Karlsruher Institut für Technologie

Karlsruhe Institute of Technology
Elizabeth Salesky
Institute for Anthropomatics and Robotics, Interactive Systems Labs
elizabeth.salesky@kit.edu

MT Praktikum - Language Modeling and Domain - Solutions

17. Juli 2018

Environment Setup

First we need to have access to our cluster enviroment.
If you can use the available machines in the pool room, please log in with username: smt[30-45] (any number between 30 to 45, for example smt35), password=123456.

If you use your own laptop, you can directly connect to the cluster using ssh, using this command:

```
ssh smt[30-45]@i13hpc1.ira.uka.de
```

Next, please log into i13hpc50, using the following commands:

```
ssh i13hpc1 (this is our login server; if you use your laptop, you are already here)
ssh i13hpc50
```

From there, go to your working directory:

```
cd /project/smtstud/ss18/systems/{username}/
```

Now enter the pre-installed virtual environment using these 2 commands:

```
bash
. /project/smtstud/ss18/commands/setup.sh
```

If you see the (praktikum) at the beginning of your terminal line, the setup was successful.

This directory contains today's data and scripts. Copy this directory into your working directory:

```
/project/smtstud/ss18/data/lmdom
```

Language Modeling
We are going to use n-gram language modeling to look at domain, an important consideration when training NMT models.

To train an order-n language model, use this command:
`lmplz -o {order} < {training_data} > {lm_name.arpa}`

To get the perplexity of a dataset using an arpa file, use this command:
`python perp.py {arpa_file} {text_data}`
(this may take a minute or so to execute, depending on the order of the LM and dataset)

- First, train a bigram LM on the English UDHR. Use this model `dev` data. What is the perplexity?
  613.3
- Now, train a bigram LM on the wikipedia data. Use this model to calculate the perplexity of the `dev` data. What is the perplexity?
  2455.3
- Finally, train a bigram LM on the novel chapter. Use this model to calculate the perplexity of the `dev` data.What is the perplexity?
  516.8
- How was the perplexity different with each of the models? Look at the first few lines of each training dataset, and the `dev` data. Why might this be?
  Different domains: bigrams have different frequencies in each of the datasets, and the dev data is most similar to the novel chapter.

  (Can you figure out which books they are?)
- How do you think this would change if you used a larger order (e.g. 5) LMs?
  It will go down considerably. Ex) wikipedia order-5 LM has perplexity of 1948 on `dev`.

  N-grams can be good models, but only if the test corpus looks like the training corpus. In reality, it often does not. We need to come up with adaptation and smoothing methods to account for this!
- Perplexity is often used as intrinsic evaluation for language models. Let's think about what these values mean more closely. Suppose a 'sentence' consists of random digits. What is the perplexity of this sentence, if our model assigns probability $p = 1/10$ to each digit?

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$

- Consider now a natural language sentence. What is the maximum perplexity of a sentence with 10 tokens? With 100 tokens?

  Maximum entropy / perplexity is when every word is different / has uniform probability. With 10, it is the same as above. With 100, it is 100.

- Let's return to our language models. Pick one of the three datasets, and train trigram and 4-gram LMs as well. Evaluate the perplexity of the `dev` data. How is it different between the bigram, trigram, and 4-gram models? Why might this be?
  The perplexity is higher with lower order n-gram models. With unigrams, each individual word occurs in more contexts: the individual word probability doesn't tell us so much. With bigrams, the probabilities are more specific to domain: we can better model 'the boy' with a bigram than $p(the) * p(boy)$. However, we see each n-gram fewer times, so overall probabilities are also lower.

- One major problem with models is generalization. If we have a bigram we have never seen before in `dev`, our model will produce a probability of 0 for the sentence and we can't compute perplexity (can't divide by 0!). Not good!

  To do something about this, people typically use smoothing methods. The simplest is called add-one or Laplace smoothing. This is as simple as it sounds: we increment the counts of all seen word types (unique) by 1, and the vocabulary by the same amount (size of the vocabulary, number of unique words).

  Now, there is a small probability alloted for unknown words: unseen n-grams have 1/N+V instead of 0!)

  $$P_{Laplace}(w_i) = \frac{count_i + 1}{N + V}$$

  A basic bigram language model has been coded for you in python, `bigram_lm.py`. Use this script to train a bigram LM on hpchapter1.txt. It will print the model entropy.
  3.052 bits

  In this last exercise, modify this script to use add-one smoothing. Now train a bigram LM. How has the entropy changed?
  The training data becomes less likely when we remove probability mass from observed bigrams. The more we smooth, the higher the entropy of the model. However, if we have n-grams we didn't observe in training in our test set, test data perplexity will go from zero to calculable.