

Suggested solution for Assignment 3

Teaching assistant: Wilson Tam (yct@cs.cmu.edu)

Due on:

Problem 1: LM estimation

In this problem, we estimate a statistical bigram language model $\Lambda = \{P(v|u)\}$ using Maximum Likelihood Estimation (MLE) given a training corpus $W = \{w_1, w_2 \dots w_N\}$ with vocabulary size V . With the 1st-order Markov assumption, we know that $P(w_i | w_{i-1} w_{i-2} \dots w_1) = P(w_i | w_{i-1})$. Assume there are no OOVs (out of vocabulary), i.e. all the words in the test data are covered by the training corpus W .

1. Derive the log likelihood of the training data given the model Λ , i.e. $L(W; \Lambda) = \log P(w_1, w_2 \dots w_N; \Lambda)$.

Solution: Let $w_0 = \langle s \rangle$ be the symbol of the start of a sentence. $\Lambda = \{P(v|u)\}$

$$\begin{aligned} L(W; \Lambda) &= \sum_{i=1}^N \log P(w_i | w_{i-1}) \\ &= \sum_{u,v} C(u, v) \log P(v|u) \end{aligned}$$

2. By differentiation, show that the MLE of $\hat{\Lambda}$ is given by:

$$\hat{P}(v|u) = \frac{C(u, v)}{C(u)} \quad (1)$$

where $C(u, v)$ denotes the word bigram count of (u, v) and $C(u)$ denotes the word unigram count of u . [Hint: You need to integrate the constraints $\sum_v P(v|u) = 1 \forall u$ into $L(W)$.]

Solution: We introduce Lagrange multipliers λ_u to integrate the probability constraints that $\sum_v P(v|u) = 1 \forall u$. The criteria to maximize is the log likelihood with Lagrange multipliers as follows:

$$\begin{aligned} L'(W; \Lambda) &= L(W; \Lambda) - \sum_u \lambda_u \left(\sum_v P(v|u) - 1 \right) \\ \frac{\partial}{\partial P(v|u)} L'(W; \Lambda) &= \frac{C(u, v)}{P(v|u)} - \lambda_u = 0 \\ \Rightarrow \hat{P}(v|u) &= \frac{C(u, v)}{\lambda_u} \end{aligned}$$

Use the probability constraint: $\sum_v P(v|u) = 1$, we can determine $\lambda_u = \sum_v C(u, v)$ which is equal to $C(u)$ by definition. Therefore, the MLE of $\hat{\Lambda}$ is $\hat{P}(v|u) = \frac{C(u, v)}{C(u)}$.

3. Show that a MLE bigram LM always performs better than a MLE unigram LM on the same training corpus W , i.e. prove that $\log P_{bg}(W) \geq \log P_{ug}(W)$ where bg and ug denotes the bigram and unigram LM respectively.

Solution:

$$\begin{aligned}
 \log P_{bg}(W) &= \sum_{u,v} C(u,v) \log P(v|u) \\
 \log P_{ug}(W) &= \sum_v C(v) \log P(v) = \sum_{u,v} C(u,v) \log P(v) \\
 \Rightarrow \log P_{bg}(W) - \log P_{ug}(W) &= \sum_{u,v} C(u,v) \log \frac{P(v|u)}{P(v)} \\
 &= \sum_{u,v} C(u,v) \log \frac{P(u,v)}{P(u)P(v)} \\
 &= N \cdot \sum_{u,v} \frac{C(u,v)}{N} \log \frac{P(u,v)}{P(u)P(v)} \\
 &= N \cdot \sum_{u,v} P(u,v) \log \frac{P(u,v)}{P(u)P(v)} \\
 &= N \cdot KL(P(u,v)||P(u)P(v)) \geq 0
 \end{aligned}$$

4. What is the major problem in Eqn 1?

The bigram LM returns zero probability on unseen word bigram (u,v).

