Language Modeling

Yoav Goldberg

with slides from Michael Collins, Noah Smith, Yoav Artzi.

Goal: assign a probability distribution over sentences

p("he saw her duck")

We would like..

p(אכלתי את התפוח) > p(את אכלתי התפוח)

p(שני סוסים גדולים) > p(שני סוסים גדול)

p(שני סוסים חומים) > p(שני סוסים ירוקים)

p(הלכתי הביתה) > p(הלכתי בית)

p(הוא רוצה מילקי) > p(הוא חושק במילקי)

The Language Modeling Problem

- We have some (finite) vocabulary, say $\mathcal{V} = \{$the, a, man, telescope, Beckham, two, $\ldots\}$

- We have an (infinite) set of strings, $\mathcal{V}^{\dagger}$

  the STOP
  a STOP
  the fan STOP
  the fan saw Beckham STOP
  the fan saw saw STOP
  the fan saw Beckham play for Real Madrid STOP

The Language Modeling Problem (Continued)

- We have a *training sample* of example sentences in English

The Language Modeling Problem (Continued)

- We have a *training sample* of example sentences in English
- We need to "learn" a probability distribution p i.e., p is a function that satisfies

$$\sum_{x \in \mathcal{V}^\dagger} p(x) = 1, \quad p(x) \geq 0 \text{ for all } x \in \mathcal{V}^\dagger$$

The Language Modeling Problem (Continued)

- We have a *training sample* of example sentences in English
- We need to "learn" a probability distribution p i.e., p is a function that satisfies

$$\sum_{x \in \mathcal{V}^\dagger} p(x) = 1, \quad p(x) \geq 0 \text{ for all } x \in \mathcal{V}^\dagger$$

$p(\text{the STOP}) = 10^{-12}$
$p(\text{the fan STOP}) = 10^{-8}$
$p(\text{the fan saw Beckham STOP}) = 2 \times 10^{-8}$
$p(\text{the fan saw saw STOP}) = 10^{-15}$
$\ldots$
$p(\text{the fan saw Beckham play for Real Madrid STOP}) = 2 \times 10^{-9}$
$\ldots$

Why would we like to do this?

- Central in many NLP applications
    - Speech Recognition.
    - Machine Translation.
    - Spelling Correction.
    - OCR.
    - Summarization.
    - ... whenever we "generate" text.

Why would we like to do this?

- Central in many NLP applications
  - Speech Recognition.
    recognize speech / wreck a nice beach
  - Machine Translation.
  - Spelling Correction.
  - OCR.
  - Summarization.
  - ... whenever we "generate" text.

Why would we like to do this?

- Central in many NLP applications
  - Speech Recognition.
    recognize speech / wreck a nice beach
  - Machine Translation.
    he went home -> הוא הלך בית / הוא הלך הביתה
  - Spelling Correction.
  - OCR.
  - Summarization.
  - ... whenever we "generate" text.

Why would we like to do this?

- Central in many NLP applications
    - Speech Recognition.
      recognize speech / wreck a nice beach
    - Machine Translation.
      he went home -> הוא הלך בית / הוא הלך הביתה
    - Spelling Correction.
      the office is fifteen **minuets** from here
    - OCR.
    - Summarization.
    - ... whenever we "generate" text.

Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, X and Y:

$$\boxed{\text{source}} \longrightarrow Y \longrightarrow \boxed{\text{channel}} \longrightarrow X$$

Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, X and Y:

$$\boxed{\text{source}} \longrightarrow \underset{\textbf{ideal}}{Y} \longrightarrow \boxed{\text{channel}} \longrightarrow X$$

- Y is the plaintext, the true message, the missing information, the output

Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, X and Y:

$$\boxed{\text{source}} \longrightarrow \underset{\text{ideal}}{Y} \longrightarrow \boxed{\text{channel}} \longrightarrow \underset{\text{what we see}}{X}$$

- Y is the plaintext, the true message, the missing information, the output
- X is the ciphertext, the garbled message, the observable evidence, the input

Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, X and Y:

$$\boxed{\text{source}} \longrightarrow \underset{\textbf{ideal}}{Y} \longrightarrow \boxed{\text{channel}} \longrightarrow \underset{\textbf{what we see}}{X}$$

- Y is the plaintext, the true message, the missing information, the output
- X is the ciphertext, the garbled message, the observable evidence, the input
- Decoding: select y given $X = x$.

$$\begin{aligned} y^* &= \operatorname*{argmax}_y p(y \mid x) \\ &= \operatorname*{argmax}_y \frac{p(x \mid y) \cdot p(y)}{p(x)} \\ &= \operatorname*{argmax}_y \underbrace{p(x \mid y)}_{\text{channel model}} \cdot \underbrace{p(y)}_{\text{source model}} \end{aligned}$$

Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, X and Y:

$$\boxed{\text{source}} \longrightarrow \underset{\textbf{ideal}}{Y} \longrightarrow \boxed{\text{channel}} \longrightarrow \underset{\textbf{what we see}}{X}$$

- Y is the plaintext, the true message, the missing information, the output
- X is the ciphertext, the garbled message, the observable evidence, the input
- Decoding: select y given $X = x$.

$$\begin{aligned} y^* &= \operatorname*{argmax}_y p(y \mid x) \\ &= \operatorname*{argmax}_y \frac{p(x \mid y) \cdot p(y)}{p(x)} \\ &= \operatorname*{argmax}_y \underbrace{p(x \mid y)}_{\text{channel model}} \cdot \underbrace{p(y)}_{\text{source model}} \end{aligned}$$

Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, X and Y:

$$\boxed{\text{source}} \longrightarrow \underset{\textbf{text}}{Y} \longrightarrow \boxed{\text{channel}} \longrightarrow \underset{\textbf{acoustic wave}}{X}$$

- Y is the plaintext, the true message, the missing information, the output
- X is the ciphertext, the garbled message, the observable evidence, the input
- Decoding: select y given $X = x$.

$$\begin{aligned} y^* &= \operatorname*{argmax}_y p(y \mid x) \\ &= \operatorname*{argmax}_y \frac{p(x \mid y) \cdot p(y)}{p(x)} \\ &= \operatorname*{argmax}_y \underbrace{p(x \mid y)}_{\text{channel model}} \cdot \underbrace{p(y)}_{\text{source model}} \end{aligned}$$

Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, X and Y:

$$\boxed{\text{source}} \longrightarrow \underset{\text{"good text in English"}}{Y} \longrightarrow \boxed{\text{channel}} \longrightarrow \underset{\text{"corrupted text in French"}}{X}$$

- Y is the plaintext, the true message, the missing information, the output
- X is the ciphertext, the garbled message, the observable evidence, the input
- Decoding: select y given $X = x$.

$$\begin{aligned} y^* &= \operatorname*{argmax}_y p(y \mid x) \\ &= \operatorname*{argmax}_y \frac{p(x \mid y) \cdot p(y)}{p(x)} \\ &= \operatorname*{argmax}_y \underbrace{p(x \mid y)}_{\text{channel model}} \cdot \underbrace{p(y)}_{\text{source model}} \end{aligned}$$

Motivation: Noisy Channel Models

A pattern for modeling a pair of random variables, X and Y:

$$\boxed{\text{source}} \longrightarrow \underset{\text{good text in English}}{Y} \longrightarrow \boxed{\text{channel}} \longrightarrow \underset{\text{text with spelling errors}}{X}$$

- Y is the plaintext, the true message, the missing information, the output
- X is the ciphertext, the garbled message, the observable evidence, the input
- Decoding: select y given $X = x$.

$$\begin{aligned} y^* &= \operatorname*{argmax}_y p(y \mid x) \\ &= \operatorname*{argmax}_y \frac{p(x \mid y) \cdot p(y)}{p(x)} \\ &= \operatorname*{argmax}_y \underbrace{p(x \mid y)}_{\text{channel model}} \cdot \underbrace{p(y)}_{\text{source model}} \end{aligned}$$

Noisy Channel Example: Speech Recognition

$$\boxed{\text{source}} \longrightarrow \text{sequence in } \mathcal{V}^\dagger \longrightarrow \boxed{\text{channel}} \longrightarrow \text{acoustics}$$

- Acoustic model defines $p(\text{sounds} \mid \boldsymbol{x})$ (channel)
- Language model defines $p(\boldsymbol{x})$ (source)

Noisy Channel Example: Speech Recognition

Credit: Luke Zettlemoyer

word sequence	$\log p(\text{acoustics} \mid \text{word sequence})$
the station signs are in deep in english	-14732
the stations signs are in deep in english	-14735
the station signs are in deep into english	-14739
the station 's signs are in deep in english	-14740
the station signs are in deep in the english	-14741
the station signs are indeed in english	-14757
the station 's signs are indeed in english	-14760
the station signs are indians in english	-14790
the station signs are indian in english	-14799
the stations signs are indians in english	-14807
the stations signs are indians and english	-14815

Noisy Channel Example: Machine Translation

Also knowing nothing official about, but having guessed and inferred considerable about, the powerful new mechanized methods in cryptography—methods which I believe succeed even when one does not know what language has been coded—one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."

Warren Weaver, 1955

The Language Modeling Problem

- We have some (finite) vocabulary, say $\mathcal{V} = \{$the, a, man, telescope, Beckham, two, $\ldots\}$

- We have an (infinite) set of strings, $\mathcal{V}^{\dagger}$

  the STOP
  a STOP
  the fan STOP
  the fan saw Beckham STOP
  the fan saw saw STOP
  the fan saw Beckham play for Real Madrid STOP

Is "finite $\mathcal{V}$" realistic?

Is “finite $\mathcal{V}$” realistic?

No

Is "finite $\mathcal{V}$" realistic?

No

no

n0

-no

notta

№

/no

//no

(no

|no

The Language Modeling Problem (Continued)

- We have a *training sample* of example sentences in English
- We need to "learn" a probability distribution p i.e., p is a function that satisfies

$$\sum_{x \in \mathcal{V}^\dagger} p(x) = 1, \quad p(x) \geq 0 \text{ for all } x \in \mathcal{V}^\dagger$$

A Naive Method

- We have N training sentences
- For any sentence $x_1 \ldots x_n$, $c(x_1 \ldots x_n)$ is the number of times the sentence is seen in our training data
- A naive estimate:

$$p(x_1 \ldots x_n) = \frac{c(x_1 \ldots x_n)}{N}$$

A Naive Method

- We have N training sentences
- For any sentence $x_1 \ldots x_n$, $c(x_1 \ldots x_n)$ is the number of times the sentence is seen in our training data
- A naive estimate:

$$p(x_1 \ldots x_n) = \frac{c(x_1 \ldots x_n)}{N}$$

does this satisfy the probability requirements?

