CMPT-413: Computational Linguistics

Anoop Sarkar
anoop@cs.sfu.ca
www.sfu.ca/~anoop/courses/CMPT-413-Spring-2003.html

n-grams

- A simple model of language, applying a simple probability model to *rank* sentences as being generated by the same source as the training data.

- One example we've seen before: *spelling correction*
  . . . was called a "stellar and versatile **acress** *whose combination of sass and glamour has defined her . . .*
  $\longrightarrow$ without the context around the word the best guess was **acres**

n-grams

- Let's take another view and take all the different sentences we can form with the various candidates:
  . . . was called a "stellar and versatile **acress** *whose combination of sass and glamour has defined her . . .*
  . . . was called a "stellar and versatile **acres** *whose combination of sass and glamour has defined her . . .*
  . . . was called a "stellar and versatile **actress** *whose combination of sass and glamour has defined her . . .*

- Each sentence is a sequence of words: $w_1, \ldots, w_n$
  Use a probability model to find the most likely sequence of words: $P(w_1, \ldots, w_n)$

n-grams

- Words in different orders, which sequence is most likely?

1. *physical Brainpower not plant is chief , now a ’s asset , . firm*
2. *, a Brainpower not now chief asset firm ’s is . plant physical ,*
3. *chief a physical , . firm not , Brainpower plant is asset ’s now*
4. *not plant Brainpower now physical ’s . a chief , asset firm , is*
5. *plant Brainpower is now , , not . firm a ’s physical asset chief*
6. *physical is ’s plant firm not chief . Brainpower now asset , , a*
7. *Brainpower , not physical plant , is now a firm ’s chief asset .*

Probability: The Chain Rule

- $P(e_1, e_2, \ldots, e_n) = P(e_1) \times P(e_2 \mid e_1) \times P(e_3 \mid e_1, e_2) \ldots$
- $P(w_1, w_2, \ldots, w_n) = P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1, w_2) \times P(w_4 \mid w_1, w_2, w_3) \times \ldots \times P(w_n \mid w_1, \ldots, w_{n-1})$
- Each of these probabilities can be estimated from *training data*. However, we need to use these probabilities on *unseen* data.
- $P(w_n \mid w_1, \ldots, w_{n-1})$ or even $P(w_4 \mid w_1, w_2, w_3)$ can lead to zeroes.

The Markov Assumption

- $P(w_1, w_2, \ldots, w_n) = P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1, w_2) \times P(w_4 \mid w_1, w_2, w_3) \times \ldots \times P(w_n \mid w_1, \ldots, w_{n-1})$

- The Markov assumption is to use a fixed finite history, for example, only use two previous words in the conditioning context:
  $P(w_1, w_2, \ldots, w_n) =$
  $P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1, w_2) \times P(w_4 \mid w_2, w_3) \times$
  $\ldots P(w_i \mid w_{i-2}, w_{i-1}) \ldots \times P(w_n \mid w_{n-2}, \ldots, w_{n-1})$

The Markov Assumption

- If we use one word in the conditioning context, we have two words in the probability model so we call a **bigram** model.
  Similarly if we have two words in the conditioning context, we call it a **trigram** model.

- Predict each word based on a finite local context. Now we can use the context, although in a limited way.
  even a trigram or bigram probability model will contain zeroes: why?

Trigram Models

- Let's take an example sentence and *generate* it using a trigram model:
  $S =$ *a stellar and versatile* **acres** *whose combination of sass and glamour has defined her*

- $P(S) = P(a) \times P(stellar \mid a) \times P(and \mid a, stellar) \times$
  $P(versatile \mid stellar, and) \times P(\textbf{acres} \mid and, versatile) \times$
  $P(whose \mid versatile, \textbf{acres}) \times P(combination \mid \textbf{acres}, whose) \times$
  $P(of \mid whose, combination) \times \ldots$

- What will $P(S)$ be if we replace **acres** with **actress** or with **acress**?
  $P(S)$ is called a *language model*

The Shannon Game

- Predict the next word after looking at a prefix of the sentence or utterance:

$$\ldots \textit{was called a stellar and versatile} \begin{Bmatrix} \textit{acres} \\ \textit{actress} \\ \textit{acress} \\ \textit{program} \\ \textit{premier} \\ \ldots \end{Bmatrix} \ldots$$

- Construct a sentence that can fool a trigram model.

The Shannon Game

- *We will* **either** *go to watch a movie* **or** *go out to eat.*
- **If** *I can get through this week* **then** *I can relax.*
- . . .

Word prediction and the Markov Assumption

- Consider the sentence:
  The contract ended with a loss of 7 cents after . . .

