ΓΕ 77 COMPUTATIONAL LINGUISTICS

Athanasios N. Karasimos

akarasimos@gmail.com

BA in Linguistics | School of English Language and Literature
National and Kapodistrian University of Athens

Lecture 4 | Wed 21 Mar 2018

LANGUAGE MODELLING WITH N-GRAMS

A short introduction

LANGUAGE MODELLING

Introduction to N-Grams

PROBABILISTIC LANGUAGE MODELS

- Today's goal: assign a probability to a sentence
  - Machine Translation:
    - P(**high** winds tonight) > P(**large** winds tonight)
  - 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, etc.

PROBABILISTIC LANGUAGE MODELING

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5\dots w_n)$

- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2\dots w_{n-1})$ is called a **language model**.

- Better: **the grammar** But **language model** or **LM** is standard

LANGUAGE MODELS & N-GRAMS

- Models that assign probabilities to sequences of words are called **language models** or LMs.
- The simplest model that assigns probabilities LM to sentences and sequences of words, the **N-gram**.
  - an N-gram is a sequence of *N* N-gram words: a 2-gram (or bigram) is a two-word sequence of words like "please turn", "turn your", or "your homework", and
  - a 3-gram (or trigram) is a three-word sequence of words like "please turn your", or "turn your homework".

HUNTING BIGRAMS AND TRIGRAMS

But our fish said,"No! No!
Make that cat go away!
Tell that Cat in the Hat
You do NOT want to play.
He should not be here.
He should not be about.
He should not be here
When your mother is out!"

"Have no fear!" said the cat.
"I will not let you fall.
I will hold you up high
As I stand on a ball.
With a book on one hand!
And a cup on my hat!
But that is not ALL I can do!'
Said the cat…

From Dr. Seuss *The Cat in the Hat.*

COMPUTING P(W)

- How to compute this joint probability:
  - P(its, water, is, so, transparent, that)
- Intuition: let’s rely on the Chain Rule of Probability

THE CHAIN RULE

- The definition of conditional probabilities

  $\mathbf{p(B|A) = P(A,B)/P(A)}$ Rewriting: $\mathbf{P(A,B) = P(A)P(B|A)}$

- 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

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

THE CHAIN RULE

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

HOW TO ESTIMATE THESE PROBABILITIES

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We’ll never see some data for estimating these.

THE MARKOV ASSUMPTION

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

THE MARKOV ASSUMPTION

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

UNIGRAM AND BIGRAM MODEL

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

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

N-GRAM 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(s) which I had just put into the machine room on the fifth floor is (are) crashing."
- But we can often get away with N-gram models

ESTIMATING N-GRAM PROBABILITIES

ESTIMATING BIGRAM PROBABILITIES

- The Maximum Likelihood Estimate

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

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

ESTIMATING BIGRAM PROBABILITIES: AN EXAMPLE

<s> I am Sam </s>

<s> Sam I am </s>

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

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

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

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

BIGRAM ESTIMATES OF SENTENCE PROBABILITIES

P(<s> I want to learn </s>) =

P(I|<s>)

× P(want|I)

× P(to|want)

× P(learn|to)

× P(</s>|learn)

= **X?**

ESTIMATING BIGRAM PROBABILITIES: A TASK

- **TASK II: Lets create some bigrams**
  - Subtask 1: Produce 5 sentences with 3-6 words (try to use several words at least twice)
  - Subtask 2: Give the P of 10 bigrams.
  - Subtask 3: Calculate the P of 2 sentences.

PRACTICAL ISSUES

- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

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

LANGUAGE MODELING TOOLKITS

- SRILM
  - http://www.speech.sri.com/projects/srilm/
- KenLM
  - https://kheafield.com/code/kenlm/
- Google N-Grams
  - https://books.google.com/ngrams
- N-Grams Data (COCA)
  - https://www.ngrams.info/

N-GRAMS PROBABILITIES ESTIMATION

Perplexity, Smoothing and Evaluation

EVALUATION: HOW GOOD IS OUR MODEL?

- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to "real" or "frequently observed" sentences
    - Than "ungrammatical" or "rarely observed" sentences?
- We train parameters of our model on a **training set**.
- We test the model's performance on data we haven't seen.
  - A **test set** is an unseen dataset that is different from our training set, totally unused.
  - An **evaluation metric** tells us how well our model does on the test set.

TRAINING ON THE TEST SET

- We can’t allow test sentences into the training set
- We will assign it an artificially high probability when we set it in the test set
- “Training on the test set”
- Bad science!
- And violates the honor code

EXTRINSIC EVALUATION OF N-GRAM MODELS

- Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - Compare accuracy for A and B

DIFFICULTY OF EXTRINSIC EVALUATION OF N-GRAM MODELS

- Extrinsic evaluation
  - Time-consuming; can take days or weeks
- So
  - Sometimes use **intrinsic** evaluation: **perplexity**
  - Bad approximation
    - unless the test data looks **just** like the training data
    - So **generally only useful in pilot experiments**
  - But is helpful to think about.