A Naive Method

- We have N training sentences
- For any sentence $x_1 \ldots x_n$, $c(x_1 \ldots x_n)$ is the number of times the sentence is seen in our training data
- A naive estimate:

$$p(x_1 \ldots x_n) = \frac{c(x_1 \ldots x_n)}{N}$$

do you see a problem with this?

Predicting the next word

- We can either compute the probability of a sequence:
  $p(w1, w2, w3, w4, w5, w6)$

- Or the probability of the next item given a prefix:
  $p(w6 \mid w1, w2, w3, w4, w5)$

- These are equivalent (why?)

- A model that computes any of these is called a Language Model (LM)

Aside: Language Modeling as Perfect AI

- If we can always predict the next word as good as a human could, we have achieved human-level intelligence.

  (why?)

A Naive Method

Reminder:

- We have N training sentences
- For any sentence $x_1 \ldots x_n$, $c(x_1 \ldots x_n)$ is the number of times the sentence is seen in our training data
- A naive estimate:

$$p(x_1 \ldots x_n) = \frac{c(x_1 \ldots x_n)}{N}$$

do you see a problem with this?

lets take the next-word-prediction view

Using the chain rule?

$$\begin{aligned} p(w_1, w_2, ..., w_n) = & p(w_1) \\ & \times p(w_2|w_1) \\ & \times p(w_3|w1, w2) \\ & \times p(w_4|w_1, w_2, w_3) \\ & ... \\ & \times p(w_n|w1, w2, ..., w_{n-1}) \end{aligned}$$

Using the chain rule?

$$\begin{aligned} p(w_1, w_2, ..., w_n) = & p(w_1) \\ & \times p(w_2|w_1) \\ & \times p(w_3|w1, w2) \\ & \times p(w_4|w_1, w_2, w_3) \\ & ... \\ & \times p(w_n|w1, w2, ..., w_{n-1}) \end{aligned}$$

mostly the same problem..

Markov Assumption

The future is independent of the past given the present.

Markov Assumption

The future is independent of the past given the present.

First order markov assumption:

$$p(w_n|w1, w2, w3, ..., w_{n-1}) \approx p(w_n|w_{n-1})$$

Markov Assumption

The future is independent of the past given the present.

First order markov assumption:

$$p(w_n|w1, w2, w3, ..., w_{n-1}) \approx p(w_n|w_{n-1})$$

Second order markov assumption:

$$p(w_n|w1, w2, w3, ..., w_{n-1}) \approx p(w_n|w_{n-2}, w_{n-1})$$

Markov Assumption

The future is independent of the past given the present.

First order markov assumption:

$$p(w_n|w1, w2, w3, ..., w_{n-1}) \approx p(w_n|w_{n-1})$$

Second order markov assumption:

$$p(w_n|w1, w2, w3, ..., w_{n-1}) \approx p(w_n|w_{n-2}, w_{n-1})$$

zero-order:

$$p(w_n|w1, w2, w3, ..., w_{n-1}) \approx p(w_n)$$

Markov Assumption

Unigram Model (zero order):

$$\begin{aligned} p(w_1, w_2, ..., w_n) = & p(w_1) \\ & \times p(w_2) \\ & \times p(w_3) \\ & \times p(w_4) \\ & ... \\ & \times p(w_{n-1}) \\ & \times p(w_n) \end{aligned}$$

Markov Assumption

Unigram Model (zero order):

$$\begin{aligned} p(w_1, w_2, ..., w_n) = & p(w_1) \\ & \times p(w_2) \\ & \times p(w_3) \\ & \times p(w_4) \\ & ... \\ & \times p(w_{n-1}) \\ & \times p(w_n) \end{aligned}$$

"bag of words"

Markov Assumption

Bigram Model (first order):

$$\begin{aligned} p(w_1, w_2, ..., w_n) = & p(w_1) \\ & \times p(w_2|w_1) \\ & \times p(w_3|w_2) \\ & \times p(w_4|w_3) \\ & ... \\ & \times p(w_{n-1}|w_{n-2}) \\ & \times p(w_n|w_{n-1}) \end{aligned}$$

Markov Assumption

Trigram Model (second order):

$$\begin{aligned} p(w_1, w_2, ..., w_n) = & p(w_1) \\ & \times p(w_2 | w_1) \\ & \times p(w_3 | w_1, w_2) \\ & \times p(w_4 | w_2, w_3) \\ & ... \\ & \times p(w_{n-1} | w_{n-3}, w_{n-2}) \\ & \times p(w_n | w_{n-2}, w_{n-1}) \end{aligned}$$

Is the markov assumption correct?

Is the markov assumption correct?

he is from Italy, so if I had to guess
I'd say his first language is ...

Is the markov assumption correct?

he is from Italy, so if I had to guess
I'd say his first language is ...

the boy with the blue shirt and the brown eyes ... [is/are]

Is the markov assumption correct?

he is from Italy, so if I had to guess
I'd say his first language is ...

the boy with the blue shirt and the brown eyes ... [is/are]

Long distance dependencies. Recursive structure.

Why CAN we often get away
with markovian (n-gram) models?