5. One approach to fix the problem in Eqn 1 is interpolate a bigram LM with a unigram LM, i.e. $P_I(v|u) = \lambda \cdot P(v|u) + (1 - \lambda) \cdot P(v)$. The interpolated model has 3 components $\Lambda_I = \{P(v|u), P(v), \lambda\}$. The ML estimation is used to estimate P(v) in a similar fashion using the same training corpus W. Draw the interpolated model as a one-state HMM specifying the transition probability and emission probability.

Solution:

It is simply a one-state HMM with 2 mixtures $P(v)$ and $P(v|u)$.

6. Using the above results, write down the ML estimate of the interpolation weight λ .

Solution:

We can use the EM algorithm to reestimate the mixture weight $\lambda^{(t)}$ at iteration t based on the previous value $\lambda^{(t-1)}$.

$$\begin{aligned}
 \lambda^{(t)} &= \frac{\sum_{i=1}^N P^{(t-1)}(q_i = 0|W)}{N} \\
 \text{where } P^{(t-1)}(q_i = 0|W) &= \frac{\lambda^{(t-1)} P(w_i|w_{i-1})}{\lambda^{(t-1)} P(w_i|w_{i-1}) + (1 - \lambda^{(t-1)}) P(w_i)}
 \end{aligned}$$

where q_i denotes the mixture index chosen to generate the current word w_i . $q_i = 0$ means the bigram model is chosen.

7. Why did the interpolation scheme not work in this case? [Hint: you may want to show $\hat{\lambda} = 1$ at the fixed point (i.e. at the training convergence).] Suggest a simple method to avoid it.

Solution:

Use the result from 1.3, the bigram *training* likelihood is always better than the unigram *training* likelihood. Based on the MLE principle, λ should be converged to 1 in order to maximize the likelihood of the *training* data. Therefore, LM interpolation fails.

One way to avoid this is to use an independent development set to estimate λ .

Problem 2: LM smoothing

1. The LM estimation using the Laplace (add-one) smoothing can be expressed as follows:

$$\hat{P}(v) = \frac{C(v) + 1}{\sum_v (C(v) + 1)} \quad (2)$$

where each unigram count is at least one. Show that this estimate can be written as a linear LM interpolation as follows:

$$\hat{P}(v) = \lambda \frac{C(v)}{\sum_v C(v)} + (1 - \lambda) \frac{1}{V} \quad (3)$$

where V is the size of the vocabulary.

Solution:

$$\lambda = \frac{\sum_v C(v)}{V + \sum_v C(v)} \in (0, 1) \quad (4)$$

$$1 - \lambda = \frac{V}{V + \sum_v C(v)} \in (0, 1) \quad (5)$$

2. Argue that Laplace smoothing is a bad smoothing approach. [Hint: Consider the case when V is very large.]

Solution:

Clearly when V gets large, λ in Eqn 4 will tend to 0 and $1 - \lambda$ will tend to 1 meaning that the the interpolated LM will use the lower-order distribution (uniform distribution in this case). Thus, the predictive power of the LM will be weakened.

3. Show that the linear interpolation of a bigram LM and a unigram LM can be implemented as a backoff bigram LM. A backoff bigram LM is given as follows:

$$P_{bo}(v|u) = \begin{cases} f(v|u) & \text{if } (u,v) \text{ exists} \\ bo(u) \cdot P(v) & \text{otherwise} \end{cases} \quad (6)$$

where $f(v|u)$ is a discounted distribution with $\sum_{v:(u,v) \text{ exists}} f(v|u) < 1$.

Solution:

Recall that an interpolated LM with a bigram and a unigram distribution is represented as follows:

$$P_I(v|u) = \lambda(u)P(v|u) + (1 - \lambda(u))P(v) \quad (7)$$

where $P(v|u)$ and $P(v)$ are a unsmoothed bigram LM and a unsmoothed unigram LM respectively using MLE. Notice that $P(v|u)$ can return zero probability on unseen bigrams (u,v) .

