Ling/CSE 472: Introduction to Computational Linguistics

4/28: n-grams

Big picture

- A class with “intro” in the title usually provides a structured overview of some masterable basics
- This is a different kind of “intro” (being a 400-level course): Exposure to a wide range of ideas and problems, without the expectation (or sense of satisfaction) of mastering a specific set of lessons
- What’s something that you know now about computational linguistics that you didn’t know before?
- What’s been the most surprising thing so far?

Overview

- What are N-grams? (high-level)
- When are they useful?
- Evaluation
- Simple (un-smoothed) N-grams
- Training and test sets
- Unknown words
- N-grams and linguistic knowledge
- Smoothing, back-off, interpolation
- Reading questions

What are N-grams?

- A way of modeling the probability of a string of words.
- ... or the probability of the N+1st word being *w* given words 1-N.

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

- We'll come back to this in more detail in a moment

When are they useful?

- When would you want to know the relative probability of two strings?
- ... the relative probability of two words given the previous N words?
- ... the absolute probability of a string?

When are they useful?

- When would you want to know the relative probability of two strings?
  - Choosing among ASR candidates
  - Choosing among MT candidates
- ... the relative probability of two words given the previous N words?
  - Choosing among spell correction candidates
  - Predictive text (T9, AAC): Does your cell phone store an n-gram model?
- ... the absolute probability of a string? (kind of)
  - Spell checking: relative to a threshold
  - Language ID, authorship ID: relative to probability from some other model

Evaluating N-gram models

- What kinds of extrinsic evaluation are possible?
- What kinds of intrinsic evaluation are possible?
- What different kinds of models could you compare?

Evaluating N-gram models

- What kinds of extrinsic evaluation are possible?
  - ASR, MT, ...
- What kinds of intrinsic evaluation are possible?
  - Perplexity: Given an n-gram model trained on some training set, how well does it predict the test set? (i.e., what probability does it assign to the test set?)
- What different kinds of models could you compare?
  - Different: training data, smoothing/back-off techniques, higher-level tokens

Training and test sets

- Training and test sets must be distinct, otherwise probabilities will be artificially high
- This is just a special case of a more general reason: The purpose of test sets is to see if the method generalizes to unseen data
- Training and test sets are typically drawn from the same or comparable corpora
  - Why?
  - What if they aren't?

Simple (un-smoothed) N-grams

- What we'd really like to calculate:

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

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

- But we'd never find a corpus big enough.

- Why not?

So: An approximation

- Markov assumption: The probability of a given word only depends on the previous word

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

- Is this assumption valid?

- (NB: That’s a bigram model ... for a trigram model, we look at the previous two words, etc.)

Maximum Likelihood Estimates for bigram counts

- Bigram probability for a word *y* given a previous word *x*:
- Out of all the times you saw *x*, in what percentage was it followed by *y*?

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

Probability v. Frequency

- Probability: How likely something is to happen
- Frequency: How frequently something has happened in a set of observations
- Probability clearly influences frequency
- Frequency can be used to estimate probability (what else can?)
  - ... but they are not the same thing
- If a bigram never appears in a training corpus,
  - What is its observed frequency?
  - What is its probability?

Example

- What are the bigrams in the following mini corpus? What are their MLEs?

<s> How much wood would a wood chuck chuck if a wood chuck could chuck wood? </s> <s> As much wood as a wood chuck could if a wood chuck could chuck wood. </s>

- What probability does that bigram model assign to the following sentences?

<s> How much wood. </s>
<s> How much wood? </s>
<s> As much wood chuck chuck chuck wood. </s>
<s> How would a wood chuck chuck ? </s>

bigrams

- <s> How = 1/2
- How much = 1
- much wood = 1
- wood would = 1/8
- would a = 1
- a wood = 1
- wood chuck = 1/2
- chuck chuck = 1/7
- chuck if = 1/7
- if a = 1
- chuck could = 3/7
- could chuck = 2/3
- chuck wood = 2/7
- wood ? = 1/8
- ? </s> = 1
- <s> As = 1/2
- As much = 1/2
- wood as = 1/8
- as a = 1/2
- could if = 1/3
- wood . = 1/8
- . </s> = 1

Sentences

<s> How much wood. </s>
<s> How much wood? </s>
<s> As much wood chuck chuck chuck wood. </s>
<s> How would a wood chuck chuck ? </s>

1. 1/2 * 1 * 1 * 1/8 * 1 = 1/16
2. 1/2 * 1 * 1 * 1/8 * 1 = 1/16
3. 1/2 * 1/2 * 1 * 1/2 * 1/7 * 1/7 * 2/7 * 1 = 1/1372
4. 1/2 * 0 ... = 0

Counting things in a corpus

- Type/token distinction
- But what counts as a token? What are some cases where this is not obvious?
- And what counts as the same type? What are some cases where this is not obvious?
- Is there a single right answer?

Counting things in a corpus

- Type/token distinction
- But what counts as a token? What are some cases where this is not obvious?
  - Contracted forms, punctuation, hyphenated forms, words with spaces (*New York*), ...
- And what counts as the same type? What are some cases where this is not obvious?
  - Caps/non-caps, word-form/lemma, homographs, ...
- Is there a single right answer?
  - No: It depends on the application context

Unknown words

- What would a n-gram model trained as described so far say about the probability of a sentence with an unknown word in it?

- What could be done about that?