Trigram Language Models

- A trigram language model consists of:
  1. A finite set $\mathcal{V}$
  2. A parameter $q(w|u,v)$ for each trigram u, v, w such that $w \in \mathcal{V} \cup \{\text{STOP}\}$, and $u, v \in \mathcal{V} \cup \{*\}$.

Trigram Language Models

- A trigram language model consists of:
  1. A finite set $\mathcal{V}$
  2. A parameter $q(w|u,v)$ for each trigram u,v,w such that $w \in \mathcal{V} \cup \{\text{STOP}\}$, and $u,v \in \mathcal{V} \cup \{*\}$.

- For any sentence $x_1 \ldots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \ldots (n-1)$, and $x_n = \text{STOP}$, the probability of the sentence under the trigram language model is

$$p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-2}, x_{i-1})$$

  where we define $x_0 = x_{-1} = *$.

An Example

For the sentence

the dog barks STOP

we would have

$$\begin{aligned} p(\text{the dog barks STOP}) &= q(\text{the}|\text{*, *}) \\ &\quad \times q(\text{dog}|\text{*, the}) \\ &\quad \times q(\text{barks}|\text{the, dog}) \\ &\quad \times q(\text{STOP}|\text{dog, barks}) \end{aligned}$$

N-gram language models define proper distributions

If **every local estimate $p(w_i \mid w_{i-k}, ,..., w_{i-1})$**
is a probability distribution (positive, sums to 1),
then
the **sequence probabilities** under an N-gram (K-gram...) language model with a STOP symbol
are **also a probability distribution**.

(can you prove this?)

The Trigram Estimation Problem

Remaining estimation problem:

$$q(w_i \mid w_{i-2}, w_{i-1})$$

For example:

$$q(\text{laughs} \mid \text{the, dog})$$

The Trigram Estimation Problem

Remaining estimation problem:

$$q(w_i \mid w_{i-2}, w_{i-1})$$

For example:

$$q(\text{laughs} \mid \text{the, dog})$$

A natural estimate (the "maximum likelihood estimate"):

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

$$q(\text{laughs} \mid \text{the, dog}) = \frac{\text{Count}(\text{the, dog, laughs})}{\text{Count}(\text{the, dog})}$$

Language Model as a Generative Process

- We can use the LM to:
  - Assign scores to existing sentences / sequences.
  - Generate sentences (how)?

[some LM generation examples - Small Hebrew Twitter]

Measuring Model Quality

- The goal isn't to pound out fake sentences!
  - Obviously, generated sentences get "better" as we increase the model order
  - More precisely: using ML estimators, higher order always gives better likelihood on train, but not test
- What we really want to know is:
  - Will our model prefer good sentences to bad ones?
  - Bad $\neq$ ungrammatical!
  - Bad $\approx$ unlikely
  - Bad = sentences that our acoustic model really likes but aren't the correct answer

Measuring Model Quality

- The Shannon Game:
  - How well can we predict the next word?

    When I eat pizza, I wipe off the ____

    Many children are allergic to ____

    I saw a ____

$$\left\{\begin{array}{ll} \text{grease} & 0.5 \\ \text{sauce} & 0.4 \\ \text{dust} & 0.05 \\ \ldots & \\ \text{mice} & 0.0001 \\ \ldots & \\ \text{the} & \text{1e-100} \end{array}\right.$$

Claude Shannon

  - Unigrams are terrible at this game. (Why?)

- A better model of a text…
  - is one which assigns a higher probability to the word that actually occurs

Evaluating a Language Model: Perplexity

- We have some test data, m sentences

$$s_1, s_2, s_3, \ldots, s_m$$

Evaluating a Language Model: Perplexity

- We have some test data, m sentences

$$s_1, s_2, s_3, \ldots, s_m$$

- We could look at the probability under our model $\prod_{i=1}^{m} p(s_i)$. Or more conveniently, the *log probability*

$$\log \prod_{i=1}^{m} p(s_i) = \sum_{i=1}^{m} \log p(s_i)$$

Evaluating a Language Model: Perplexity

- We have some test data, m sentences

$$s_1, s_2, s_3, \ldots, s_m$$

- We could look at the probability under our model $\prod_{i=1}^{m} p(s_i)$. Or more conveniently, the *log probability*

$$\log \prod_{i=1}^{m} p(s_i) = \sum_{i=1}^{m} \log p(s_i)$$

- In fact the usual evaluation measure is *perplexity*

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

and M is the total number of words in the test data.

Some Intuition about Perplexity

- Say we have a vocabulary $\mathcal{V}$, and $N = |\mathcal{V}| + 1$ and model that predicts

$$q(w|u,v) = \frac{1}{N}$$

  for all $w \in \mathcal{V} \cup \{\text{STOP}\}$, for all $u, v \in \mathcal{V} \cup \{*\}$.
- Easy to calculate the perplexity in this case:

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \log \frac{1}{N}$$

$\Rightarrow$

$$\text{Perplexity} = N$$

Perplexity is a measure of effective “branching factor”

Typical Values of Perplexity

- Results from Goodman ("A bit of progress in language modeling"), where $|\mathcal{V}| = 50,000$
- A trigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-2}, x_{i-1})$. Perplexity $= 74$

Typical Values of Perplexity

- Results from Goodman ("A bit of progress in language modeling"), where $|\mathcal{V}| = 50,000$
- A trigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i | x_{i-2}, x_{i-1})$. Perplexity $= 74$
- A bigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i | x_{i-1})$. Perplexity $= 137$

Typical Values of Perplexity

- Results from Goodman ("A bit of progress in language modeling"), where $|\mathcal{V}| = 50,000$
- A trigram model: $p(x_1 \dots x_n) = \prod_{i=1}^{n} q(x_i | x_{i-2}, x_{i-1})$.
  Perplexity $= 74$
- A bigram model: $p(x_1 \dots x_n) = \prod_{i=1}^{n} q(x_i | x_{i-1})$.
  Perplexity $= 137$
- A unigram model: $p(x_1 \dots x_n) = \prod_{i=1}^{n} q(x_i)$.
  Perplexity $= 955$

Some History

- Shannon conducted experiments on entropy of English i.e., how good are people at the perplexity game?

