Ling/CSE 472:
Introduction to Computational Linguistics

4/25: 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:

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

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

- 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

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

- 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

- Are these n-gram models only used for things like guessing the next word a user is going to type in a text message? Or can they also carry over to chatbots as well?

- Are these models based off of probability for predicting the next words still largely employed in NLP today? The paper brings up issues with, if a certain word or phrase is not there, then the probability of the machine choosing it will be 0, and I'm sure there are a wide variety of different issues that could cause people who use that LM (even with all of the strategies that try to address that 0 probability).

Reading questions

- How recently did this type of text prediction become commonly in use? Following the example on page 1 about the use of probabilities in machine translation, it seems like online translators were a lot worse only 5-10 years ago than they are now. Today, Google translate can accurately translate the Mandarin sentence "他 向 记者 介绍了 主要 内容" to "He introduced the main content to the reporter." But I feel like in the 2010s you might've gotten something like "He to reporters introduced main content," which directly translates word for word but doesn't make much sense syntactically. Is it accurate to think that machine translation has improved a lot in a very short time? And is that due to n-grams?

Reading questions

- The reading mentions that n-grams can be used in machine translation to select the most natural-sounding sentence out of a few options - is this common/standard in machine translators?

Reading questions

- What is the largest kind of n-gram that exists? The reading mentioned up to 5-gram models, but are there any larger than that?

- At what point does having a larger n-gram become unhelpful? Why are trigrams more common than bigrams, but why not 4-grams?

Reading questions

- I've had the opportunity to implement n-grams in practice and one of the problems I've found is the occasional tendency of these systems to end up producing nonsensical or grammatically incorrect phrases. This is especially true when using lower order n-grams (with smaller values of n) where the context is too limited to capture dependencies earlier in the sentence. This may also result in the end of the sentence not being captured correctly, leading to long nonsensical run-on sentences. Is the only solution to increase the value of n (a solution which is not always ideal because it almost directly leads to overfitting)?

Reading questions

- I'm wondering if there are n-gram models that start by looking at the last 5 words, and then if the phrase doesn't exist in the training data, go down to 4 words, or 3 words, and so on. This seems like it wouldn't be too complicated and might work better.

- Are n-grams ever used to calculate the probability of several words/a phrase coming next rather than one word at a time?

Reading questions

- I had homework for another course on this exact topic and we implemented all kinds of smoothing from Laplace, ML, Ney-Essen, Witten-Bell. We spent literally so much time on difference methods on giving unseen data some probability using the "better" method, but python tells me that Maximum Likelihood gave the best score on test data. Isn't Maximum Likelihood the naivest method, and why does it work so well?

- I'm also confused about how perplexity yields different results to probability in use--could we get an example of this?

Reading questions

- I am a bit confused about stupid backoff as a method for smoothing. Giving up the idea of making the language model a real probability distribution seems like defeating the purpose to me, since n-grams models supposedly utilizes probability in the first place. Why is this algorithm especially good for very large language models and in what ways is it not good enough for smaller ones?

Reading questions

- How exactly can/do n-gram models account for contextual/cultural information that affects the probabilities of words in a sequence?

NLP/Compling in the news

- https://www.technologyreview.com/2023/04/19/1071789/openais-hunger-for-data-is-coming-back-to-bite-it/