LANGUAGE MODELS

Based on slides from David Kauchak and Philipp Koehn.

March 9, 2015

Language modeling

What does natural language look like?

More specifically in NLP, probabilistic model

Two related questions:

- p(sentence)
  - p(“I like to eat pizza”)
  - p(“pizza like I eat”)
- p(word | previous words)
  - p(“pizza” | “I like to eat”)
  - p(“garbage” | “I like to eat”)
  - p(“run” | “I like to eat”)

Language modeling

How might these models be useful?

- Language generation tasks
  - machine translation
  - summarization
  - speech recognition
- Text correction
  - spelling correction
  - grammar correction
- Topic modeling
- Genre modeling

Language Model Requirements

- Assign a probability to *every sentence* (i.e., string of words)

$$\sum_{\mathbf{e} \in \Sigma^*} p_{\mathrm{LM}}(\mathbf{e}) = 1$$

$$p_{\mathrm{LM}}(\mathbf{e}) \geq 0 \quad \forall \mathbf{e} \in \Sigma^*$$

How do LMs help?

- Goal: Assign a higher probability to *good sentences in English*

p_{LM}(*the house is small*) > p_{LM}(*small the is house*)

translations of German *Haus: home, house* …

p_{LM}(*I am going home*) > p_{LM}(*I am going house*)

Ideas?

p("I like to eat pizza")

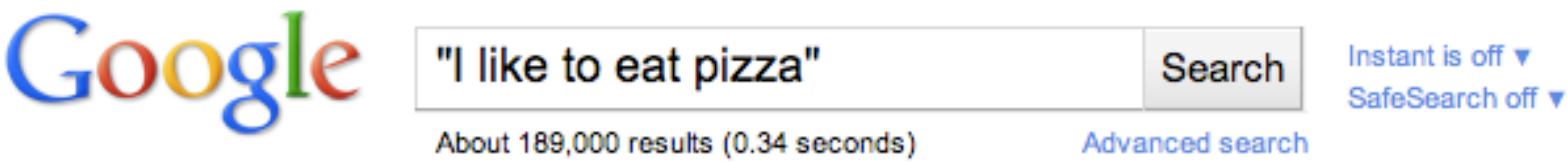
p("pizza like I eat")

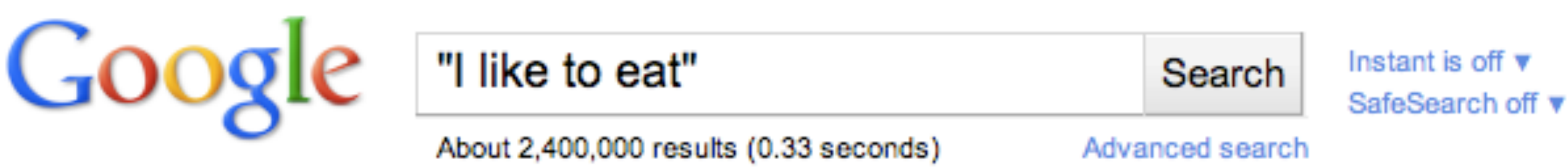
p("pizza" | "I like to eat")

p("garbage" | "I like to eat")

p("run" | "I like to eat")