Unknown words

- What would a n-gram model trained as described so far say about the probability of a sentence with an unknown word in it? -- 0
- What could be done about that?
  - Choose a vocabulary smaller than the actual V, and replace all other words with <UNK> -- Any words you might want to be sure to include in V?
  - Or: Replace the first occurrence of every word in the training set with <UNK>
  - Then: Estimate probabilities of <UNK> just like any other word

N-grams and linguistic knowledge

- Is an n-gram model a grammar?
- What kinds of information about a language does it capture?
- What kinds of information about a language does it miss?

N-grams and linguistic knowledge

- N-gram models are supposed to be “language-independent” in that they don’t require specific knowledge about a language to create --- just text.

- Would you expect them to work equally well across languages? Why or why not?

Review:
Calculating simple (unsmoothed) n-grams

- Bigram probability for a word *y* given a previous word *x*:

- Out of all the times you saw *x*, in what percentage was it followed by *y*?

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

- What's wrong with this?

- How can it be improved?

Solutions

- Smoothing: redistribute probability mass from seen to unseen n-grams
- Backoff: Use lower-order n-grams when higher-order ones aren’t available
- Interpolation: Use lower-order and higher-order ones together, with weights

Smoothing

- Add-one smoothing: Before normalizing the counts, add one to every possible n-gram (given vocabulary + <UNK>)
  - What's wrong with this?
- Simple Good-Turing Discounting: Use the count of things observed only once (*hapax legomena*) to estimate the count of the unseen
  - *missing mass* = P(things with freq 0 in training) = hapaxes/all items

Simple Linear Interpolation

- Combine different order N-grams by linear interpolation

$$\begin{aligned}\hat{P}(w_n \mid w_{n-2}w_{n-1}) \quad &= \quad \lambda_1 P(w_n \mid w_{n-2}w_{n-1}) \\ &\quad\quad \lambda_2 P(w_n \mid w_{n-1}) \\ &\quad\quad \lambda_3 P(w_n)\end{aligned}$$

- Lambdas must sum to 1. Why?

- How are lambdas set?

Backoff

- Intuition: Use information from lower-order n-grams only if higher-order ones aren't there

- Because probabilities must sum to 1 over whole model, use discounting to get revised probabilities for each n-gram

Practical Issues

- N-gram probabilities get problematic for computation (underflow) because they are so small
- Solution: convert to log probabilities, changing multiplication to addition and working with numbers that aren't so small
- Toolkits: This has all been implemented already, so you don't need to reimplement.
  - SRILM: http://www.speech.sri.com/projects/srilm/
  - Already on patas: /NLP_TOOLS/ml_tools/lm/srilm/latest

Reading questions

- Why do we call this 'language modeling'? It hardly seems to be modeling the way language works, or at least only superficially.

- A sentence in most languages that I know of is not always a linear structure. For example, in English a syntax tree can be used to describe the structure of sentences. N-gram seems to disregard this fact entirely, and simplifies the concept of sentence and paragraph by making the assumption that "an utterance of token solely depends on preceding n tokens". Given this, how does an n-gram LM gnerally perform in terms of the complexity of the possible outputs? Can increase in n eventually encompass highly complex sentence structures?

Reading questions

- Why exactly is it undesirable to have an individual probability distribution for each sentence length?

- In the reading, there was an example about a n-gram model trained on Shakespeare's works. It showed that as we increase N, the model outputs more cohesive text. In this sense, I guess the ideal N would be equal to the number of unique sentences? I wonder how is N usually determined?

Reading questions

- Training sets need to account for sparsity, genre, and dialect; it seems that a lot of errors occur when programs are used on sets of data they're not intended to be used on. Why is this tested? Should programmers be trying to make universal systems?

- The reading seems to suggest that there can be an "improvement in perplexity" so what ways are there to improve perplexity of an n-gram model besides changing the n in n-gram, say from unigram vs bigram vs trigram? Also, does that mean there is a limit for how much improvement there can be for a model?

Reading questions

- Also, about n-grams, do language like Japanese undergo word segmentation before applying n-gram? As about OOV, I think this is a problem for tokenization and word segmentation since we simply cannot include all the vocabulary of a language. So how can we balance between OOV rate and performance?

- When creating a sentence, how can the training model stop itself from creating sequences of determiner/preposition gibberish?

Reading questions

- It’s not clear to me what a word appearing more frequently as a “novel continuation” entails linguistically - there’s not really an intuitive connection between it and a syntactic category as far as I can tell, at least not for most words. I think I understand why the resulting equation leads to both a more balanced distribution and less unpredictability regarding unigrams, but the assumptions behind the formula seem weird.

Reading questions

- Would it be possible to train based upon the classification of the words themselves? For example, that nouns come after "eat" most often (page 6). Would this require tagging every noun by category? Or could the computers be trained to recognize words with noun tags, and transitive verb tags, and then a syntactic-semantic rule structure be set up?

- Would it be possible, or is it already in practice to, along with training n-gram models, also train them on the syntax of the corpus and allow the syntax to be used as a tool to guess whether an <UNK> word is viable in a certain context based on its part of speech, and the part of speech of the words surrounding it?

- One idea that has come up in the past is syntactic n-grams. I would expect an n-gram generation with respect to dependency relations would be more informative than a base n-gram.

Reading questions

- I'm still confused about what exactly 'stupid backoff' smoothing does and why it's called 'stupid.' I understand that the purpose of smoothing is to account for unseen events but I'm still confused as to how this is done.

- Is there any downside to stupid backoff? Is there anything lost when we stop building a language model as a proper probability distribution?

Reading questions

- Some of the math went over my head. I've seen 'perplexity' show up before, so it seems worth understanding. Why is the probability of the test set inversed? How is perplexity more generally calculated?

- What's the point of using perplexity as an evaluation metric if it's just the inverse probability of the test set? Couldn't probability be used instead?