Language Models

– handling unseen sequences

&

Information Theory

Problems with Unseen Sequences

Suppose we want to evaluate bigram models, and our test data contains the following sentence,

*"I couldn't submit my homework, because my **horse ate** it"*

Further suppose that our training data did not have the sequence "***horse ate***".

- What is the probability of $p(\text{“}ate\text{”} \mid \text{“}horse\text{”})$ according to our bigram model?
- What is the probability of the above sentence based on bigram approximation?

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

- Note that higher N-gram models suffer more from previously unseen word sequences (why?).

Laplace (Add-One) Smoothing

- "Hallucinate" additional training data in which each word occurs exactly once in every possible (N–1)-gram context

Bigram: $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

N-gram: $$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)+1}{C(w_{n-N+1}^{n-1})+V}$$

where V is the total number of possible words (i.e. the vocabulary size).

- Problem: tends to assign too much mass to unseen events.
- Alternative: add $0<\delta<1$ instead of 1 (normalized by δV instead of V).

Bigram counts – top: original counts; bottom: after adding one

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

bottom: normalized counts with respect to each row (each bigram prob)

Note that this table shows only 8 words out of V = 1446 words in BeRP corpus

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Smoothing Techniques

1. **Discounting**
2. **Back off**
3. **Interpolation**

1. Discounting

- Discounting – discount the probability mass of seen events, and redistribute the subtracted amount to unseen events
- Laplace (Add-One) Smoothing is a type of “discounting” method => simple, but doesn’t work well in practice
- Better discounting options are
  - **Good-Turing**
  - **Witten-Bell**
  - **Kneser-Ney**

  => Intuition: Use the count of events you’ve seen just once to help estimate the count of events you’ve never seen.

2. Backoff

- Only use lower-order model when data for higher-order model is unavailable (i.e. count is zero).
- Recursively back-off to weaker models until data is available.

$$P_{katz}(w_n \mid w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n \mid w_{n-N+1}^{n-1}) & \text{if } C(w_{n-N+1}^{n-1}) > 1 \\ \alpha(w_{n-N+1}^{n-1}) P_{katz}(w_n \mid w_{n-N+2}^{n-1}) & \text{otherwise} \end{cases}$$

Where P* is a discounted probability estimate to reserve mass for unseen events and α's are back-off weights (see text for details).

3. Interpolation

- Linearly combine estimates of N-gram models of increasing order.

Interpolated Trigram Model:

$$\hat{P}(w_n \mid w_{n-2,} w_{n-1}) = \lambda_1 P(w_n \mid w_{n-2,} w_{n-1}) + \lambda_2 P(w_n \mid w_{n-1}) + \lambda_3 P(w_n)$$

Where: $\sum_i \lambda_i = 1$

- Learn proper values for λ_i by training to (approximately) maximize the likelihood of a held-out dataset.

Smoothing Techniques

1. **Discounting**
2. **Back off**
3. **Interpolation**

- Advanced smoothing techniques are usually a mixture of [discounting + back off] or [discounting + interpolation]
- Popular choices are
  - Good-Turing discounting + Katz backoff
  - Kneser-Ney Smoothing: discounting + interpolation

OOV words: <UNK> word

- **Out Of Vocabulary** = OOV words
- Create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - L can be created as the set of words in the training data that occurred more than once.
    - At text normalization phase, any word not in L is changed to <UNK>
    - Now we train its probabilities like a normal word
  - At test/decoding time
    - If text input: Use UNK probabilities for any word not in training
- Difference between handling unseen sequences VS unseen words?

Class-Based N-Grams

- To deal with data sparsity
- Example:
  - Suppose LMs for a flight reservation system
  - Class: City = { Shanghai, London, Beijing, etc}

$$p(w_i|w_{i-1}) \approx p(c_i|c_{i-1})\, p(w_i|c_i)$$

- Classes can be manually specified, or automatically constructed via clustering algorithms
- Classes based on syntactic categories (such as part-of-speech tags – “noun”, “verb”, “adjective”, etc) do not seem to work as well as semantic classes.

Language Model Adaptation

- Language models are domain dependent
- Useful technique when we have a small amount of in-domain training data + a large amount of out-of-domain data
- Mix in-domain LM with out-of-domain LM
- Alternative to harvesting a large amount of data from the web: "web as a corpus (Keller and Lapata, 2003)"

  $\Rightarrow$ Approximate n-gram probabilities by "page counts" obtained from web search

$$p(w_3|w_1w_2) = \frac{c(w_1w_2w_3)}{c(w_1w_2)}$$

Long-Distance Information

- Many simple NLP approaches are based on short-distance information for their computational efficiency.
- Higher N-gram can incorporate longer distance information, but suffers from ***data sparsity.***
- Topic-based models
  - $p(w|h) = \sum_t p(w|t)p(t|h)$
  - Train a separate LM $p(w|t)$ for each topic t and mix them with weight $p(t|h)$, which indicates how likely each topic is given the history h.
- ***"Trigger"*** based language models
  - Condition on an additional word (trigger) outside the recent history (n-1 gram) that is likely to be related with the word to be predicted.

Practical Issues for Implementation

Always handle probabilities in log space!