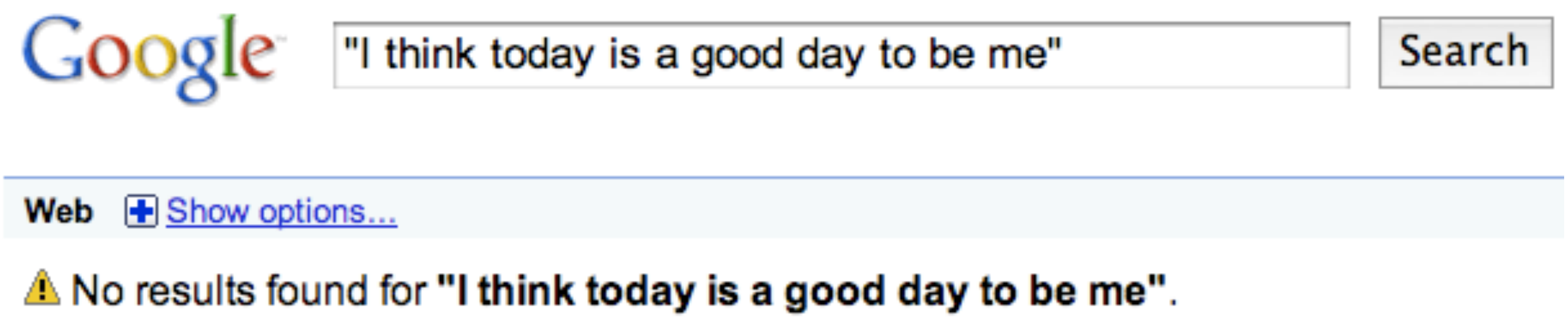
Look at a corpus





Language modeling

I think today is a good day to be me



Language modeling is about dealing with data sparsity!

Language modeling

A language model is really a probabilistic explanation of how the sentence was generated

Key idea:

- break this generation process into smaller steps
- estimate the probabilities of these smaller steps
- the overall probability is the combined product of the steps

Language modeling

Two approaches:

- n-gram language modeling
  - Start at the beginning of the sentence
  - Generate one word at a time based on the previous words
- syntax-based language modeling
  - Construct the syntactic tree from the top down
  - e.g. context free grammar
  - eventually at the leaves, generate the words

Pros/cons?

n-gram language modeling

I think today is a good day to be me

Google "I think" Search

Web Show options... Results **1** - **10** of about **564,000,000** for "**I think**". (**0.28** seconds)

Google "today is a good day" Search

Web Show options... Results **1** - **10** of about **10,100,000** for "**today is a good day**".

Google "to be me" Search

Web Show options... Results **1** - **10** of about **70,200,000** for "**to be me**".

Our friend the chain rule

Step 1: decompose the probability

P(I think today is a good day to be me) =

P(I | <start>) x

P(think | I) x

P(today| I think) x

P(is| I think today) x

P(a| I think today is) x

P(good| I think today is a) x

…

The n-gram approximation

Assume each word depends only on the previous n-1 words
(e.g. trigram: three words total)

(Also called a Markov assumption)

$$P(\text{is} \mid \text{I think today}) \approx P(\text{is} \mid \text{think today})$$

$$P(\text{a} \mid \text{I think today is}) \approx P(\text{a} \mid \text{today is})$$

$$P(\text{good} \mid \text{I think today is a}) \approx P(\text{good} \mid \text{is a})$$

Estimating probabilities

How do we find probabilities?

