CSE 517: Homework 1
Language Models

Due Jan 20 Fri 11:59pm 2017

Submission instructions Submit 2 files on Canvas:

1 **Report:** (`HW1.pdf`): You will submit a typed report in the pdf format. The quality of report will be a very important aspect of grading. Key points to keep in mind are (1) clarity (neat organization with graphs and tables whenever applicable) and (2) insightful analysis and commentaries. When writing about *"error analysis"*, an important practice in any NLP project, try to make concrete arguments through examples. The report will typically be about 2-5 pages in total, but there will be no formal restriction on font sizes, page margins, or even number of pages.

2 **Code:** (`HW1.tgz`): Whenever applicable, you will also submit your code together with a neatly written README file to instruct how to run your code with different settings. We prefer either Python or Java code for compatibility reasons. Discuss with the teaching crew in advance if you must use some other programming language. We assume that you always follow good practice of coding (commenting, structuring) and we will not grade your code based on such qualities of code writing practice.

- *Late submission:* please check the class homepage for the late submission policy. These credits are to accommodate your conference trips, deadlines, and illness. There is really no need to contact the teaching crew to get the permission before using your free late submission credits. For some extremely unusual circumstances, we might grant you additional late day credits, but for most cases we will stick to the original policy so as to be fair with the rest of the class.

1 [Written] Smoothing (2 pts)

In class, we discussed how back off can improve the performance of language models. Consider the following back-off scheme. First, we define the sets

$$\begin{array}{lcl}\mathcal{A}(w_{i-1}) & = & \{w : c(w_{i-1}, w) > 0\} \\ \mathcal{B}(w_{i-1}) & = & \{w : c(w_{i-1}, w) = 0\} \\ \mathcal{A}(w_{i-2}, w_{i-1}) & = & \{w : c(w_{i-2}, w_{i-1}, w) > 0\} \\ \mathcal{B}(w_{i-2}, w_{i-1}) & = & \{w : c(w_{i-2}, w_{i-1}, w) = 0\}\end{array}$$

where c is a function that counts n-grams in the training set. For example, if the bigram "Honey Bunny" appears 22 times in the corpus, we will have $c(Honey, Bunny) = 22$.

Now, we can define a back-off trigram model:

$$p(w_i|w_{i-2}, w_{i-1}) = \begin{cases} p_1(w_i|w_{i-2}, w_{i-1}) & \text{If } w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\ p_2(w_i|w_{i-2}, w_{i-1}) & \text{If } w_i \in \mathcal{A}(w_{i-1}) \text{ and } w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \\ p_3(w_i|w_{i-2}, w_{i-1}) & \text{If } w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

Where:

$$\begin{aligned} p_1(w_i|w_{i-2},w_{i-1}) &= p_{ML}(w_i|w_{i-2},w_{i-1}) \\ p_2(w_i|w_{i-2},w_{i-1}) &= \frac{p_{ML}(w_i|w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2},w_{i-1})} p_{ML}(w|w_{i-1})} \\ p_3(w_i|w_{i-2},w_{i-1}) &= \frac{p_{ML}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} p_{ML}(w)} \end{aligned}$$

1.1 Deliverables

1. Does the above model form a valid probability distribution? Prove your answer. If it does not form a valid probability distribution, suggest how to make it one by modifying p_1, p_2 and p_3, using p_{ML} (the maximum likelihood estimate) and/or the count function c, and briefly explain why your modification works.

2 [Written] Language Model Classifier (2 pts)

One of the possible applications of language models, not covered in the lecture, is text categorization: given a document x as a sequence of words, assigning a category label y. Examples of text categorization include spam email detection $y \in \{spam, not\_spam\}$, news topic categorization $y \in \{sport, politics, science, ...\}$, authorship attribution $y \in \{shakespeare, woolf, pinker, ...\}$.

2.1 Deliverables

1. Sketch out how you can make use of language models for text categorization. Use equations whenever possible.

3 [Programming] Language Models (3 pts)

For the first part of this assignment, you will implement unigram, bigram, and trigram language models. Recall that the probability of seeing a sentence $\mathbf{s} = \{x_1, \ldots, x_n\}$ can be modeled as follows if using the trigram models without the special START symbols:

$$p(\mathbf{s}) = p(x_1 \ldots x_n) = q(x_1)q(x_2|x_1)\prod_{i=3}^{n} q(x_i \mid x_{i-2}, x_{i-1})$$

$q(x_i \mid x_{i-2}, x_{i-1})$ is the probability of seeing the current word x_i after seeing the bigram (x_{i-2}, x_{i-1}). Always assume that the last word is the special STOP symbol, i.e., x_n = STOP. Alternatively, if you like to include the special START symbols, you can instead do the following:

$$p(\mathbf{s}|\text{START}_2, \text{START}_1) = p(x_1 \ldots x_n|\text{START}_2, \text{START}_1) = \prod_{i=1}^{n} q(x_i \mid x_{i-2}, x_{i-1})$$

3.1 Dataset

For this part, you are provided with the following files:

- *brown.train.txt*: data to train your language model with (i.e. to estimate $q(x_i \mid x_{i-2}, x_{i-1})$)
- *brown.dev.txt*: development data for you to choose the best smoothing parameters
- *brown.test.txt*: test data for evaluating your language model

The data files are formatted with each line containing a tokenized sentence (white spaces mark token boundaries). Note that using the test set for training or finding the best parameters is considered cheating.

