Natural Language Processing
Language Models

Sudeshna Sarkar

24 July 2019

Language Understanding

How likely is a sentence?

- P (the cat is eating a sandwich on a couch)
- P (about fifteen minutes from)
  P (about fifteen minuets from)
- P (I saw a bus) >> P (eyes awe a boss)

Language Model Definition

- How likely is a sentence $(w_1, w_2, \ldots, w_n)$?

- A statistical language model is a probability distribution over sequences of words.

$$P(w_1, w_2, \ldots, w_n) = P(w_n | w_{n-1}, w_{n-2}, \ldots, w_1)$$

Application

Application	Signal Y
automatic speech recognition	acoustic signal
machine translation	sequence of words in a foreign language
spelling correction	sequence of characters produced by a possibly imperfect typist

source-channel model

Goal: to determine W from Y

Probabilistic Language Models

- The goal: assign a probability to a sentence
  - Machine Translation:
    - » P(**high** winds tonite) > P(**large** winds tonite)
  - Spelling 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.

Motivation: Noisy Channel Models

- A pattern for modeling a pair of random variables, X and Y :

source → Y → channel → X

- Y is the plaintext, the true message, the missing information, the output
- X is the ciphertext, the garbled message, the observable evidence, the input
- Decoding: select y given X = x.

$$y^* = \underset{y}{argmax}\ p(y|x)$$

$$= \underset{y}{argmax}\ \frac{p(x|y).p(y)}{p(x)}$$

$$= \underset{y}{argmax}\ \mathrm{p(x|y)}\ .\ \mathrm{p(y)}$$

Channel model

Source model

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:

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

For bigrams:

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

Minimizing perplexity is the same as maximizing probability

Completion Prediction

- A language model also supports predicting the completion of a sentence.
  - Please turn off your cell _____
  - Your program does not ______
  - *Stocks plunged this ….*
  - Let’s meet in Times ….
- *Predictive text input* systems can guess what you are typing and give choices on how to complete it.

Human Word Prediction

- The ability to predict future words in an utterance.
- How?
  - Domain knowledge
  - Syntactic knowledge
  - Lexical knowledge

Corpora

- Corpora are online collections of text and speech
  - Brown Corpus
  - Wall Street Journal
  - AP newswire
  - Hansards
  - DARPA/NIST text/speech corpora (Call Home, ATIS, switchboard, Broadcast News, TDT, Communicator)
  - TRAINS, Radio News

N-Gram Models

- Estimate probability of each word given prior context.
  - P(phone | Please turn off your cell)
- Number of parameters required grows exponentially with the number of words of prior context.
- An N-gram model uses only N–1 words of prior context.
  - Unigram: P(phone)
  - Bigram: P(phone | cell)
  - Trigram: P(phone | your cell)
- The ***Markov assumption*** is the presumption that the future behavior of a dynamical system only depends on its recent history. In particular, in a ***kth-order Markov model***, the next state only depends on the *k* most recent states, therefore an N-gram model is a (N–1)-order Markov model.

Google 1-T Corpus

- *1 trillion word tokens*
  - Number of tokens –1,024,908,267,229
  - Number of sentences –95,119,665,584
  - Number of unigrams –13,588,391
  - Number of bigrams –314,843,401
  - Number of trigrams –977,069,902
  - Number of fourgrams– 1,313,818,354
  - Number of fivegrams– 1,176,470,663

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234

N-Gram Model Formulas

- Word sequences $w_1^n = w_1 \ldots w_n$
- Chain rule of probability

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

- Bigram approximation

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

- N-gram approximation

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

Estimating Probabilities

- N-gram conditional probabilities can be estimated from raw text based on the ***relative frequency*** of word sequences.

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

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

Generative Model & MLE

- An N-gram model can be seen as a probabilistic automata for generating sentences.

- Relative frequency estimates are ***maximum likelihood estimates*** (MLE) since they maximize the probability that the model *M* will generate the training corpus *T*.

$$\hat{\lambda} = \operatorname*{argmax}_{\lambda} P(T \mid M(\lambda))$$

Train and Test Corpora

- A language model is trained on a large corpus of text to estimate good parameter values.
- Model can be evaluated based on its ability to predict a high probability for a disjoint (held-out) test corpus
- May need to ***adapt*** a general model to a small amount of new (***in-domain***) data by adding highly weighted small corpus to original training data.