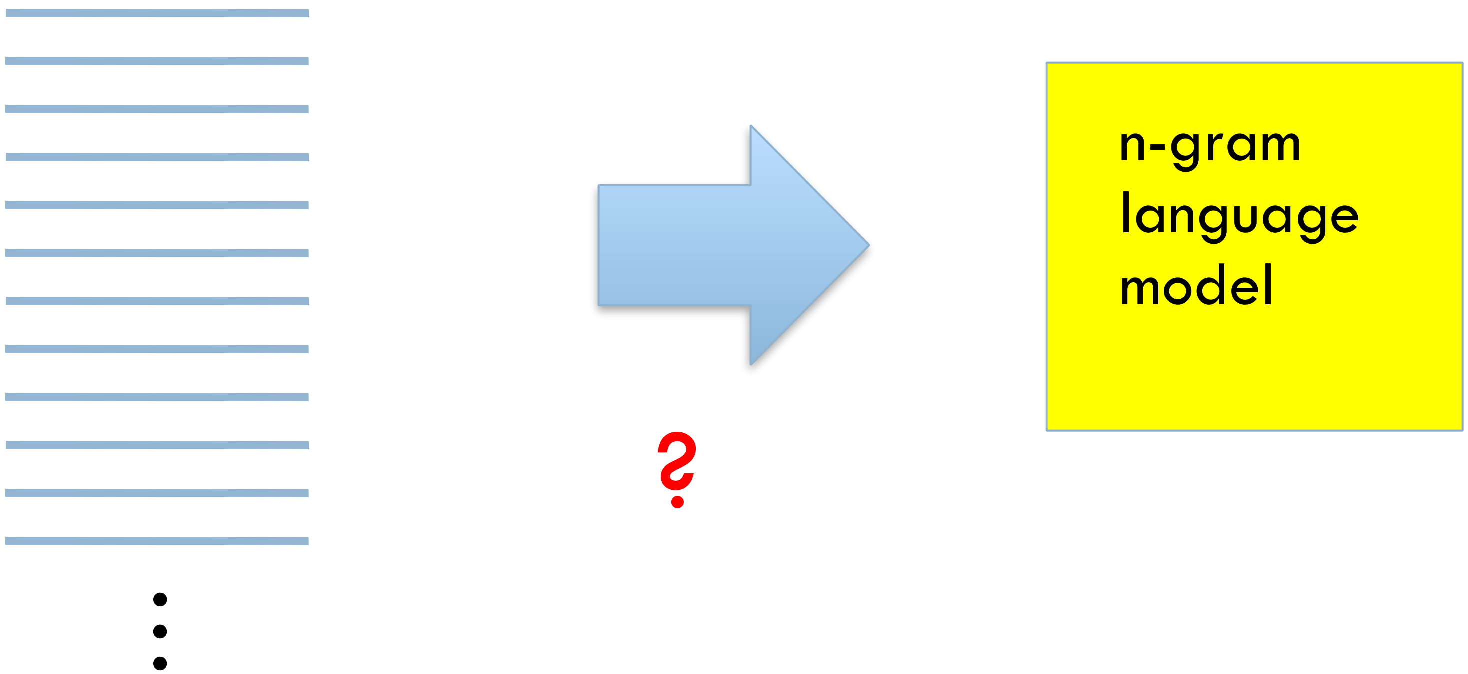
$P(\text{is} \mid \text{think today})$

Get real text, and start counting (MLE)!

$$P(\text{is} \mid \text{think today}) = \frac{\text{count}(\text{think today is})}{\text{count}(\text{think today})}$$

Estimating from a corpus

Corpus of sentences
(e.g. gigaword corpus)



Estimating from a corpus

I am ready for spring break now .

count all of the trigrams

<start> <start> I
<start> I am
I am ready
am ready for
ready for spring
for spring break
spring break now
break now .
now . <end>
. <end> <end>

why do we need
<start> and <end>?

Estimating from a corpus

I am ready for spring break now .

count all of the trigrams

<start> <start> I
<start> I am
I am ready
am ready for
ready for spring
for spring break
spring break now
break now .
now . <end>
. <end> <end>

Do we need to count anything else?

Estimating from a corpus

I am ready for spring break now .

count all of the bigrams

<start> <start> I
<start> I am
I am ready
am ready for
ready for spring
for spring break
spring break now
break now .
now . <end>
. <end> <end>

$$p(c \mid a\ b) = \frac{\text{count}(a\ b\ c)}{\text{count}(a\ b)}$$

Estimating from a corpus

1. Go through all sentences and count trigrams and bigrams
   - usually you store these in some kind of data structure

2. Now, go through all of the trigrams and use the count and the bigram count to calculate MLE probabilities
   - do we need to worry about divide by zero?

Applying a model

Given a new sentence, we can apply the model

p(Spring break will be here soon .) = ?

p(Spring | <start> <start>) *

p(break| <start> Spring) *

p(will | Spring break) *

⋮

p(<end>| . <end>) *

Generating examples

We can also use a trained model to generate a random sentence

Ideas?

<start> <start> ______

We have a distribution over all possible starting words

Draw one from this distribution

p(A | <start> <start>)

p(Apples | <start> <start>)

p(I | <start> <start>)

p(The | <start> <start>)

⋮

p(Zebras | <start> <start>)

Generating examples

<start> <start> Zebras ________

repeat!

p(are | <start> Zebras)

p(eat | <start> Zebras)

p(think | <start> Zebras)

p(and | <start> Zebras)

⋮

p(mostly | <start> Zebras)

Generation examples

Unigram

are were that ères mammal naturally built describes jazz territory heteromyids film tenor prime live founding must on was feet negro legal gate in on beside . provincial san ; stephenson simply spaces stretched performance double-entry grove replacing station across to burma . repairing ères capital about double reached omnibus el time believed what hotels parameter jurisprudence words syndrome to ères profanity is administrators ères offices hilarius institutionalized remains writer royalty dennis , ères tyson , and objective , instructions seem timekeeper has ères valley ères " magnitudes for love on ères from allakaket , , ana central enlightened . to , ères is belongs fame they the corrected , . on in pressure %NUMBER% her flavored ères derogatory is won metcard indirectly of crop duty learn northbound ères ères dancing similarity ères named ères berkeley . . off-scale overtime . each mansfield stripes dānu traffic ossetic and at alpha popularity town

Generation examples

Bigrams

the wikipedia county , mexico .

maurice ravel . it is require that is sparta , where functions . most widely admired .

halogens chamiali cast jason against test site .

