N-Gram Language Modeling

Mausam

(Based on slides of Michael Collins, Dan Jurafsky, Dan Klein, Chris Manning, Luke Zettlemoyer)

Outline

- Motivation
- Task Definition
- N-Gram Probability Estimation
- Evaluation

The Language Modeling Problem

- Setup: Assume a (finite) vocabulary of words

  $\mathcal{V} = \{\text{the, a, man, telescope, Beckham, two, Madrid, ...}\}$

- We can construct an (infinite) set of strings

  $\mathcal{V}^\dagger = \{\text{the, a, the a, the fan, the man, the man with the telescope, ...}\}$

- Data: given a *training set* of example sentences $x \in \mathcal{V}^\dagger$
- Problem: estimate a probability distribution

$$\sum_{x \in \mathcal{V}^\dagger} p(x) = 1$$

and $p(x) \geq 0$ for all $x \in \mathcal{V}^\dagger$

$$p(\text{the}) = 10^{-12}$$
$$p(\text{a}) = 10^{-13}$$
$$p(\text{the fan}) = 10^{-12}$$
$$p(\text{the fan saw Beckham}) = 2 \times 10^{-8}$$
$$p(\text{the fan saw saw}) = 10^{-15}$$
$$\ldots$$

The Noisy-Channel Model

- We want to predict a sentence given acoustics:

$$w^* = \arg\max_w P(w|a)$$

- The noisy channel approach:

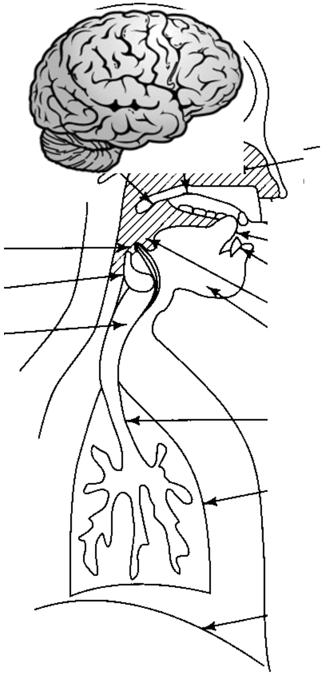
$$w^* = \arg\max_w P(w|a)$$

$$= \arg\max_w P(a|w)P(w)/P(a)$$

$$\propto \arg\max_w P(a|w)P(w)$$

Acoustic model: Distributions over acoustic waves given a sentence

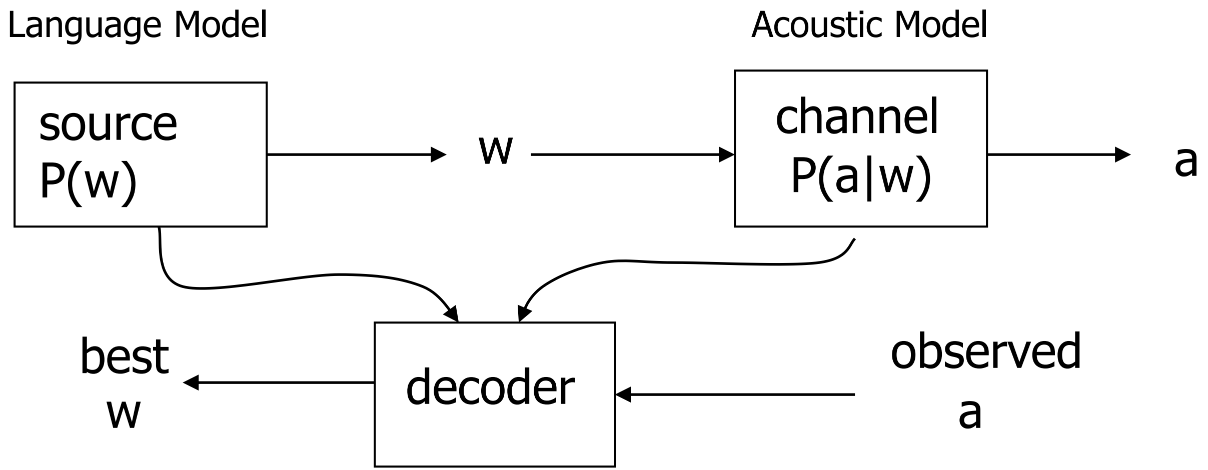
Language model: Distributions over sequences of words (sentences)



Acoustically Scored Hypotheses

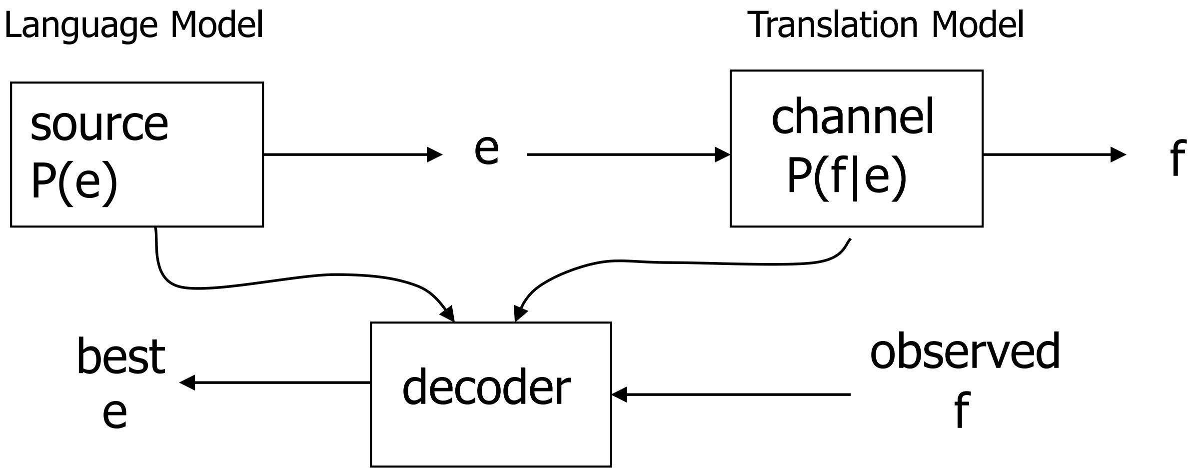
the station signs are in deep in english	-14732
the stations signs are in deep in english	-14735
the station signs are in deep into english	-14739
the station 's signs are in deep in english	-14740
the station signs are in deep in the english	-14741
the station signs are indeed in english	-14757
the station 's signs are indeed in english	-14760
the station signs are indians in english	-14790
the station signs are indian in english	-14799
the stations signs are indians in english	-14807
the stations signs are indians and english	-14815