- Compare the trigram *7, cents, after* with the trigram model that would skip some words to get a trigram *contract, ended, after*
  The latter should be a better predictor of the word *after*

n-grams

- Let's count how many possible **parameter values** we can have in our n-gram probability models for different values of n

- Total number of words: **word tokens**;

- Total number of unique words: **word types** is the vocabulary size;

n-grams

- Let $\mathcal{V}$ be the vocabulary set, then the size of the vocabulary of the total number of *word types* is $\mid \mathcal{V} \mid$:
  - $P(W_i = x)$: how many different values for W_i
  - $P(W_i = y \mid W_j = x)$: how many different values of W_i, W_j
  - $P(W_k = z \mid W_i = x, W_j = y)$: how many different values of W_i, W_j, W_k

n-grams

- Let $\mathcal{V}$ be the vocabulary set, then the size of the vocabulary of the total number of *word types* is $\mid \mathcal{V} \mid$:
  - $P(W_i = x)$: how many different values for W_i; 3×10^3
  - $P(W_i = y \mid W_j = x)$: how many different values of W_i, W_j; 9×10^6
  - $P(W_k = z \mid W_i = x, W_j = y)$: how many different values of W_i, W_j, W_k; 27×10^9

Jane Austen

Jane Austen: 1775-1817

- Three novels by Jane Austen:
  Emma, *Sense and Sensibility* and *Pride and Prejudice*

- Removed punctuation and kept paragraph structure

- Train a trigram model on this text

Jane Austen

f(3gram)	f(2gram)	f(1gram)	w_0	w_1	w_2
378	518	10381	I	do	not
366	1366	10381	I	am	sure
214	1917	9182	in	the	world
202	572	6917	she	could	not
189	462	2751	would	have	been
174	184	10381	I	dare	say
173	179	5758	as	soon	as
173	357	11135	a	great	deal
171	332	7573	it	would	be
155	945	3017	could	not	be

Jane Austen

3gram $\frac{f(w_0,w_1,w_3)}{f(w_0,w_1)}$	2gram $\frac{f(w_0,w_1)}{f(w_0)}$	1gram $\frac{f(w_0)}{N}$	w_0	w_1	w_2
0.72	0.04	0.016	I	do	not
0.26	0.13	0.016	I	am	sure
0.11	0.20	0.014	in	the	world
0.35	0.08	0.011	she	could	not
0.40	0.16	0.004	would	have	been
0.94	0.01	0.016	I	dare	say
0.96	0.03	0.009	as	soon	as
0.48	0.03	0.018	a	great	deal
0.51	0.04	0.012	it	would	be
0.16	0.31	0.004	could	not	be

Jane Austen

f(3gram)	f(2gram)	f(1gram)	w_0	w_1	w_2
1	1	1	favor	of	your
1	1	1	peerage	his	wealth
1	1	1	stagnation	Mrs	Elton’s
1	1	1	genteelly	and	paid
1	1	1	adept	in	the
1	1	1	deckers	now	in
1	1	1	oracle	Fanny’s	explanations
1	1	1	Ashworth	is	too
1	1	1	puddles	with	impatient
1	1	1	Harringtons	to	come
1	1	1	roasted	No	coffee
1	1	1	coherent	Dearest	Lizzy

Jane Austen

3gram	2gram	1gram	w_0	w_1	w_2
0.00039	0.13	0.029	of	the	Middletons
0.00039	0.13	0.029	of	the	Meryton
0.00039	0.13	0.029	of	the	Lucases
0.00039	0.13	0.029	of	the	London
0.00039	0.13	0.029	of	the	Irish
0.00039	0.13	0.029	of	the	History
0.00039	0.13	0.029	of	the	First
0.00039	0.13	0.029	of	the	Esquire
0.00039	0.13	0.029	of	the	Elegant
0.00039	0.13	0.029	of	the	Dashwoods
0.00039	0.13	0.029	of	the	Crown

How good is a model

- So far we've seen the probability of a sentence: $P(w_0, \ldots, w_n)$
- What is the probability of a collection of sentences, that is what is the probability of a corpus
- Let $T = s_0, \ldots, s_m$ be a text corpus with sentences s_0 through s_m
- What is $P(T)$?
  Let us assume that we trained $P(\cdot)$ on some *training data*, and T is the *test data*

How good is a model

- $T = s_0, \ldots, s_m$ is the text corpus with sentences s_0 through s_m
- $P(T) = \prod_{i=0}^{m} P(s_i)$
- $P(s_i) = P(w_0^i, \ldots, w_n^i)$
  Let W_T be the length of the text T measured in words
- Cross entropy for T: $H(T) = -\frac{1}{W_T}\log_2 P(T)$
  the average number of bits needed to encode each of the W_T words in the *test data*
  Perplexity: $PP(T) = 2^{H(T)}$