Generation examples

Trigrams

is widespread in north africa in june %NUMBER% %NUMBER% units were built by with .

jewish video spiritual are considered ircd , this season was an extratropical cyclone .

the british railways ' s strong and a spot .

Evaluation

We can train a language model on some data

How can we tell how well we're doing?

- for example
  - bigrams vs. trigrams
  - 100K sentence corpus vs. 100M
  - …

Evaluation

A very good option: extrinsic evaluation

If you're going to be using it for machine translation

- build a system with each language model
- compare the two based on their approach for machine translation

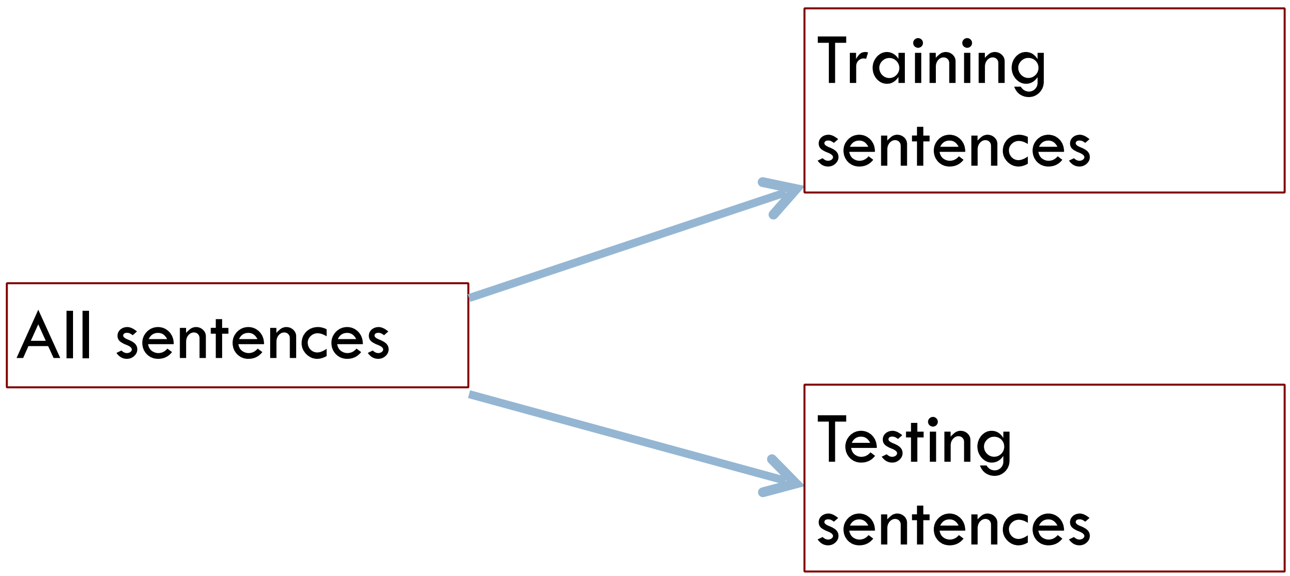
Sometimes we don't know the application

Can be time consuming

Granularity of results

Intrinsic Evaluation

Common NLP/machine learning/AI approach



Evaluation

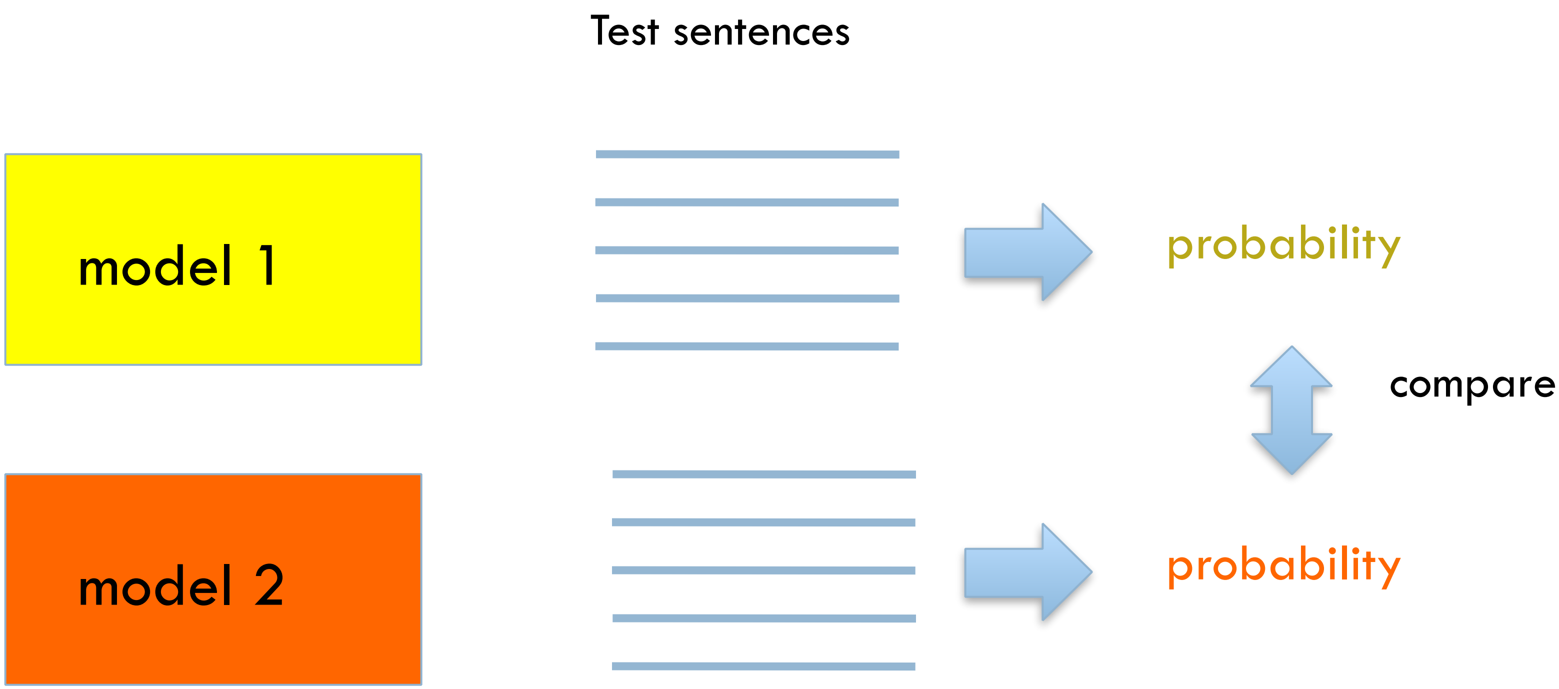
Test sentences

n-gram
language
model

Ideas?

Evaluation

A good model should do a good job of predicting actual sentences



Perplexity

View the problem as trying to predict the test corpus one word at a time in sequence

