Language models

CS 4705

Human Word Prediction

- Clearly, at least some of us have the ability to predict future words in an utterance.
- How?
  - Domain knowledge: red blood vs. red hat
  - Syntactic knowledge: the…<adj|noun>
  - Lexical knowledge: baked <potato vs. steak>

Claim

- A useful part of the knowledge needed to allow Word Prediction can be captured using simple statistical techniques
- In particular, we'll be interested in the notion of the probability of a sequence (of letters, words,…)

Useful Applications

- Why do we want to predict a word, given some preceding words?
  - Rank the likelihood of sequences containing various alternative hypotheses, e.g. for ASR

  Theatre owners say popcorn/unicorn sales have doubled...

  - Assess the likelihood/goodness of a sentence, e.g. for text generation or machine translation

  The doctor recommended a cat scan.

  El doctor recommendó una exploración del gato.

N-Gram Models of Language

- Use the previous N-1 words in a sequence to predict the next word
- Language Model (LM)
  - unigrams, bigrams, trigrams,…
- How do we train these models?
  - Very large corpora

Corpora

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

Counting Words in Corpora

- What is a word?
  - e.g., are cat and cats the same word?
  - September and Sept?
  - zero and oh?
  - Is _ a word? * ?) . ,
  - How many words are there in don’t ? Gonna ?
  - In Japanese and Chinese text -- how do we identify a word?

Terminology

- Sentence: unit of written language
- Utterance: unit of spoken language
- Word Form: the inflected form as it actually appears in the corpus
- Lemma: an abstract form, shared by word forms having the same stem, part of speech, word sense – stands for the class of words with same stem
- Types: number of distinct words in a corpus (vocabulary size)
- Tokens: total number of words

Simple N-Grams

- Assume a language has T word types in its lexicon, how likely is word x to follow word y?
  - Simplest model of word probability: 1/T
  - Alternative 1: estimate likelihood of x occurring in new text based on its general frequency of occurrence estimated from a corpus (unigram probability)

    popcorn is more likely to occur than unicorn
  - Alternative 2: condition the likelihood of x occurring in the context of previous words (bigrams, trigrams,…)

    mythical unicorn is more likely than mythical popcorn

Computing the Probability of a Word Sequence

- Compute the product of component conditional probabilities?
  - P(the mythical unicorn) = P(the) * P(mythical|the) * P(unicorn|the mythical)
- But…the *longer* the sequence, the *less likely* we are to find it in a training corpus

  P(Most biologists and folklore specialists believe that in fact the mythical unicorn horns derived from the narwhal)

- What can we do?

Bigram Model

- Approximate $P(w_n | w_1^{n-1})$ by $P(w_n | w_{n-1})$
  - E.g., P(unicorn|the mythical) by P(unicorn|mythical)
- Markov *assumption*: the probability of a word depends only on the probability of a limited history
- ***Generalization: the probability of a word depends only on the probability of the n previous words***
  - trigrams, 4-grams, 5-grams,…
  - the higher n is, the more data needed to train
  - backoff models…

Using N-Grams

- For N-gram models
  - $P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-N+1}^{n-1})$
    - E.g. for bigrams, $P(w_8 | w_1^{8-1}) \approx P(w_8 | w_{8-2+1}^{8-1})$
  - $P(w_{n-1},w_n) = P(w_n | w_{n-1}) P(w_{n-1})$
  - $P(w_{8-1},w_8) = P(w_8 | w_7) P(w_7)$
  - By the Chain Rule we can decompose a joint probability, e.g. $P(w_1,w_2,w_3)$ as follows

    $P(w_1,w_2, ...,w_n) = P(w_1|w_2,w_3,...,w_n) P(w_2|w_3, ...,w_n)$ … $P(w_{n-1}|w_n) P(w_n)$

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

- For bigrams then, the probability of a sequence is just the product of the conditional probabilities of its bigrams, e.g.

P(the,mythical,unicorn) = P(unicorn|mythical) P(mythical|the) P(the|<start>)

Training and Testing

- N-Gram probabilities come from a training corpus
  - overly narrow corpus: probabilities don't generalize
  - overly general corpus: probabilities don't reflect task or domain
- A separate test corpus is used to evaluate the model
  - held out test set; development (dev) test set
  - Simple baseline
  - results tested for statistical significance – how do they differ from a baseline? Other results?