How good is a model

- Lower cross entropy values and perplexity values are better
  Lower values mean that the model is *better*
  Correlation with performance of the language model in various applications

- Performance of a language model is its cross-entropy or perplexity on *test data* (unseen data)
  corresponds to the number bits required to encode that data

- On various real life datasets, typical perplexity values yielded by n-gram models on English text range from about 50 to almost 1000 (corresponding to cross entropies from about 6 to 10 bits/word)

Bigram Models

- In practice:
  P(Mork read a book) $= P$(Mork $\mid$ $< start >$) $\times P$(read $\mid$ Mork) $\times$
  P(a $\mid$ read) $\times P$(book $\mid$ a) $\times P(< stop >$ $\mid$ book)

- $P(w_i \mid w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$
  On unseen data, $c(w_i, w_{i-1})$ or worse $c(w_{i-1})$ could be zero

$$\sum_{w_i} \frac{c(w_i, w_{i-1})}{c(w_{i-1})} = ?$$

Smoothing

- **Smoothing** deals with events that have been observed zero times

  Best summary of smoothing methods for language models is: *An Empirical Study of Smoothing Techniques for Language Modeling* by Stanley Chen and Joshua Goodman.

- Smoothing algorithms also tend to improve the accuracy of the model

$$P(w_i \mid w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$$

- Not just unobserved events: what about events observed once?

Add-one Smoothing

$$P(w_i \mid w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$$

- Add-one Smoothing:

$$P(w_i \mid w_{i-1}) = \frac{1 + c(w_i, w_{i-1})}{V + c(w_{i-1})}$$

- Let V be the number of words in our vocabulary
  Remember that we *observe* only V many bigrams
  Assigns count of 1 to unseen bigrams

Add-one Smoothing

$P(\text{Mindy read a book}) = P(\text{Mindy} \mid < start >) \times P(\text{read} \mid \text{Mindy}) \times P(\text{a} \mid \text{read}) \times P(\text{book} \mid \text{a}) \times P(< stop > \mid \text{book})$

- Without smoothing:

$$P(\text{read} \mid \text{Mindy}) = \frac{c(\text{Mindy, read})}{c(\text{Mindy})} = 0$$

- With add-one smoothing (assuming c(Mindy) = 1 but c(Mindy, read) = 0):

$$P(\text{read} \mid \text{Mindy}) = \frac{1}{V + 1}$$

Additive Smoothing: (Lidstone 1920, Jeffreys 1948)

$$P(w_i \mid w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$$

- Add-one smoothing works horribly in practice. Seems like 1 is too large a count for unobserved events.

- Additive Smoothing:

$$P(w_i \mid w_{i-1}) = \frac{\delta + c(w_i, w_{i-1})}{(\delta \times V) + c(w_{i-1})}$$

- $0 < \delta \leq 1$
  Still works horribly in practice, but better than add-one smoothing.

Good-Turing Smoothing: (Good, 1953)

$$P(w_i \mid w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$$

- Imagine you're sitting at a sushi bar with a conveyor belt.
- You see going past you 10 plates of tuna, 3 plates of unagi, 2 plates of salmon, 1 plate of shrimp, 1 plate of octopus, and 1 plate of yellowtail
- How likely are you to see a new kind of seafood appear: $\frac{3}{18}$
- How likely are you to see another plate of salmon: should be $< \frac{2}{18}$

Good-Turing Smoothing

- How many types of seafood (words) were seen once? Use this to predict probabilities for unseen events
  Let n_1 be the number of events that occurred once: $p_0 = \frac{n_1}{N}$

- The Good-Turing estimate states that for any n-gram that occurs r times, we should pretend that it occurs r^* times

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

Good-Turing Smoothing

- 10 tuna, 3 unagi, 2 salmon, 1 shrimp, 1 octopus, 1 yellowtail

- How likely is new data? Let n_1 be the number of items occurring once, which is 3 in this case. N is the total, which is 18.

$$p_0 = \frac{n_1}{N} = \frac{3}{18}$$

Good-Turing Smoothing

- 10 tuna, 3 unagi, 2 salmon, 1 shrimp, 1 octopus, 1 yellowtail
- How likely is *octopus*? Since $c(\textit{octopus}) = 1$ The GT estimate is 1^*.

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

- To compute 1^*, we need $n_1 = 3$ and $n_2 = 1$

$$1^* = 2 \times \frac{1}{3} = \frac{2}{3}$$

- What happens when $n_r = 0$?