  C. Shannon. Prediction and entropy of printed English. Bell Systems Technical Journal, 30:50–64, 1951.

Some History (perplexity != grammaticality)

Chomsky (in *Syntactic Structures* (1957)):

Second, the notion "grammatical" cannot be identified with "meaningful" or "significant" in any semantic sense. Sentences (1) and (2) are equally nonsensical, but any speaker of English will recognize that only the former is grammatical.

(1) Colorless green ideas sleep furiously.

(2) Furiously sleep ideas green colorless.

. . .

. . . Third, the notion "grammatical in English" cannot be identified in any way with the notion "high order of statistical approximation to English". It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical model for grammaticalness, these sentences will be ruled out on identical grounds as equally 'remote' from English. Yet (1), though nonsensical, is grammatical, while (2) is not. . . .

Perplexity

$$PP = 2^{-l} \qquad l = \frac{1}{M}\sum_{i=1}^{m} \log_2 p(X^{(i)})$$

- Lower is better!
- Perplexity is the inverse probability of the test set normalized by the number of words
- If we ever give a test n-gram zero probability → perplexity will be infinity
- How can we avoid this?

Sparse Data Problems

Reminder:

A natural estimate (the "maximum likelihood estimate"):

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

$$q(\text{laughs} \mid \text{the, dog}) = \frac{\text{Count(the, dog, laughs)}}{\text{Count(the, dog)}}$$

Say our vocabulary size is $N = |\mathcal{V}|$, then there are N^3 parameters in the model.

e.g., $N = 20,000 \quad \Rightarrow \quad 20,000^3 = 8 \times 10^{12}$ parameters

Zeroes

- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request
- Test set:
  … denied the offer
  … denied the loan

P("offer" | denied the) = 0

- Bigrams with zero probability
  – Mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can't divide by 0)!

Smoothing

How to assign non-zero probability to zero-occurrence events?

Smoothing

How to assign non-zero probability to zero-occurrence events?

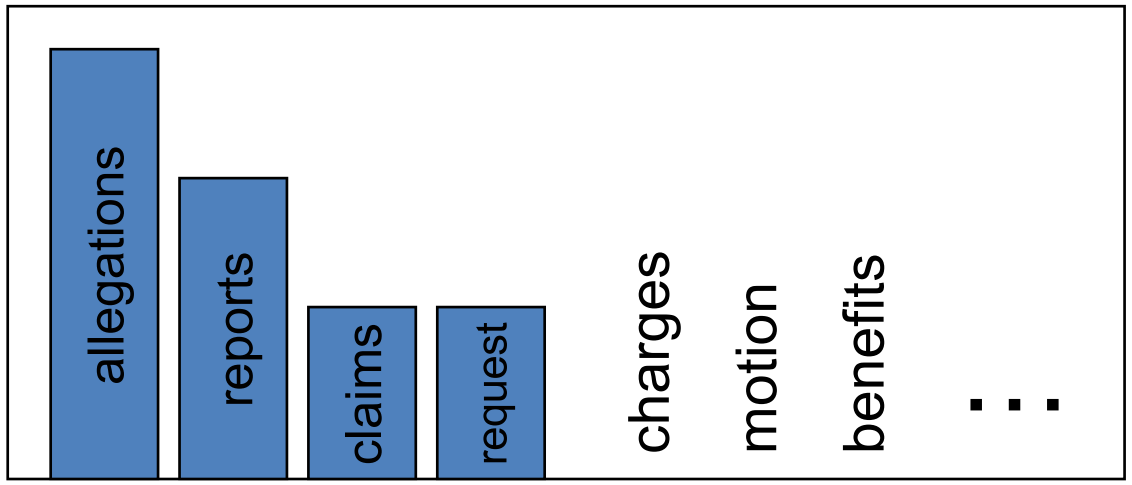
Smoothing used to be a HUGE issue.
Mitigated to a large extent by neural techniques.
Still worth knowing the basics, perhaps without the gory details.