In order to fit this into a backoff framework, we can set:

$$f(v|u) = \lambda(u)P(v|u) + (1 - \lambda(u))P(v) \quad (8)$$

$$bo(u) = 1 - \lambda(u) \quad (9)$$

4. Class-based LM is a common technique for LM smoothing. It is often represented as follows:

$$P(w_i|w_{i-1}) \approx P(w_i|c_i) \cdot P(c_i|c_{i-1}) \quad (10)$$

where each word maps into a single class. This assumption may not be reasonable when you want to allow a word to map into multiple classes with certain probability. Modify Eqn 10 to accommodate this change.

Solution:

We first use w_{i-1} to predict the class label c_{i-1} . Then predict the next class label c_i given c_{i-1} . Then predict the next word w_i given c_i :

$$P(w_i|w_{i-1}) = \sum_{c_{i-1}} \sum_{c_i} P(w_i|c_i) \cdot P(c_i|c_{i-1}) \cdot P(c_{i-1}|w_{i-1}) \quad (11)$$

If a word can only map to a single class, then $P(class(w)|w)$ equals 1 and $P(w|c)$ equals 0 when $c \neq class(w)$.

Problem 3: Computer Exercise

1. What does perplexity mean? Provide both a technical and an intuitive definition. How does reducing perplexity help with speech recognition? Can you think of a situation in which the task with lower perplexity is more difficult than the task with higher perplexity?

Solution:

Perplexity measures the “average” branching factor (geometric mean). Intuitively, it measures how many words can follow a given word on the average. Technically, it is 2^H where $H(W)$ is the entropy over the word sequence W .

Reducing the perplexity usually helps because this means that the uncertainty of word is lowered meaning that the task becomes easier from the LM point of view.

Perplexity reflects the text sources complexity from language models point of view, but it does not reflect the tasks difficulty in terms of acoustic usability. For example, the following word candidates of (in 0.1, an 0.3, and 0.7) have lower perplexity than the candidates of (January 0.3, corporate 0.4, and IBM 0.3), but the former is harder to recognize. An illustrative and practical example is letter dictation. In English the ee set has notoriously confusable acoustics. These are the letters b, c, d, e, g, p, t, v, z. The difficulty of recognition is greatly reduced by adopting the military sounds outs: bravo, charlie, delta, echo, golf, papa, tango, victor, zulu.

2. Build a statistical 3-gram LM using Good-Turing smoothing scheme with a training set. Find the perplexity of the LM on the heldout test sets.

You can download the SRILM toolkit at <http://www.speech.sri.com/projects/srilm/download.html>, and the data sets (swb_data.tgz) from the course webpage at <http://www.is.cs.cmu.edu/11-751/wiki>.

Training data: swb.train.100KW.text

Heldout test data: swb.heldout1, swb.heldout2, swb.heldout3

Report your trigram perplexity on these three test sets.

3. Look at the most frequent n-grams. Can you improve perplexity by manually collapsing some of the frequent n-grams into single words? (To do this, you need to modify the training and test sets using a script.)

Report top-20 most frequent n-grams (n=1,2,3).

Report your strategy and test result, including improvement, if any. If there is no improvement, explain.

Solution:

With that collapsing strategy, perplexity is increased because the vocabulary size is also increased meaning that the word prediction is harder in general.

4. Does a reverse (right-to-left) 3-gram LM perform better/equal/worse than a normal (left-to-right) 3-gram LM? Present the test perplexity results to support your intuition. (You need to reverse the training and test data using a script)

Solution:

On the average, a reverse N-gram LM performs slightly worse than a usual N-gram LM because of the difference in predicting the sentence-end symbol "`< s >`". Consider the following case:

`< s >` i am a boy `< /s >`

`< s >` boy a am i `< /s >`

Intuitively, the occurrence of "boy" and `< /s >` makes more sense than the occurrence of "i" and `< /s >`. Thus, the former has a better prediction of the sentence-end symbol.

5. Devise and try at least one strategy to reduce test set perplexity. (You will win extra credit if you try more than one strategies).

Solution:

This is an open ended problem with no definitive answer. Here are a couple ideas.

Approach 1: Class-based language models, i.e. clustering semantically similar words into one class, such as `DAY_OF_WEEK`, `DAY_OF_MONTH`, etc.

Approach 2: Change the cutoff value when training to avoid the problem of overfitting the language model.