d) Estimate the bigram probabilities using a language model with absolute discounting smoothing and without POS-tags. You can use your implementation from assignment 3. Compute the perplexity on the test corpus.

(e) Compare the perplexities. How do you explain the difference in model performances?

Your solution should contain the perplexity values, a sufficient explanation of the findings in this exercise, as well as the source code to reproduce the 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 students.
- If you submit source code along with your assignment, please use Python unless otherwise agreed upon with your tutor.
- NLTK modules are not allowed, and not necessary, for the assignments unless otherwise specified.
- 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. You may also upload scans or photos of high quality.
  - A `README` file with group member names, matriculation numbers and emails.
- Rename your `ZIP` submission file in the format

  `exercise02_id#1_id#2.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 under *Assignments* in the *General* channel on Microsoft Teams.
- If you have any problems with the submission, contact your tutor before the deadline.