Advanced smoothing algorithms

Some Examples

- Good-Turing
- Kneser-Ney

Advanced smoothing algorithms

Some Examples

- Good-Turing
- Kneser-Ney

Good-Turing: Basic Intuition

Use the count of things we have see once

- to help estimate the count of things we have never seen

N_c: Frequency of frequency c

Example Sentences

<s>I am here </s>
<s>who am I </s>
<s>I would like </s>

N_c: Frequency of frequency c

Example Sentences

<s>I am here </s>

<s>who am I </s>

<s>I would like </s>

Computing N_c

I	3
am	2
here	1
who	1
would	1
like	1

$N_1 = 4$

$N_2 = 1$

$N_3 = 1$

Good Turing Estimation

Idea

- Reallocate the probability mass of $n-$grams that occur $r+1$ times in the training data to the $n-$grams that occur r times
- In particular, reallocate the probability mass of $n-$grams that were seen once to the $n-$grams that were never seen

Adjusted count

For each count c, an adjusted count c^* is computed as:

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

where N_c is the number of $n-$grams seen exactly c times

Good Turing Estimation

Good Turing Smoothing

P^*_{GT}(things with frequency c) = $\frac{c^*}{N}$

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

Good Turing Estimation

Good Turing Smoothing

P^*_{GT}(things with frequency c) = $\frac{c^*}{N}$

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

What if $c = 0$

P^*_{GT}(things with frequency c) = $\frac{N_1}{N}$ where N denotes the total number of bigrams that actually occur in training

Complications

What about words with high frequency?

- For small c, $N_c > N_{c+1}$
- For large c, too jumpy

Complications

What about words with high frequency?

- For small c, $N_c > N_{c+1}$
- For large c, too jumpy

Simple Good-Turing

Replace empirical N_k with a best-fit power law once counts get unreliable

Good-Turing numbers: Example

22 million words of AP Neswire

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

Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Good-Turing numbers: Example

22 million words of AP Neswire

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

It looks like $c^* = c - 0.75$

Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Absolute Discounting Interpolation

Why don't we just substract 0.75 (or some d)?

Absolute Discounting Interpolation

Why don't we just substract 0.75 (or some d)?

$$P_{AbsoluteDiscounting}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

Absolute Discounting Interpolation

Why don't we just substract 0.75 (or some d)?

$$P_{AbsoluteDiscounting}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

We may keep some more values of d for counts 1 and 2

Absolute Discounting Interpolation

Why don't we just substract 0.75 (or some d)?

$$P_{AbsoluteDiscounting}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

We may keep some more values of d for counts 1 and 2

But can we do better than using the regular unigram correct?

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ....*: glasses/Francisco?

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"
- *But "Francisco" mostly follows "San"*

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"
- *But "Francisco" mostly follows "San"*

$P(w)$: *"How likely is w?"*

Instead, $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?"

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"
- *But "Francisco" mostly follows "San"*

$P(w)$: "How likely is w?"

Instead, $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?"

- For each word, count the number of bigram types it completes
- Every bigram type was a novel continuation the first time it was seen

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"
- *But "Francisco" mostly follows "San"*

$P(w)$: *"How likely is w?"*

Instead, $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?"

- For each word, count the number of bigram types it completes
- Every bigram type was a novel continuation the first time it was seen

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

$$P_{continuation}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

$$P_{continuation}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}$$

A frequent word (Francisco) occurring in only one context (San) will have a low continuation probability

Kneser-Ney Smoothing

$$P_{KN}(w_i|w_{i-1}) = \frac{max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1})P_{continuation}(w_i)$$

Kneser-Ney Smoothing

$$P_{KN}(w_i|w_{i-1}) = \frac{max(c(w_{i-1},w_i)-d,0)}{c(w_{i-1})} + \lambda(w_{i-1})P_{continuation}(w_i)$$

λ is a normalizing constant

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})}|\{w : c(w_{i-1},w) > 0\}|$$

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases
- But the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases
- But the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).

A general approach is to combine the results of multiple N-gram models.

Backoff and Interpolation

It might help to use less context

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff

- use trigram if you have good evidence
- otherwise bigram, otherwise unigram

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff

- use trigram if you have good evidence
- otherwise bigram, otherwise unigram

Interpolation

mix unigram, bigram, trigram

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probbaility $P(w_i|w_{i-1})$

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probbaility $P(w_i|w_{i-1})$
- If we do not have counts to compute $P(w_i|w_{i-1})$, estimate this using the unigram probability $P(w_i)$

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probbaility $P(w_i|w_{i-1})$
- If we do not have counts to compute $P(w_i|w_{i-1})$, estimate this using the unigram probability $P(w_i)$

$P_{bo}(w_i|w_{i-2}w_{i-1})$=

- $\hat{P}(w_i|w_{i-2}w_{i-1})$, if $c(w_{i-2}w_{i-1}w_i) > 0$
- $\lambda(w_{i-1}w_{i-2})P_{bo}(w_i|w_{i-1})$, otherwise

where $P_{bo}(w_i|w_{i-1})$ =

- $\hat{P}(w_i|w_{i-1})$ if $c(w_{i-1}w_i) > 0$
- $\lambda(w_{n-1})\hat{P}(w_n)$, otherwise

Linear Interpolation

Simple Interpolation

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

Linear Interpolation

Simple Interpolation

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

$$\sum_i \lambda_i = 1$$

Linear Interpolation

Simple Interpolation

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

$$\sum_i \lambda_i = 1$$

Lambdas conditional on context

$$\tilde{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1(w_{n-2},w_{n-1}) P(w_n|w_{n-1}w_{n-2})$$
$$+\lambda_2(w_{n-2},w_{n-1}) P(w_n|w_{n-1}) + \lambda_3(w_{n-2},w_{n-1}) P(w_n)$$

Setting the lambda values

Use a held-out corpus

Choose λs to maximize the probability of held-out data:

- Find the N-gram probabilities on the training data
- Search for λs that give the largest probability to held-out data