PERPLEXITY: A GAME TASK

Task 2: The Shannon Game

- How well can we predict the next word? Give 3 possible answer for each sentence (one should be phony/ unexpected).
  - I always order pizza with cheese and ____
  - The last ______ was a rainy day.
  - I like _____ (hobby).

  Unigrams are terrible at this game. (Why?)
- A better model of a text
  - is one which assigns a higher probability to the word that actually occurs

PERPLEXITY

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

Perplexity is the inverse probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

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

Chain rule:

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

For bigrams:

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

Minimizing perplexity is the same as maximizing probability

THE PERILS OF OVERFITTING

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - In real life, it often doesn’t
  - We need to train robust models that generalize!
  - One kind of generalization: Zeros!
    - Things that don’t ever occur in the training set
      - But occur in the test set

ZEROS

- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request

  P(“offer” | denied the) = 0

- Test set
  … denied the offer
  … denied the loan

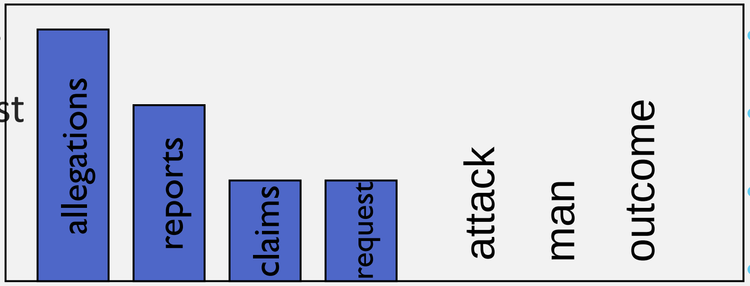
ZERO PROBABILITY BIGRAMS

- Bigrams with zero probability
  - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can't divide by 0)!
- ***So what is the solution?***

THE INTUITION OF SMOOTHING (FROM DAN KLEIN)

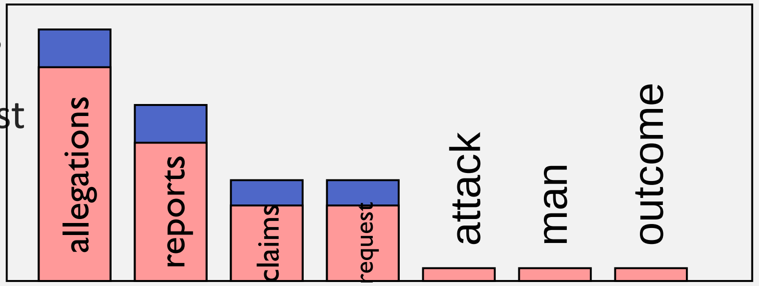
WHEN WE HAVE SPARSE STATISTICS

- 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



ADD-ONE ESTIMATION

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!
- MLE estimate:

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

- Add-1 estimate:

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

ADD-1 ESTIMATION IS A BLUNT INSTRUMENT

- So add-1 isn't used for N-grams:
  - We'll see better methods
- But add-1 is used to smooth other NLP models
  - For text classification
  - In domains where the number of zeros isn't so huge.

BACKOFF AND INTERPOLATION

- Sometimes it helps to use **less** context
  - Condition on less context for contexts you haven’t learned much about
- **Backoff:**
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- **Interpolation:**
  - mix unigram, bigram, trigram

- Interpolation works better

UNKNOWN WORDS: OPEN VERSUS CLOSED VOCABULARY TASKS

- If we know all the words in advanced
  - Vocabulary V is fixed
  - Closed vocabulary task
- Often we don’t know this
  - **Out Of Vocabulary** = OOV words
  - Open vocabulary task
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training

ADVANCED LANGUAGE MODELING

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

$$P_{CACHE}(w \mid history) = \lambda P(w_i \mid w_{i-2} w_{i-1}) + (1-\lambda)\frac{c(w \in history)}{\mid history \mid}$$

  - These perform very poorly for speech recognition (why?)

SUMMARY

- Language models offer a way to assign a probability to a sentence or other sequence of words, and to predict a word from preceding words.
- N-grams are Markov models that estimate words from a fixed window of previous words. N-gram probabilities can be estimated by counting in a corpus and normalizing (the maximum likelihood estimate).
- N-gram language models are evaluated extrinsically in some task, or intrinsically using perplexity. The perplexity of a test set according to a language model is the geometric mean of the inverse test set probability computed by the model.
- Smoothing algorithms provide a more sophisticated way to estimate the probability of N-grams. Commonly used smoothing algorithms for N-grams rely on lower-order N-gram counts through backoff or interpolation.

READING

- Jurafsky D. & J. Martin (2017). SPEECH and LANGUAGE PROCESSING
  An introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (3rd Edition). CHAPTER 4.