A Simple Bigram Example

- Estimate the likelihood of the sentence I want to eat Chinese food.
  - P(I want to eat Chinese food) = P(I | <start>) P(want | I) P(to | want) P(eat | to) P(Chinese | eat) P(food | Chinese) P(<end>|food)
- What do we need to calculate these likelihoods?
  - Bigram probabilities for each word pair sequence in the sentence
  - Calculated from a large corpus

Early Bigram Probabilities from BERP

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

<start> I	.25	Want some	.04
<start> I’d	.06	Want Thai	.01
<start> Tell	.04	To eat	.26
<start> I’m	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don’t	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

- P(I want to eat British food) = P(I|<start>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British) = .25*.32*.65*.26*.001*.60 = .000080
  - Suppose P(<end>|food) = .2?
  - How would we calculate I want to eat Chinese food ?
- Probabilities roughly capture ``syntactic'' facts and ``world knowledge''
  - eat is often followed by an NP
  - British food is not too popular
- N-gram models can be trained by counting and normalization

Early BERP Bigram Counts

	I	Want	To	Eat	Chinese	Food	lunch
I	8	1087	0	13	0	0	0
Want	3	0	786	0	6	8	6
To	3	0	10	860	3	0	12
Eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
Food	19	0	17	0	0	0	0
Lunch	4	0	0	0	0	1	0

Early BERP Bigram Probabilities

- Normalization: divide each row's counts by appropriate unigram counts for w_{n-1}

I	Want	To	Eat	Chinese	Food	Lunch
3437	1215	3256	938	213	1506	459

- Computing the bigram probability of I I
  - C(I,I)/C(I in call contexts)
  - p (I|I) = 8 / 3437 = .0023
- Maximum Likelihood Estimation (MLE): relative frequency $\frac{freq(w_1, w_2)}{freq(w_1)}$

What do we learn about the language?

- What's being captured with ...
  - P(want | I) = .32
  - P(to | want) = .65
  - P(eat | to) = .26
  - P(food | Chinese) = .56
  - P(lunch | eat) = .055
- What about...
  - P(I | I) = .0023
  - P(I | want) = .0025
  - P(I | food) = .013

- $P(I \mid I) = .0023$ I I I I want
- $P(I \mid want) = .0025$ I want I want
- $P(I \mid food) = .013$ the kind of food I want is ...

Evaluation and Data Sparsity Questions

- Perplexity and entropy: how do you ***estimate*** how well your language model fits a corpus once you're done?
- Smoothing and Backoff : how do you handle unseen n-grams?

Perplexity and Entropy

- Information theoretic metrics
  - Useful in measuring how well a grammar or language model (LM) models a natural language or a corpus
- Entropy: With 2 LMs and a corpus, which LM is the better match for the corpus? How much information is there (in e.g. a grammar or LM) about what the next word will be? More is better!
  - For a random variable ***X*** ranging over e.g. bigrams and a probability function ***p(x),*** the entropy of X is the expected negative log probability

$$H(X) = -\sum_{x=1}^{x=n} p(x) \log_2 p(x)$$

- Entropy is the lower bound on the # of bits it takes to encode information e.g. about bigram likelihood

- Cross Entropy
  - An upper bound on entropy derived from estimating true entropy by a subset of possible strings – we don't know the real probability distribution

- Perplexity $$PP(W) = 2^{H(W)}$$
  - At each choice point in a grammar
    - What are the average number of choices that can be made, weighted by their probabilities of occurrence?
    - I.e., Weighted average branching factor
  - How much probability does a grammar or language model (LM) assign to the sentences of a corpus, compared to another LM? The more information, the lower perplexity