- Avoid underflow
- Notice that multiplication in linear space becomes addition in log space => this is good, because addition is faster than multiplication!

$$p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4)$$

Information Theory

Information Theory

- A branch of applied mathematics and electrical engineering
- Developed by Shannon to formalize fundamental limits on signal processing operations such as data compression for data communication and storage.
- We extremely briefly touch on the following topics that tend to appear often in NLP research
  1. **Entropy**
  2. **Cross Entropy**
  3. Relation between **Perplexity** and Cross Entropy
  4. **Mutual Information**

     -- and Point-wise Mutual Information (**PMI**)
  5. Kullback-Leibler Divergence (**KL Divergence)**

Entropy

- Entropy is a measure of the uncertainty associated with a random variable.
- Higher entropy = more uncertain / harder to predict
- Entropy of random variable X
  - $H(X) = -\sum_x p(x) log_2 p(x)$
  - This quantity tells the expected (average) number of bits to encode a certain information in the optimal coding scheme!
- Example: How to encode IDs for 8 horses in binary bits?

How to encode IDs for 8 horses in bits?

- $H(X) = -\sum_x p(x) log_2 p(x)$
  - Random variable X represent the horse-id
  - P(x) represent the probability of horse-x to appear
  - H(X) indicates the expected (average) number of bits required to encode the ID of horses based on the optimal coding scheme.

- Suppose p(x) is uniform – each horse appears equally likely.
  - 001 for horse-1, 010 for horse-2, 011 for horse-3 etc

    ➔Need 3 bits
  - $H(x) = -\sum_{i=1}^{i=8} \frac{1}{8} log \frac{1}{8} = -\log \frac{1}{8}$ = 3 bits!

How to encode IDs for 8 horses in bits?

- $H(X) = -\sum_x p(x) log_2 p(x)$
  - Random variable X represent the horse-id
  - P(x) represent the probability of horse-x to appear
- Suppose p(x) is given as :

Horse 1	$\frac{1}{2}$	Horse 5	$\frac{1}{64}$
Horse 2	$\frac{1}{4}$	Horse 6	$\frac{1}{64}$
Horse 3	$\frac{1}{8}$	Horse 7	$\frac{1}{64}$
Horse 4	$\frac{1}{16}$	Horse 8	$\frac{1}{64}$

  - Use shorter encoding for frequently appearing horses, and longer encoding for rarely appearing horses
  - 0, 10, 110, 1110, 111100, 111101, 111110, 111111

    => Need 2 bits in average
  - $H(x) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{4}\log\frac{1}{4} - \frac{1}{8}\log\frac{1}{8}$ ... = 2 bits!

How about Entropy of Natural Langugage?

- Natural language can be viewed as a sequence of random variables (a stochastic process L), where each random variable corresponds to a word.
- After some equation rewriting based on certain theorems and assumptions (stationary and ergodic) about the stochastic process, we arrive at …

$$H(L) = \lim_{n \to \infty} -\frac{1}{n} \log p\,(w_1 w_2 \ldots w_n)$$

- Strictly speaking, we don't know what true $p\,(w_1 w_2 \ldots w_n)$ is, and we only have an estimate $q\,(w_1 w_2 \ldots w_n)$, which is based on our language models.

Cross Entropy

- Cross Entropy is used when we don't know the true probability distribution p that generated the observed data (natural language).
- Cross Entropy $H(p, q)$ provides an upper bound on the Entropy $H(p)$
- After some equation rewriting invoking certain theorems and assumptions…

$$H(p, q) = \lim_{n \to \infty} -\frac{1}{n} \log q\ (w_1 w_2 \ldots w_n)$$

- Note that the above formula is extremely similar to the formula we've seen in the previous slide for entropy. For this reason, people often use the term "entropy" to mean "cross entropy".

Relating Perplexity to Cross Entropy

- Recall Perplexity is defined as

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

- In fact, this quantity is the same as

$$PP(W) = 2^{H(W)}$$

Where H(W) is the cross entropy of the sequence W.

Mutual Information

- Mutual information measures the information shared by two random variables.

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x)\, p_2(y)} \right),$$

  - In other words, it measures how much knowing one of the variables reduces the uncertainty about the other.
  - If X and Y are independent variables, then mutual information is zero.
  - If X and Y are identical, then the mutual information is the same as the entropy of X (or Y)

Pointwise Mutual Information (PMI)

- Recall that Mutual Information is defined for random variables X and Y

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_1(x)\, p_2(y)} \right),$$

- In contrast, Pointwise mutual information – often called as PMI – is defined for specific values of X and Y

$$PMI(x,y) = \log \frac{Pr(X=x, Y=y)}{Pr(X=x)Pr(Y=y)}.$$

- When computed for a pair of words, PMI can measure the semantic relatedness of two words

  e.g.) PMI (“drink”, “beer”) > PMI (“drink”, “homework”)

Kullback-Leibler (KL) Divergence

- KL Divergence is a non-symmetric measure of the difference between two probability distributions P and Q.

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

Recap (Quiz)

- List three general techniques for smoothing, and explain each briefly.
- What is the problem with Laplace smoothing? what is another name for Laplace smoothing?
- Name two popular choices for smoothing
- what are two practical reasons to handle probabilities in log space?
- explain how to compute the entropy of natural language approximately.
- what is the relation between entropy and perplexity?
- how do you measure the distance between two probability distributions?
- how do you measure the semantic relatedness between two words?

Recap (Quiz)

- suggest how you'd measure PMI between (drink, beer) and (drink, homework) from the Wall Street Journal corpus (a collection of Wall Street Journal news articles)

- suggest how you'd measure which of the three corpora is the closest to the Wall Street Journal corpus:
  - Shakespeare corpus (consists of novels written by Shakespeare)
  - ACL Anthology corpus (consists of NLP papers)
  - TripAdvisor corpus (consists of tripadvisor's webpages)