Smoothing

- We often want to make estimates from sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request

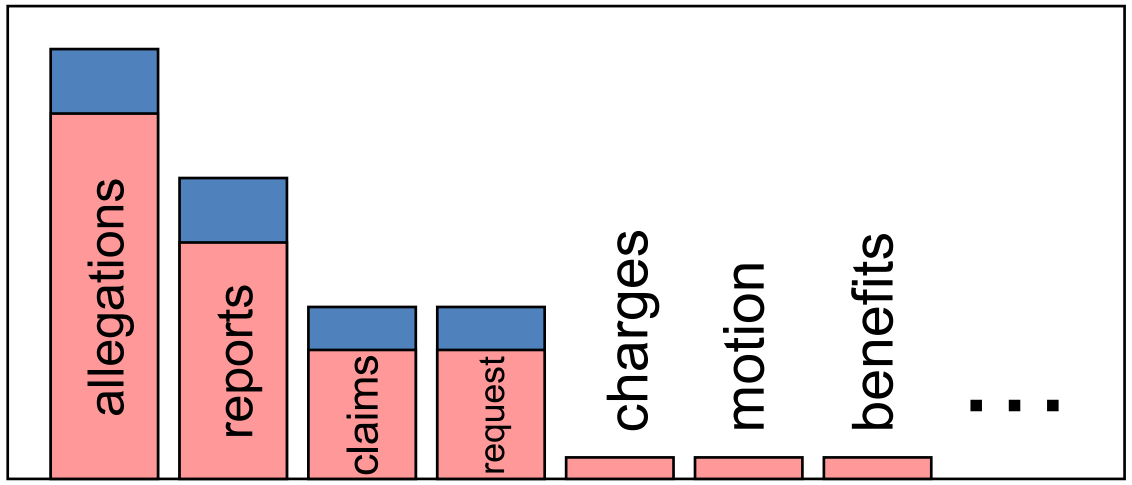
7 total



- Smoothing flattens spiky distributions so they generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other

7 total



- Very important all over NLP (and ML more generally), but easy to do badly!

Add-one Estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!

- MLE estimate:

$$P_{\text{MLE}}(x_i \mid x_{i-1}) = \frac{c(x_{i-1}, x_i)}{c(x_{i-1})}$$

- Add-1 estimate:

$$P_{\text{Add-1}}(x_i \mid x_{i-1}) = \frac{c(x_{i-1}, x_i) + 1}{c(x_{i-1}) + V}$$

Add-one Estimation

this is a very crude method, that over-assigns probability to unseen events.

- MLE estimate:

$$P_{\text{MLE}}(x_i \mid x_{i-1}) = \frac{c(x_{i-1}, x_i)}{c(x_{i-1})}$$

- Add-1 estimate:

$$P_{\text{Add-1}}(x_i \mid x_{i-1}) = \frac{c(x_{i-1}, x_i) + 1}{c(x_{i-1}) + V}$$

More General Formulation

- Add-K:

$$P_{\text{Add-k}}(x_i \mid x_{i-1}) = \frac{c(x_{i-1}, x_i) + k}{c(x_{i-1}) + kV}$$

$$P_{\text{Add-k}}(x_i \mid x_{i-1}) = \frac{c(x_{i-1}, x_i) + m\frac{1}{V}}{c(x_{i-1}) + m}$$

- Unigram Prior Smoothing:

$$P_{\text{Add-k}}(x_i \mid x_{i-1}) = \frac{c(x_{i-1}, x_i) + mP(x_i)}{c(x_{i-1}) + m}$$

Linear Interpolation (another smoothing method)

- Take our estimate $q(w_i \mid w_{i-2}, w_{i-1})$ to be

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = \quad & \lambda_1 \times q_{\mathsf{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\ & +\lambda_2 \times q_{\mathsf{ML}}(w_i \mid w_{i-1}) \\ & +\lambda_3 \times q_{\mathsf{ML}}(w_i) \end{aligned}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i.

Katz Back-Off Models (Bigrams)

- For a bigram model, define two sets

$$\begin{aligned} \mathcal{A}(w_{i-1}) &= \{w : \text{Count}(w_{i-1}, w) > 0\} \\ \mathcal{B}(w_{i-1}) &= \{w : \text{Count}(w_{i-1}, w) = 0\} \end{aligned}$$

- A bigram model

$$q_{BO}(w_i \mid w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-1}, w_i)}{\text{Count}(w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-1}) \\ \alpha(w_{i-1}) \frac{q_{\text{ML}}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} q_{\text{ML}}(w)} & \text{If } w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-1})} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

Katz Back-Off Models (Trigrams)

- For a trigram model, first define two sets

$$\begin{aligned}\mathcal{A}(w_{i-2}, w_{i-1}) &= \{w : \text{Count}(w_{i-2}, w_{i-1}, w) > 0\} \\ \mathcal{B}(w_{i-2}, w_{i-1}) &= \{w : \text{Count}(w_{i-2}, w_{i-1}, w) = 0\}\end{aligned}$$

- A trigram model is defined in terms of the bigram model:

$$q_{BO}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})} \\ \qquad \text{If } w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\ \\ \frac{\alpha(w_{i-2}, w_{i-1}) q_{BO}(w_i | w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} q_{BO}(w | w_{i-1})} \\ \qquad \text{If } w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-2}, w_{i-1})} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w)}{\text{Count}(w_{i-2}, w_{i-1})}$$

Advanced Smoothing Algorithms

- Intuition: Use the count of things we've **seen once**
  - To help estimate the count of things we've **never seen**
- Used by many smoothing algorithms
  - Good-Turing
  - Kneser-Ney
  - Also: Witten-Bell

Invented during WWII by Alan Turing and later published by Good. Frequency estimates were needed for Enigma code-breaking effort

Advanced Smoothing Algorithms

- Intuition: Use the count of things we've **seen once**
  - To help estimate the count of things we've **never**

Smoothing used to be a HUGE issue.
Mitigated to a large extent by neural techniques.
Still worth knowing the basics,
perhaps without the gory details.