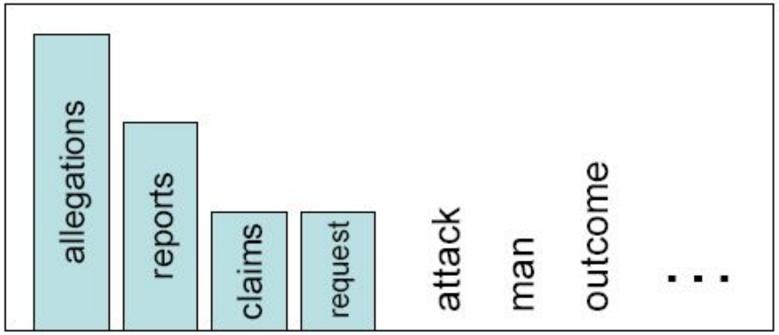
Smoothing

- Words follow a Zipfian distribution
  - Small number of words occur very frequently
  - A large number are seen only once
  - Zipf's law: *a word's frequency is approximately inversely proportional to its rank in the word distribution list*
- Zero probabilities on one bigram cause a zero probability on the entire sentence
- So….how do we estimate the likelihood of unseen n-grams?

Smoothing is like Robin Hood:
Steal from the rich and give to the poor (in probability mass)

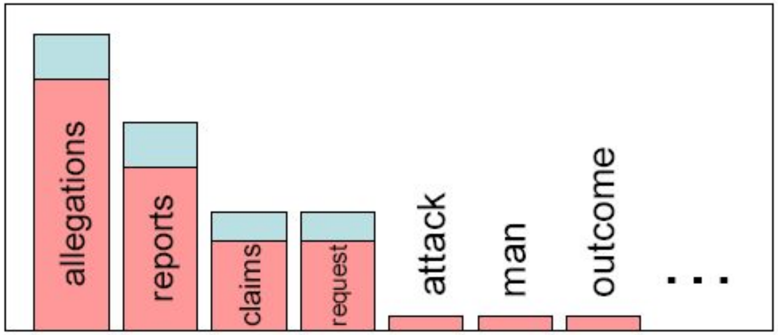
- We often want to make predictions from sparse statistics:

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



- Smoothing flattens spiky distributions so they generalize better

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



- Very important all over NLP, but easy to do badly!

Slide from Dan Klein

Laplace Smoothing

- For unigrams:
  - Add 1 to every word (type) count to get an adjusted count c^*
  - Normalize by N (#tokens) + V (#types)
  - Original unigram probability

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

  - New unigram probability

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

Unigram Smoothing Example

- Tiny Corpus, V=4; N=20

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

Word	True Ct	Unigram Prob	New Ct	Adjusted Prob
eat	10	.5	11	.46
British	4	.2	5	.21
food	6	.3	7	.29
happily	0	.0	1	.04
	20	1.0	~20	1.0

- ***So, we lower some (larger) observed counts in order to include unobserved vocabulary***
- For bigrams:
  - Original $$P(w_n|w_{n-1}) = \frac{c(w_n|w_{n-1})}{c(w_{n-1})}$$
  - New $$P(w_n|w_{n-1}) = \frac{c(w_n|w_{n-1})+1}{c(w_{n-1})+V}$$
  - ***But this change counts drastically***:
    - ***Too much weight*** given to unseen ngrams
    - In practice, unsmoothed bigrams often work better!
    - Can we smooth more usefully?

Good-Turing Discounting

- Re-estimate amount of probability mass for zero (or low count) ngrams by looking at ngrams with higher counts
  - Estimate $$c^* = (c+1)\frac{N_{c+1}}{N_c}$$
  - E.g. N_0's adjusted count is a function of the count of ngrams that occur once, N_1
  - Assumes:
    - Word bigrams each follow a binomial distribution
    - We know number of unseen bigrams (VxV-seen)

- Add-one smoothing (easy, but inaccurate)
  - Add 1 to every word count (Note: this is type)
  - Increment normalization factor by Vocabulary size: N (tokens) + *V (types)*
- Good-Turing
  - Re-estimate amount of probability mass for zero (or low count) ngrams by looking at ngrams with higher counts

Summary

- N-gram probabilities can be used to *estimate* the likelihood
  - Of a word occurring in a context (N-1)
  - Of a sentence occurring at all
- Entropy and perplexity can be used to evaluate the information content of a language and the goodness of fit of a LM or grammar
- Smoothing techniques and backoff models deal with problems of unseen words in corpus