ASR System Components



$$\underset{w}{\operatorname{argmax}} P(w|a) = \underset{w}{\operatorname{argmax}} P(a|w)P(w)$$

MT System Components



$$\underset{e}{\operatorname{argmax}}\ P(e|f) = \underset{e}{\operatorname{argmax}}\ P(f|e)P(e)$$

Probabilistic Language Models: Other Applications

- Why assign a probability to a sentence?
  - Machine Translation:
    - P(**high** winds tonite) > P(**large** winds tonite)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - Spell Correction
    - The office is about fifteen **minuets** from my house
      - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - + Summarization, question-answering, etc., etc.!!

Outline

- Motivation
- Task Definition
- N-Gram Probability Estimation
- Evaluation

Probabilistic Language Modeling

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

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

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

How to compute P(W)

- How to compute this joint probability:
  - P(its, water, is, so, transparent, that)

P("its water is so transparent") =

$P(\text{its}) \times P(\text{water}|\text{its}) \times P(\text{is}|\text{its water})$

$\times P(\text{so}|\text{its water is}) \times P(\text{transparent}|\text{its water is so})$

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 enough data for estimating these

Markov Assumption

Andrei Markov

- Simplifying assumption:

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

- Or maybe

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

Markov Assumption

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

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

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

Simplest Case: Unigram Models

- Simplest case: unigrams

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

- Generative process: pick a word, pick a word, … until you pick </s>
- Graphical model:

- Examples:
  - fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass
  - thrift, did, eighty, said, hard, 'm, july, bullish
  - that, or, limited, the

- Big problem with unigrams: P(the the the the) >> P(I like ice cream)!

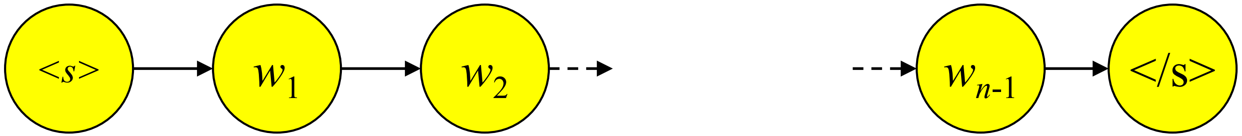
Bigram Models

- Conditioned on previous single word

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

- Generative process: pick <s>, pick a word conditioned on previous one, repeat until to pick </s>

- Graphical model:



- Examples:
  - texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen
  - outside, new, car, parking, lot, of, the, agreement, reached
  - this, would, be, a, record, november

N-Gram Models

- We can extend to trigrams, 4-grams, 5-grams
- N-gram models are (weighted) regular languages
  - Many linguistic arguments that language isn't regular.
    - Long-distance effects: "The computer which I had just put into the machine room on the fifth floor ___."
    - Recursive structure
  - We often get away with n-gram models

- PCFG LM (later):
  - [This, quarter, 's, surprisingly, independent, attack, paid, off, the, risk, involving, IRS, leaders, and, transportation, prices, .]
  - [It, could, be, announced, sometime, .]
  - [Mr., Toseland, believes, the, average, defense, economy, is, drafted, from, slightly, more, than, 12, stocks, .]

Outline

- Motivation
- Task Definition
- N-Gram Probability Estimation
- 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.

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 (in-vivo) evaluation of N-gram models

- Extrinsic evaluation
  - Time-consuming; requires building applications, new data
- 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.

Intuition of Perplexity

- The Shannon Game:
  - How well can we predict the next word?

    I always order pizza with cheese and ____

    The 33rd President of the US was ____

    I saw a ____

  - 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

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

….

and 1e-100

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:

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

For bigrams:

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

Minimizing perplexity is the same as maximizing probability

The Shannon Game intuition for perplexity

- From Josh Goodman
- How hard is the task of recognizing digits '0,1,2,3,4,5,6,7,8,9'
  - Perplexity 10
- How hard is recognizing (30,000) names at Microsoft.
  - Perplexity = 30,000
- If a system has to recognize
  - Operator (1 in 4)
  - Sales (1 in 4)
  - Technical Support (1 in 4)
  - 30,000 names (1 in 120,000 each)
  - Perplexity is 53
- Perplexity is weighted equivalent branching factor

Perplexity as branching factor

- Let's suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

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

Another form of Perplexity

$$2^{-l} \text{ where } l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

- Lower is better!
- Example: $|\mathcal{V}| = N$ and $q(w|\ldots) = \frac{1}{N}$
  - uniform model → perplexity is N
- Interpretation: effective vocabulary size (accounting for statistical regularities)
- Typical values for newspaper text:
  - Uniform: 20,000; Unigram: 1000s, Bigram: 700-1000, Trigram: 100-200

- Important note:
  - Its easy to get bogus perplexities by having bogus probabilities that sum to more than one over their event spaces. Be careful!

Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109