3.2 Perplexity

One way to evaluate the language model is computing the perplexity of the model on previously unseen data. Recall that computing the perplexity involves computing the average log probability of the entire corpus:

$$\text{perplexity} = 2^{-l}$$

where

$$l = \frac{1}{M}\sum_{i=1}^{m} \log_2 p(\mathbf{s}_i) = \frac{1}{M}\sum_{i=1}^{m} \log_2 p(x_{i1}, \dots x_{in_i})$$

and M is the total number of words in the text (i.e. the sum of lengths of all sentences).

A word of caution: You will primarily use *brown.dev.txt* as the previously unseen data while (i) developing and testing your code, (ii) trying out different design decisions (e.g., different ways of handling out-of-vocabulary words), (iii) tuning the hyperparameters (see problem # 2 below), and (iv) performing error analysis. For scientific integrity, it is extremely important that you will use the real test data *brown.test.txt* only once just before you report all the final results. Otherwise, you will start overfitting on the test set indirectly. Please don't be tempted to run the same experiment more than once on the test data.

3.3 Tips

When implementing the language model, there are several things to consider:

- Don't forget to handle the out-of-vocabulary (OOV) words by converting some of the low frequency words into UNK during training.
- You will need to include the special STOP word at the end of each sentence. It is optional if you include the special START words.
- Make sure that $q(x_i \mid x_{i-2}, x_{i-1})$ is a valid probability distribution. More concretely, each trigram probability conditioning on a different history must sum up to 1, e.g., $\sum_i q(x_i|\text{“the cat”}) = 1$ and $\sum_i q(x_i|\text{“the dog”}) = 1$.
- Be mindful of underflow! Code up in the log space, where $\log \prod_i q(x_i) = \sum_i \log q(x_i)$.

3.4 Deliverables

1. Report the perplexity scores of the unigram, bigram, trigram language models for your *training*, *dev* and *test* sets, and elaborate all your design choices (e.g., if you introduced START symbols in addition to STOP symbols, how you handled out-of-vocabulary words). Briefly discuss the experimental results.

4 [Programming] Smoothing (3 pts)

To make your language model work better, you will need to implement the following smoothing techniques:

4.1 Add-K Smoothing

$$p(x_i \mid x_{i-2}, x_{i-1}) = \frac{K + C(x_{i-2}, x_{i-1}, x_i)}{\sum_{x\in\mathcal{V}^*}(K + C(x_{i-2}, x_{i-1}, x))}$$

where $C(x_{i-2}, x_{i-1}, x_i)$ is the trigram count, and $\mathcal{V}^*$ is the set of your entire vocabulary (including special tokens such as OOV words and sentence boundaries).

4.2 Linear Interpolation

$$p'(x_i \mid x_{i-2}, x_{i-1}) = \lambda_1 \times p(x_i \mid x_{i-2}, x_{i-1}) + \lambda_2 \times p(x_i \mid x_{i-1}) + \lambda_3 \times p(x_i)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

4.3 Choosing Hyper-parameters

You can use the development data *brown.dev.txt* to decide the best values of the hyper-parameters (K, λ_1, λ_2, and λ_3). The fancier way of picking the hyper-parameters for linear interpolation is through Expectation-Maximization (EM), but you haven't learned it yet. For this homework, you will instead do a *"grid search"*, which means you will simply try a few combinations of "reasonable" values.

4.4 Deliverables

For this part of the homework, you will only work with trigram language models.

1. (a) Report perplexity scores on *training* and *dev* sets for various values of K (no interpolation). Try to cover a couple of orders of magnitude (e.g. $K = 10$, $K = 1$, 0.1, ...).
   (b) Similarly, report perplexity scores on *training* and *dev* sets for various values of λ_1, λ_2, λ_3 (no K smoothing). Report no more than 5 different values for your λ's.
   (c) Putting it all together, report perplexity on the *test* set, using different smoothing techniques and the corresponding hyper-parameters that you chose from the dev set. Specify those hyper-parameters.

2. If you use only half of the training data, would it in general increase or decrease the perplexity on the previously unseen data? Discuss the reason why.

3. If you convert all words that appeared less then 5 times as UNK (a special symbol for out-of-vocabulary words), would it in general increase or decrease the perplexity on the previously unseen data compared to an approach that converts only a fraction of words that appeared just once as UNK? Discuss the reason why.