Simple Backoff Smoothing: incorrect version

$$P(w_i \mid w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$$

- In add-one or Good-Turing: $P(\text{the} \mid \text{string}) = P(\text{Fonz} \mid \text{string})$
- If $c(w_i, w_{i-1}) = 0$, then use $P(w_{i-1})$ (back off)
- Works for trigrams: back off to bigrams and then unigrams
- Works better in practice, but probabilities get mixed up (unseen bigrams, for example will get higher probabilities than seen bigrams)

Backoff Smoothing: Jelinek-Mercer Smoothing

$$P_{ML}(w_i \mid w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$$

- $P_{JM}(w_i \mid w_{i-1}) = \lambda P_{ML}(w_i \mid w_{i-1}) + (1 - \lambda) P_{ML}(w_i)$
  where, $0 \leq \lambda \leq 1$

- Notice that $P_{JM}(\text{the} \mid \text{string}) > P_{JM}(\text{Fonz} \mid \text{string})$ as we wanted

- Jelinek-Mercer (1980) describe an elegant form of this **interpolation**:

$$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1 - \lambda) P_{JM}(n - 1\text{gram})$$

- What about $P_{JM}(w_i)$?

Backoff Smoothing: Many alternatives

$$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1-\lambda) P_{JM}(n-1\text{gram})$$

- Different methods for finding the values for λ correspond to variety of different smoothing methods
  - Katz Backoff (include Good-Turing with Backoff Smoothing)

$$P_{katz}(y \mid x) = \begin{cases} \frac{c^*(xy)}{c(x)} & \text{if } c(xy) > 0 \\ \alpha(x) P_{katz}(y) & \text{otherwise} \end{cases}$$

  - Deleted Interpolation (Jelinek, Mercer)
    compute λ values from **held-out** data

Backoff Smoothing: Many alternatives

$$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1-\lambda) P_{JM}(n-1\text{gram})$$

- Witten-Bell smoothing
  use the $n-1$gram model when the $n$gram model has too few unique words *in the $n$gram context*

- Absolute discounting (Ney, Essen, Kneser)

$$P_{abs}(y \mid x) = \begin{cases} \frac{c(xy)-D}{c(x)} & \text{if } c(xy) > 0 \\ \alpha(x) P_{abs}(y) & \text{otherwise} \end{cases}$$

Backoff Smoothing: Many alternatives

$$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1-\lambda) P_{JM}(n-1\text{gram})$$

- Kneser-Ney smoothing
  $P(\text{Francisco} \mid \text{eggplant}) > P(\text{stew} \mid \text{eggplant})$
  - *Francisco* is common, so interpolation gives $P(\text{Francisco} \mid \text{eggplant})$ a high value
  - But *Francisco* occurs in few contexts (only after *San*)
  - *stew* is common, **and** occurs in many contexts
  - Hence weight the interpolation based on number of contexts for the word using discounting