  - Kneser-Ney
  - Also: Witten-Bell

Invented during WWII by Alan Turing and later published by Good. Frequency estimates were needed for Enigma code-breaking effort

Kneser-Ney Smoothing
(main intuition)

- Better estimate for probabilities of lower-order unigrams!
  - Shannon game: I can't see without my reading___________? glasses? / Francisco?
  - "Francisco" is more common than "glasses"
  - … but "Francisco" always follows "San"
- Instead of P(w): "How likely is w"
- $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?

What Actually Works?

- Trigrams and beyond:
  - Unigrams, bigrams generally useless
  - Trigrams much better (when there's enough data)
  - 4-, 5-grams really useful in MT, but not so much for speech
- Discounting
  - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell, etc…

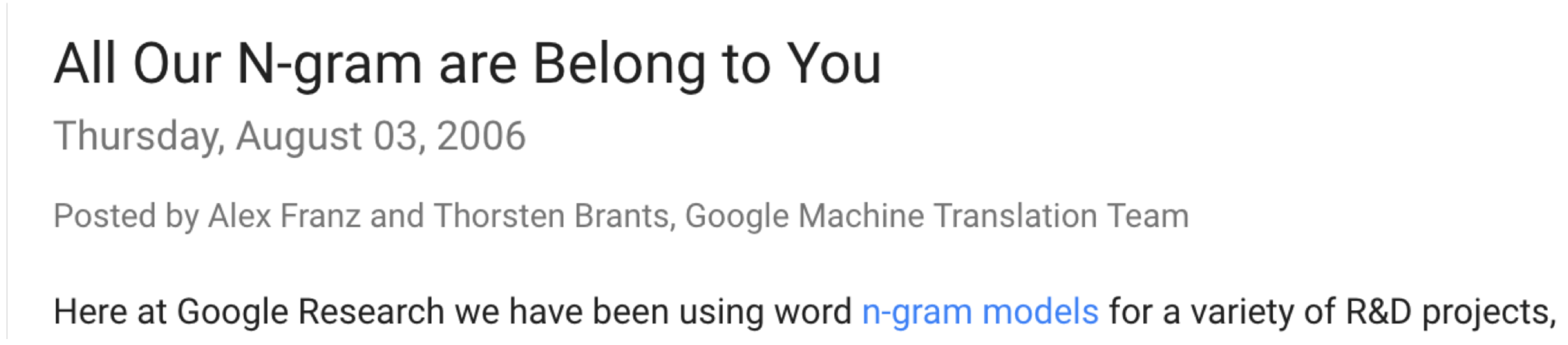
  Kneser-Ney is very competitive.
- See [Chen+Goodman] reading for tons of graphs…

- Three steps in deriving the language model probabilities:
  1. Expand $p(w_1, w_2 \ldots w_n)$ using Chain rule.
  2. Make Markov Independence Assumptions
     $p(w_i \mid w_1, w_2 \ldots w_{i-2}, w_{i-1}) = p(w_i \mid w_{i-2}, w_{i-1})$
  3. Smooth the estimates using low order counts.

- Other methods used to improve language models:
  - "Topic" or "long-range" features.
  - Syntactic models.

Smoothing "alternative":

Web-scale N-grams



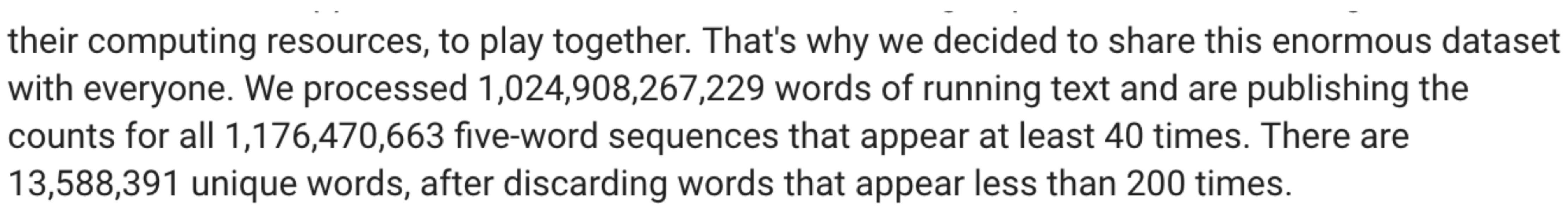
All Our N-gram are Belong to You

Thursday, August 03, 2006

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

...



their computing resources, to play together. That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