A perfect model would always know the next word with probability 1 (like people who finish each other's sentences)

Test sentences

I like to eat banana peels .

Aside: Some Information Theory

Entropy H(X): The average uncertainty of a random variable.

Mathematically, the average number of bits needed to encode the result of the random variable.

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

Aside: Some Information Theory

Perplexity PPL

$$\mathrm{PPL} = 2^{H(X,\hat{p})}$$

$$\mathrm{PPL} = 2^{-\sum_{x \in X} p(x) \log_2 \hat{p}(x)}$$

Intuitively: X is as random as if it had PPL equally-likely outcomes.

Entropy & Perplexity Practice

Suppose we have a 10-sided die, and we don't know if it's fair or not.

Theory 1 says that it probably is fair — $p(x) = 0.1$ for all x

Theory 2 is more suspicious — it argues that $p(x) = .05$ if x is odd, and $.15$ if x is even.

We roll the die and get the following:

1 2 2 4 8 2 9 3 6 7

For each theory, what is the cross-entropy? The perplexity?

Perplexity

We will use **perplexity** to evaluate models

Given: $\mathbf{w}, p_{\text{LM}}$

$$\text{PPL} = 2^{\frac{1}{|\mathbf{w}|} \log_2 p_{\text{LM}}(\mathbf{w})}$$

$$0 \leq \text{PPL} \leq \infty$$