N-grams

April 13, 2009

Overview

- N-grams motivation
- Basic N-gram counts
- Training and test sets
- Entropy and perplexity
- Smoothing
  - Why
  - Example strategies
- Interpolation and Backoff
- Discussion/comprehension questions

Language models

- Given a sample of text (sentence, paragraph, document), when might you want to:
  - Tell how much it looks like “English”?
  - Tell how much it looks like Shakespeare, or the WSJ?
  - Tell whether it looks more or less like “English” than some other sample?
- How would you do that?

N-gram models: Probabilities of sequences of words

- Ideally: Probability of word N, given the presence of words 1 through N-1.
- Approximation: Probability of word N depends only on last M words.
  - Is this approximation true?
- Given a trained up N-gram model, how do we use it?
- How do we train up an N-gram model?

Digression: Probability vs. frequency

- Probability: how likely something is to happen
- Frequency: how frequently something actually happens
- Probability clearly influences frequency.
- Frequency can be used to estimate probability.
- … but they are not the same thing
- If a bigram never appears in the training corpus
  - What is its observed frequency?
  - What is its probability?

N-grams training

- How do we train up an N-gram model?
- Why not just go for really long N-grams?
- Look ahead: Smoothing, back-off

N-grams: interim summary

- Probabilistic language models can be very useful components in NLP systems.
- N-grams only capture dependencies within their gram length.
- Longer n-grams capture more information, but have greater problems with data sparsity.
- Data sparsity can be addressed somewhat by smoothing and back-off.

Simple N-grams

- 0-gram: Any word can follow any other with uniform probability.
- unigram: Each word is assigned its own probability, constant regardless of context.
- But words appear with different frequencies in different contexts:

$$P(w_1, w_2, \ldots, w_{n-1}, w_n) = P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1})$$

Bigrams

- Approximate $P(w_n \mid w_1^{n-1})$ by $P(w_n \mid w_{n-1})$
- Markov assumption: The probability of an event is dependent on only a finite history. Here, a finite number of previous words.
- Bigrams make a 'first-order' Markov assumption: look back only one word

$$P(w_1^n) \approx P(w_1 \mid \langle start \rangle) P(w_2 \mid w_1) P(w_3 \mid w_2) ... P(w_n \mid w_{n-1})$$

- In practice, we add logs of probabilities rather than multiplying raw probabilities
- Trigrams look back two words, etc.

Estimating bigram probabilities

- Count bigram occurrences in some corpus, and divide by the bigram frequencies of the first word.

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

- Maximum Likelihood Estimation (MLE): Estimates 'true' probabilities as those that make the training set most likely (but not necessarily any other corpus).
- In this equation, which probabilities sum to 1?
- What is the corresponding equation for trigrams?

Training sets and test sets

- Like many of the models we see in NLP, the probabilities of an N-gram model come from a training corpus.
- When evaluating a model, compute probabilities on a separate test set.
- Testing on sentences you've trained the model on gives you artificially high results.
- A development set is often useful as well.

Perplexity

- The most common intrinsic evaluation metric for N-gram models
- A function of the probability the model assigns to a test set

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

- A measure of ‘surprise’: we would be as surprised to see the next word as we would if there were on average *PP* possibilities.
- The lower the perplexity, the better

Sparse Data

- Even very large corpora have relatively few bigram types.
- The complete works of Shakespeare (884,647 word tokens/29,066 wordform types), contains less than 300,000 bigram types.
  - How many bigram types are possible with 29,066 wordform types?
  - Are all of them legitimate English word sequences?
  - At most, how many bigram types will an 884,647 word corpus have?
- Should the ratio of possible to attested N-gram types get higher or lower as N increases?

Overtraining (1/2)

- An 884,647-gram model will perfectly model the works of Shakespeare (in a particular order), but give a probability of 0% to anything else.
- Likewise, even a bigram model overestimates the probability of the bigrams it saw in the training corpus and underestimates the probability of other bigrams.
- Solution 1, ‘smoothing’: take some of the probability ‘mass’ from the attested bigrams and redistribute it to the unattested ones.

Overtraining (2/2)

- How do we even know what the unattested bigrams are?
- What about bigrams that contain words not in the training data?
- Why do we have to discount probability somewhere in order to reassign it?

Laplace smoothing

- Pretend we saw every *N*-gram once more than we actually did.
- For unigrams:

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

- We can look at it in terms of an adjusted count:

$$c_i^* = (c_i + 1)\frac{N}{N + V}$$

- Laplace smoothing for bigrams:

$$P^*_{Laplace}(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

Good-Turing Discounting

- Intuition: use the count of things you've seen once to help estimate the count of things you've never seen.
- N_c = the number of N-grams seen c times.
- Use adjusted counts

$$c^*_{GT} = (c+1)\frac{N_{c+1}}{N_c}$$

- Why won't that work for unseen n-grams? Instead:

$$P^*_{GT}(\textit{unseen item}) = \frac{N_1}{N}$$

Interpolation and Backoff

- To make up for data sparsity, we can use lower-order N-gram counts in place of counts of things we haven't seen.
- With interpolation, we add trigram, bigram and unigram counts:

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

- With backoff, when the N-gram has a count of 0, we use the (discounted) probability of the next lower-order N-gram.

N-grams: Comprehension questions

- Why would n-gram models trained on bigger corpora be better?
- Why do we need smoothing?
- Would you expect an email corpus or a newspaper corpus to provide a better training set if the test set is blogs? Why?
- What kind of linguistic dependencies do N-grams model?
- Would you expect these N-gram models to work equally well for Arabic? Latin? Other languages? Why or why not?
- How might you modify them?

Overview

- N-grams motivation
- Basic N-gram counts
- Training and test sets
- Entropy and perplexity
- Smoothing
  - Why
  - Example strategies
- Interpolation and Backoff
- Discussion/comprehension questions