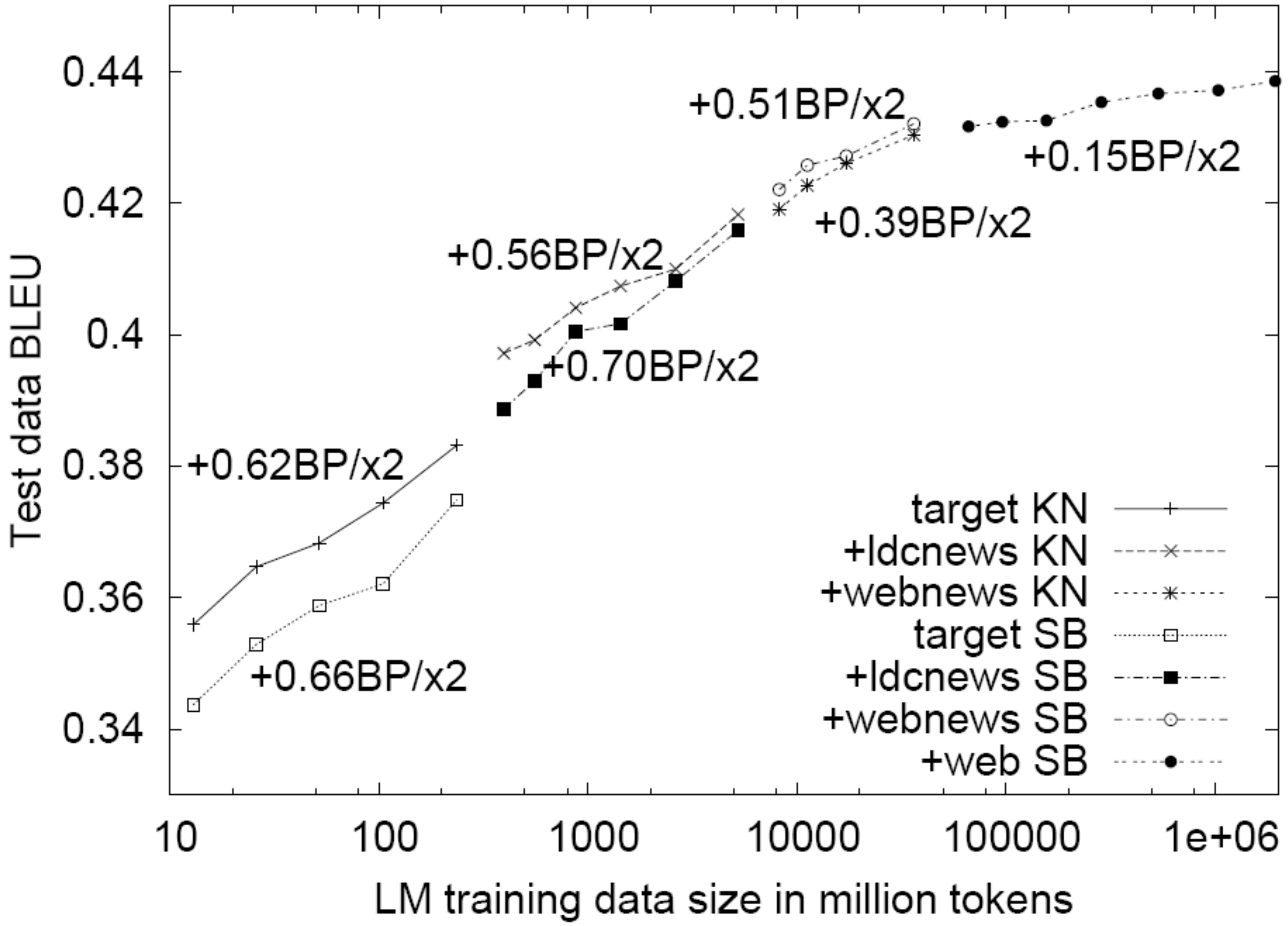
Google N-grams

- `serve as the incoming 92`
- `serve as the incubator 99`
- `serve as the independent 794`
- `serve as the index 223`
- `serve as the indication 72`
- `serve as the indicator 120`
- `serve as the indicators 45`
- `serve as the indispensable 111`
- `serve as the indispensible 40`
- `serve as the individual 234`

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Even More Data!

Tons of data closes gap, for extrinsic MT evaluation



http://www.aclweb.org/anthology/D07-1090.pdf

Practical Issues

- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)
  - (though log can be slower than multiplication)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Dealing with Unknown Words

- Realistically, many words will be unknown (OOV -- Out of vocabulary)
- What can we do?
  - Use a special <UNK> symbol.
    (how do we estimate its probability?)
  - Look for "similar" words (how?)

Case Study: Language Identification

- How can we tell what language a document is in?

The 38th Parliament will meet on Monday, October 4, 2004, at 11:00 a.m. The first item of business will be the election of the Speaker of the House of Commons. Her Excellency the Governor General will open the First Session of the 38th Parliament on October 5, 2004, with a Speech from the Throne.

La 38e législature se réunira à 11 heures le lundi 4 octobre 2004, et la première affaire à l'ordre du jour sera l’élection du président de la Chambre des communes. Son Excellence la Gouverneure générale ouvrira la première session de la 38e législature avec un discours du Trône le mardi 5 octobre 2004.

- How to tell the French from the English?

Case Study: Language Identification

- How can we tell what language a document is in?

The 38th Parliament will meet on Monday, October 4, 2004, at 11:00 a.m. The first item of business will be the election of the Speaker of the House of Commons. Her Excellency the Governor General will open the First Session of the 38th Parliament on October 5, 2004, with a Speech from the Throne.

La 38e législature se réunira à 11 heures le lundi 4 octobre 2004, et la première affaire à l'ordre du jour sera l’ élection du président de la Chambre des communes. Son Excellence la Gouverneure générale ouvrira la première session de la 38e législature avec un discours du Trône le mardi 5 octobre 2004.

- How to tell the French from the English?

Build two character-level language models:

- English language model.
- French Language Model.

Which assigns the text a higher probability?

To Summarize (this part)

- The Language Modeling Problem
- The markov assumption and N-gram language models.
- Maximum Likelihood Estimation (MLE).
- Smoothing.
- LM as classifier.