Data Sparsity

- Data sparsity:
- # of all possible n-grams: $|V|^n$, where $|V|$ is the size of the vocabulary. Most of them never occur.

Training Set:
... denied the allegations
... denied the reports
... denied the claims
... denied the request

Test Set:
... denied the offer
... denied the loan

$P(\text{offer}|\text{denied the}) = 0$

False independence assumption

- We assume that each word is only conditioned on the previous n-1 words
- “The dogs chasing the cat bark”.
- The tri-gram probability P (bark|the cat) is very low

Unknown Words

- How to handle ***out of vocabulary*** (OOV) words?

1. Train a model that includes an explicit symbol for an unknown word (<UNK>).
   - Choose a vocabulary in advance and replace other words in the training corpus with <UNK>.
   - Replace the first occurrence of each word in the training data with <UNK>.
2. Character based models

Sample Perplexity Evaluation

- Models trained on 38 million words from the Wall Street Journal (WSJ) using a 19,979 word vocabulary.
- Evaluate on a disjoint set of 1.5 million WSJ words.

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Empirical Observations

- A small number of events occur with high frequency
- A large number of events occur with low frequency
- Some of the zeroes in the table are low frequency events you haven't seen yet.

- Words follow a Zipfian distribution
  - Small number of words occur very frequently
  - A large number are seen only once
- Zipf'slaw: a word's frequency is approximately inversely proportional to its rank in the word distribution list

Smoothing

- Many rare (but not impossible) combinations never occur in training, so MLE incorrectly assigns zero to many parameters (***sparse data***).
- If a new combination occurs during testing, it is given a probability of zero and the entire sequence gets a probability of zero (i.e. infinite perplexity).
- In practice, parameters are ***smoothed*** (or ***regularized***) to reassign some probability mass to unseen events.
  - Adding probability mass to unseen events requires removing it from seen ones (***discounting***) in order to maintain a joint distribution that sums to 1.

Laplace (Add-One) Smoothing

- "Hallucinate" additional training data in which each possible N-gram occurs exactly once and adjust estimates.

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

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

V: the total number of possible (N–1)-grams (i.e. the vocabulary size for a bigram (n-gram) model).

- Tends to reassign too much mass to unseen events, so can be adjusted to add δ

Advanced Smoothing

- Improved smoothing for language models.
  - Interpolation
  - Backoff
  - Kneser-Ney
  - Class-based (cluster) N-grams

Model Combination

- As N increases, the power (expressiveness) of an N-gram model increases
  - *but* the ability to estimate accurate parameters from sparse data decreases

- A general approach is to combine the results of multiple N-gram models of increasing complexity (i.e. increasing N).

Interpolation

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

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

- Learn proper values for λ_i by training to (approximately) maximize the likelihood of an independent *development* corpus.

Backoff

- Only use lower-order model when data for higher-order model is unavailable.
- Recursively back-off to weaker models until data is available.

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

- P* is a discounted probability estimate to reserve mass for unseen events and α's are back-off weights.

Huge web-scale n-grams

- How to deal with, e.g., Google N-gram corpus
- Pruning
  - Only store N-grams with count > threshold.
    - Remove singletons of higher-order n-grams
  - Entropy-based pruning
- Efficiency
  - Efficient data structures like tries
  - Bloom filters: approximate language models
  - Store words as indexes, not strings
    - Use Huffman coding to fit large numbers of words into two bytes
  - Quantize probabilities (4-8 bits instead of 8-byte float)

A Problem for N-Grams: Long Distance Dependencies

- Syntactic dependencies
  - “The ***man*** next to the large oak tree near the grocery store on the corner **is** tall.”
  - “The ***men*** next to the large oak tree near the grocery store on the corner **are** tall.”

- Semantic dependencies
  - “The ***bird*** next to the large oak tree near the grocery store on the corner **flies** rapidly.”
  - “The ***man*** next to the large oak tree near the grocery store on the corner **talks** rapidly.”

Neural language model

$$P(w_t | w_{t-n}, \dots, w_{t-1}) = \cancel{\frac{C(w_{t-n}, \dots, w_t)}{C(w_{t-n}, \dots, w_{t-1})}}$$

$$= f_\theta(w_{t-n}, \dots, w_{t-1})$$

- Parametric estimator
- We need numerical representation of words