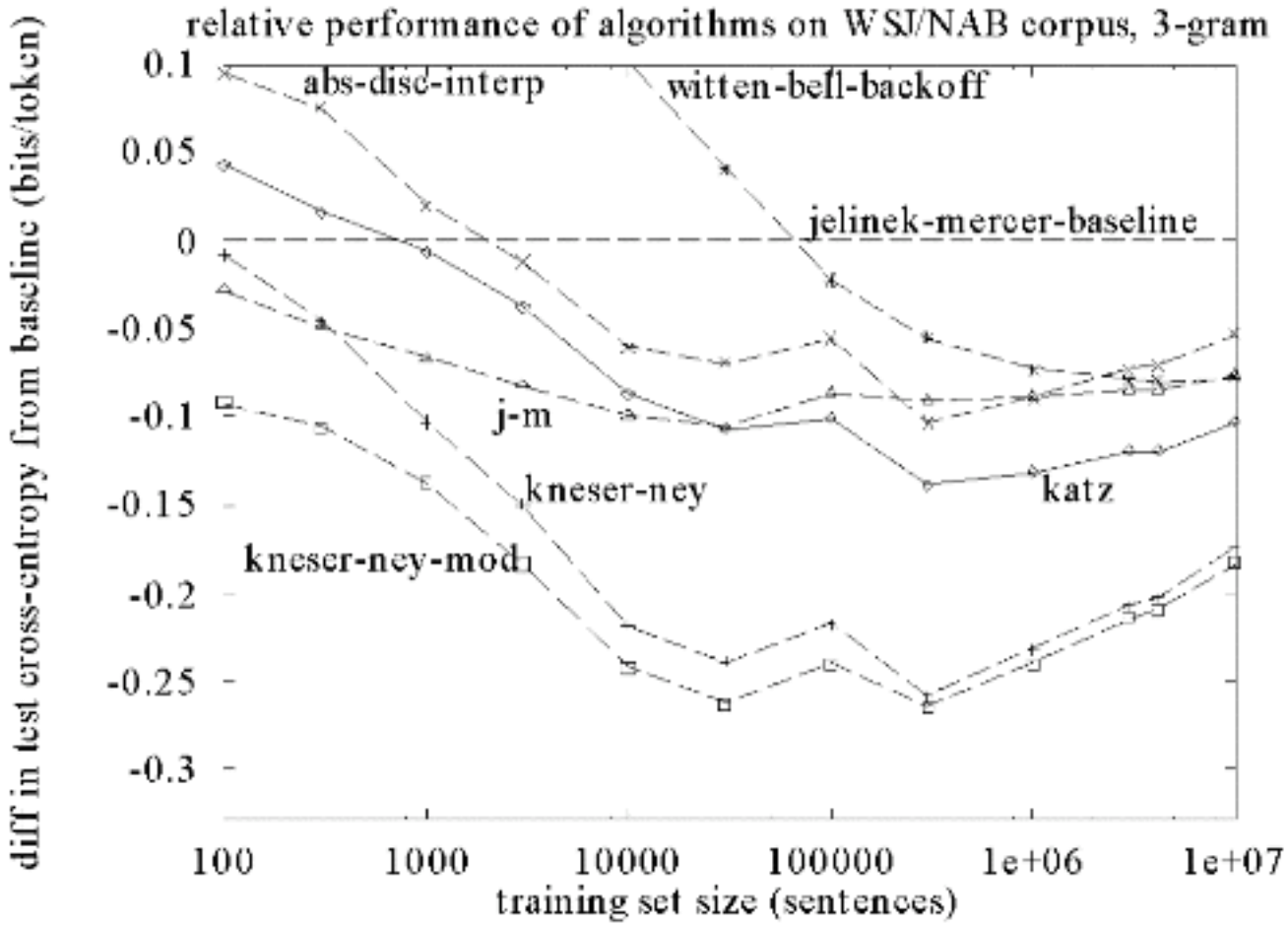
Backoff Smoothing: Many alternatives

$$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1-\lambda) P_{JM}(n-1\text{gram})$$

- Modified Kneser-Ney smoothing (Chen and Goodman)
  multiple discounts for one count, two counts and three or more counts

- Generalized search (Powell search) or the Expectation-Maximization algorithm

Relative Performance: (Goodman and Chen, 1998)



Trigram Models

- Revisiting the trigram model:
  $P(w_1, w_2, \ldots, w_n) =$
  $P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1, w_2) \times P(w_4 \mid w_2, w_3) \times$
  $\ldots P(w_i \mid w_{i-2}, w_{i-1}) \ldots \times P(w_n \mid w_{n-2}, \ldots, w_{n-1})$

- Notice that the length of the sentence n is variable

- What is the event space?

The `stop` symbol

- Let $\Sigma = \{a, b\}$ and the language be Σ^*
  so $L = \{\epsilon, a, b, aa, bb, ab, bb \ldots\}$

- Consider a unigram model: $P(a) = P(b) = 0.5$

- $P(a) = 0.5$, $P(b) = 0.5$, $P(aa) = 0.5^2 = 0.25$, $P(bb) = 0.25$
  and so on.

- But $P(a) + P(b) + P(aa) + P(bb) = 1.5$!!

$$\sum_w P(w) = 1$$

The `stop` symbol

- What went wrong?
  No probability for $P(\epsilon)$

- Add a special `stop` symbol:

$$P(a) = P(b) = 0.25$$

$$P(\texttt{stop}) = 0.5$$

- $P(\texttt{stop}) = 0.5$, $P(a\ \texttt{stop}) = P(b\ \texttt{stop}) = 0.25 \times 0.5 = 0.125$, $P(aa\ \texttt{stop}) = 0.25^2 \times 0.5 = 0.03125$ (now the sum is no longer greater than one)

The `stop` symbol

- With this new `stop` symbol we can show that $\sum_w P(w) = 1$
  Notice that the probability of any sequence of length n is $0.25^n \times 0.5$
  Also there are 2^n sequences of length n

$$\sum_w P(w) =$$

$$\sum_{n=0}^{\infty} 2^n \times 0.25^n \times 0.5$$

$$\sum_{n=0}^{\infty} 0.5^n \times 0.5 = \sum_{n=0}^{\infty} 0.5^{n+1}$$

$$\sum_{n=1}^{\infty} 0.5^n = 1$$