NGRAMS – PART 2
NATURAL LANGUAGE PROCESSING - CS 322.00

Blake Howald

September 25, 2017

AGENDA

- Presentation – Charlie Anderson, Ammar Babar
- Questions
- Ngrams – Part 2
  - Smoothing
  - Evaluation
    - Training/Testing
    - Perplexity

TRAIN/TEST CONSIDERATIONS

- Also attempt to get the largest sample of data possible:
  - Largest portion is for ***Training Set***
  - Next portion (no more than 1/3, typically) is for ***Test Set***
  - Smallest proportion for ***Development (Dev)*** and ***Holdout*** sets
    - Dev set (can be pulled from Test Set) used for the development of features
    - Holdout set used for additional information
  - Never train on Test, Dev or Holdout sets – artificially high results/ improper bias

NGRAMS - CALCULATION

Bigram	MLE	Bigram	MLE
have a	0.707	don't have	0.117
a box	0.589	I don't	0.117
box </s>	0.471	<s> I	0.589
I can't	0.471	can't be	0.471
to be	0.117	dumb </s>	0.117
so dumb	0.117	be so	0.333

- *I don't have a box*
- *P(I|<s>) x P(don't|I) x P(have|don't) x P(a|have) x P(box|a) x P(</s>|box)*
- *0.589x0.117 x 0.117 x 0.707 x 0.589 x 0.471 = 0.001581*
- *We don't have a box*

UNKNOWN WORDS

- There will be (should be) words in the test data that did not appear in the training data
  - Unless it is a *closed vocabulary* task, then there won't be any unknown words, but there still may be probabilities not estimated (we'll talk *smoothing* next).
  - In an *open vocabulary task*, you
    1. Decide on a vocabulary in advance (existing dictionary of English, e.g.)
    2. Any word(s) in the training set not in the preset vocabulary gets replaced with (<UNK> or <OOV>, e.g.)
    3. Estimate the probabilities for <UNK> or <OOV> like any other word
  - Good to track the rate of out of vocabulary (OOV) words in the test set generally.

SMOOTHING

- Multiple methods to address data sparsity (also known as discounting).
- Redistributing probability mass to unseen data
  - Laplace (add one) smoothing
    - Also add *k* smoothing
  - Good-Turing discounting
    - Kneser-Nay and other "absolute" discounting methods

LAPLACE (ADD ONE) SMOOTHING

The cat in the hat

	The	Cat	In	Hat
The	0	1	0	1
Cat	0	0	1	0
In	1	0	0	0
Hat	0	0	0	0

	The	Cat	In	Hat
The	0	.2	0	.2
Cat	0	0	.2	0
In	.2	0	0	0
Hat	0	0	0	0

	The	Cat	In	Hat
The	1	2	1	2
Cat	1	1	2	1
In	2	1	1	1
Hat	1	1	1	1

	The	Cat	In	Hat
The	0.05	.1	0.05	.1
Cat	0.05	0.05	.1	0.05
In	.1	0.05	0.05	0.05
Hat	0.05	0.05	0.05	0.05

GOOD TURING SMOOTHING

The cat in the hat

	The	Cat	In	Hat
The	0	1	0	1
Cat	2	1	3	2
In	4	1	0	0
Hat	1	1	2	1

- 4 bigrams with 0 counts
- 7 bigrams with 1 count
- 3 bigrams with 2 counts
- 1 bigram with 3 counts
- 1 bigram with 4 counts

	The	Cat	In	Hat
The	0	.05	0	.05
Cat	.1	.05	.15	.1
In	.2	.05	0	0
Hat	.05	.05	.1	.05

	The	Cat	In	Hat
The	.017	.85	.017	.85
Cat	.99	.85	…	.99
In	…	.85	.017	.017
Hat	.85	.85	.99	.85

- P* (zero counts) = 1 Counts/Total Counts
- P* (zero counts) = 7/20 = .35
- UPDATE COUNTS
  - 2 x 2 Counts / 1 Counts
  - 3 x 3 Counts / 2 Counts
  - …
  - (c+1)Nc+1/Nc

	The	Cat	In	Hat
The	.35	.042	.35	.042
Cat	.049	.042	…	.049
In	…	.042	.35	.35
Hat	.042	.042	.049	.042

INTERPOLATION & BACKOFF

- Interpolation (sum of λs = 1) - learned from the *Holdout Set*

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

- Katz (Good Turing) Backoff
  - Back off to increasingly shorter histories until you have an estimation based on observation.
  - Good Turing discounting applied for the lower order Ngram

KATZ (GOOD TURING) BACKOFF

The cat in the hat

"in cat"

- P* (zero counts) = 1 Counts/Total Counts
- P* (zero counts) = 1/20 = .0625
- UPDATE COUNTS
  - 2 x 2 Counts / 1 Counts

	The	Cat	In	Hat
The	0	1	0	1
Cat	0	0	1	0
In	1	0	0	0
Hat	0	0	0	0

	The	Cat	In	Hat
The	0	.2	0	.2
Cat	0	0	.2	0
In	.2	0	0	0
Hat	0	0	0	0

	The	Cat	In	Hat
The	0	1	0	1
Cat	0	0	1	0
In	.5	.003	0	0
Hat	0	0	0	0

	The	Cat	In	Hat
The	0	.2	0	.2
Cat	0	0	.2	.2
In	.025	.062	0	0
Hat	0	0	0	0

EVALUATION - PERPLEXITY

- Perplexity
  - Intrinsic evaluation
  - Sum all probabilities of test sentences raised to the – 1 / N (where N is the total number of word tokens encountered in the test set)
  - Lower the perplexity (good!) the higher the conditional probability – the better the model is at capturing sequence patterns
  - Perplexity typically lowers for higher order N-gram models

	Unigram	Bigram	Trigram
Perplexity	962	170	109

  - MUST exclude test data from the training data (values artificially high)
  - If (truly) random patterns – Perplexity will be (approach) N ((1/N)^-1)