Language Models

COSC 6336: Natural Language Processing
Spring 2018

UNIVERSITY of HOUSTON

Some content on these slides was borrowed from J&M

Language Models

- ★ They assign a probability to a sequence of words:
  - Machine Translation:
    - P(**high** winds tonite) > P(**large** winds tonite)
  - Spell Correction:
    - The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - Summarization, question-answering, OCR correction and many more!

More Formally

- Given a sequence of words predict the next one:
  - $P(w_5|w_1,w_2,w_3,w_4)$
- Predict the likelihood of a sequence of words:
  - $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$
- How do we compute these?

Chain Rule

- Recall the definition of conditional probabilities:
  - **$p(B|A) = P(A,B)/P(A)$** Rewriting: **$P(A,B) = P(A)P(B|A)$**
- More variables:
  - $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$
- The Chain Rule in General
  - $P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$

Back to sequence of words

P("I am the fire that burns against the cold") = P(I) x P(am|I) x P(the|I am) x P(fire|I am the) x P(that| I am the fire) x P(burns| I am the fire that) x P(against| I am the fire that burns) x P(the| I am the fire that burns against) x P(cold| I am the fire that burns against the)

- ★ How do we estimate these probabilities?

$$\frac{count(\text{I am the fire that burns against the cold})}{count(\text{I am the fire that burns against the})}$$

- ★ Any problems with this formulation?

We shorten the context (history)

Markov Assumption:

P(cold | I am the fire that burns against the) ≈ P (cold | burns against the)

This is: $P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$

When N = 1, this is a unigram language model:

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

When k =2, this is a bigram language model:

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

Count-based language models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
- because language has **long-distance dependencies**:
  - "The computer which I had just put into the machine room on the fifth floor crashed."
- But we can often get away with N-gram models

Estimating bigram probabilities

We rely on the Maximum Likelihood Estimate:

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

An example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $\quad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $\quad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $\quad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $\quad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

Raw bigram counts from the Berkeley restaurant project

	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

Raw bigram counts from the Berkeley restaurant project

Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

What kinds of knowledge?

- ★ P(english|want) = .0011
- ★ P(chinese|want) = .0065
- ★ P(to|want) = .66
- ★ P(eat | to) = .28
- ★ P(food | to) = 0
- ★ P(want | spend) = 0
- ★ P (i | <s>) = .25

Are all our problems solved?

Zeros

Training set:

… denied the allegations

… denied the reports

… denied the claims

… denied the request

Test set:

… denied the offer

… denied the loan

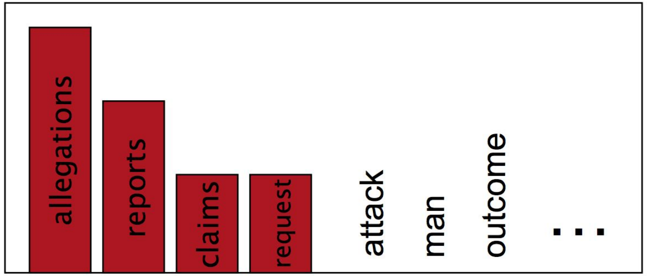
P("offer" | denied the) = 0

If there is a single bigram with prob 0 we will assign 0 prob to the entire test set!

Smoothing Intuition (taken from D. Klein)

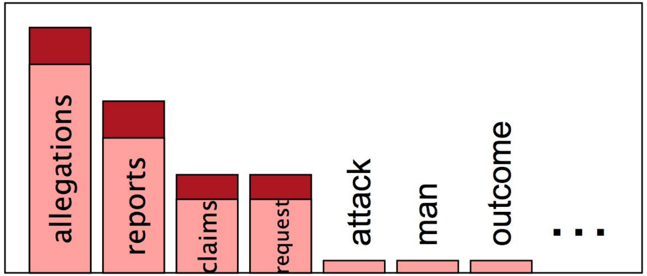
- When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Laplace Smoothing

★ Add one to all counts
★ Unigram counts:

$$P(w_i) = \frac{c_i}{N}$$

★ Laplace counts:

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

★ Disadvantages:
  ○ Drastic change in probability mass
★ But for some tasks Laplace smoothing is a reasonable choice

Leveraging Hierarchy of N-grams: Backoff

- Adding one to all counts is too drastic
- What about relying on shorter contexts?
  - Let's say we're tying to compute P(against | that burns), but count(that burns against) = 0
  - We backoff to a shorter context: P(against | that burns) ≈ P(against | burns)
- We backoff to a lower n-gram
- Katz Backoff (discounted backoff)

$$P_{\mathrm{BO}}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}), & \text{if } C(w_{n-N+1}^{n}) > 0 \\ \alpha(w_{n-N+1}^{n-1})P_{\mathrm{BO}}(w_n|w_{n-N+2}^{n-1}), & \text{otherwise.} \end{cases}$$

Leveraging Hierarchy of N-grams: Interpolation

★ Better idea: why not always rely on lower order N-grams?

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2 P(w_n|w_{n-1}) \\ &+\lambda_3 P(w_n)\end{aligned}$$

Subject to: $\sum_i \lambda_i = 1$

★ Even better, condition on context:

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1(w_{n-2}^{n-1}) P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2(w_{n-2}^{n-1}) P(w_n|w_{n-1}) \\ &+\lambda_3(w_{n-2}^{n-1}) P(w_n)\end{aligned}$$

Absolute Discounting

Bigram count in training	Bigram count in held out set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

For each bigram count in training data, what's the count in held out set?

Approx. a 0.75 difference!

Absolute Discounting

$$P_{\text{AbsoluteDiscounting}}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}v)} + \lambda(w_{i-1})P(w_i)$$

How much do we want to trust unigrams?

- Instead of P(w): “How likely is w”
- $P_{continuation}(w)$: “How likely is w to appear as a novel continuation?
- For each word, count the number of bigram types it completes

$$P_{\text{CONTINUATION}}(w) = \frac{|\{v : C(vw) > 0\}|}{|\{(u', w') : C(u'w') > 0\}|}$$

Interpolated Kneser-Ney

★ Intuition: Use estimate of $P_{CONTINUATION}(w_i)$

$$P_{\mathrm{KN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^{i}) - d, 0)}{\sum_v c_{KN}(w_{i-n+1}^{i-1}v)} + \lambda(w_{i-n+1}^{i-1})P_{KN}(w_i|w_{i-n+2}^{i-1})$$

Evaluating Language Models

- ★ Ideal: Evaluate on end task (extrinsic)
- ★ Intrinsic evaluation: use perplexity:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

- ★ Perplexity is inversely proportional to the probability of W

Lower perplexity is better

LMs trained on 38 million words and tested on 1.5 million words from WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Practical Issues

- We compute everything in log space:
  - Avoids issues with underflow
  - Faster

Advanced Language Models

- Discriminative models:
  - choose n-gram weights to improve a task, not to fit the training set
- Caching Models
  - Recently used words are more likely to appear

Advanced Language Models

★ Deep Learning for Language Models:
  - Neural Language Models have been quite the success lately (Bengio et al., 2003; Mikolov et al., 2010) in tasks such as speech recognition and machine translation

★ Although, Kneser-Ney has been shown to be competitive and even outperformed neural language models over smaller corpora (cf. Chen et al., 2016)