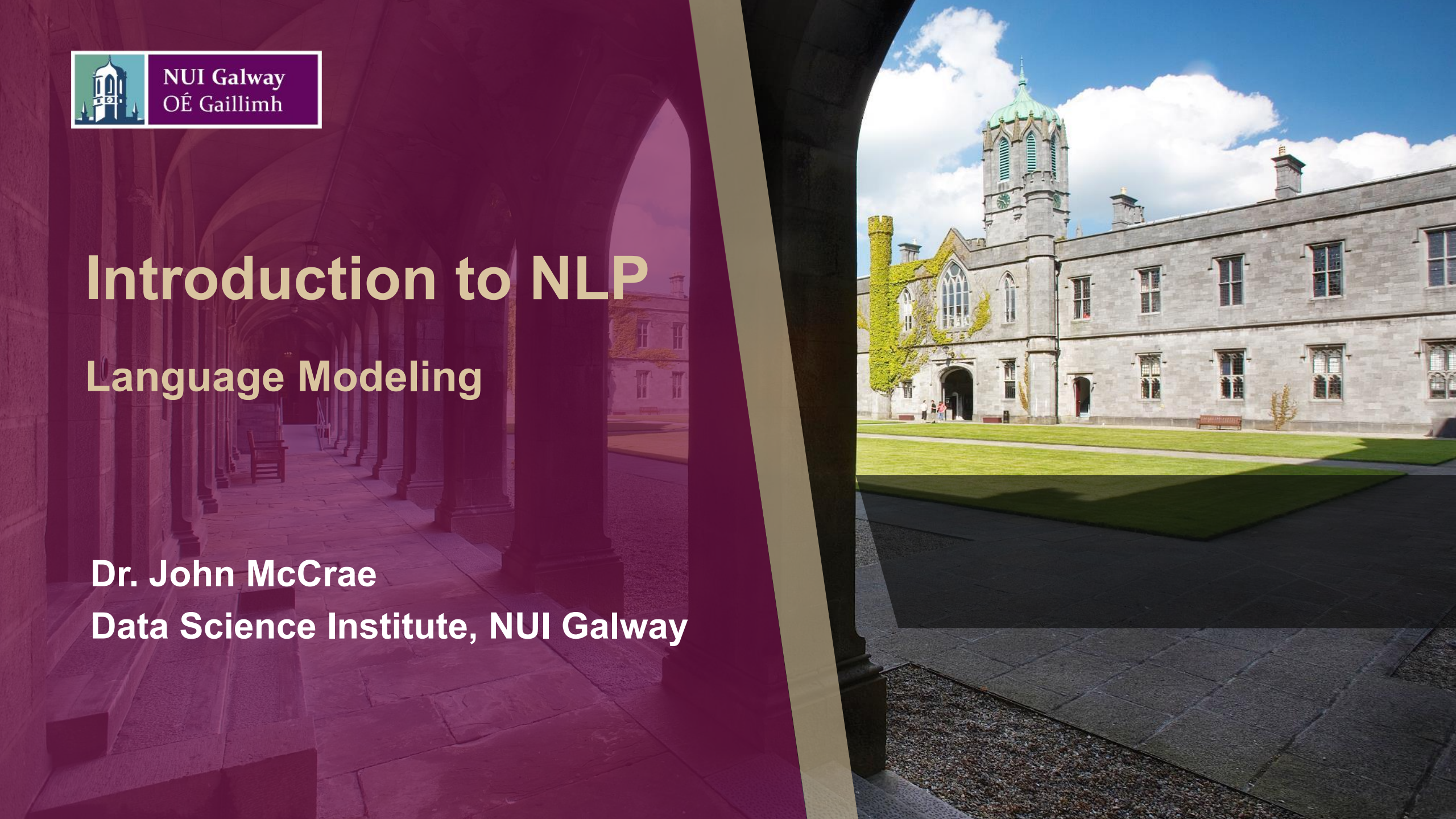


NUI Galway
OÉ Gaillimh

Introduction to NLP

Language Modeling

Dr. John McCrae
Data Science Institute, NUI Galway

Overview

Refresher on Probability

Estimating probabilities

n-grams

Language Model Smoothing

Evaluation of Language Models

Summary

Overview

Refresher on Probability

Estimating probabilities

n-grams

Language Model Smoothing

Evaluation of Language Models

Summary

Probability

p(heads) = 0.5

p(tails) = 0.5

random, repeatable event

Probability

p(rain tomorrow) = 0.3

p(rain yesterday) = 1.0 [or 0.0]

uncertainty of event

Probability in NLP

Probability of words is used in many NLP systems.

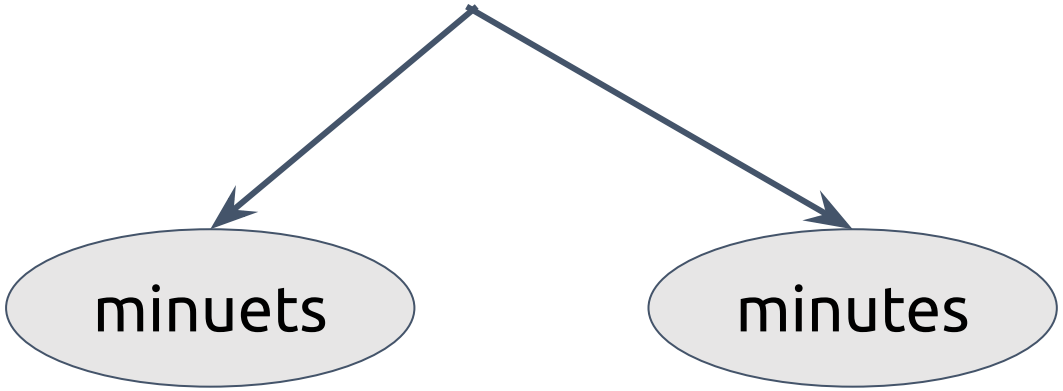
We talk about

p(“word”)

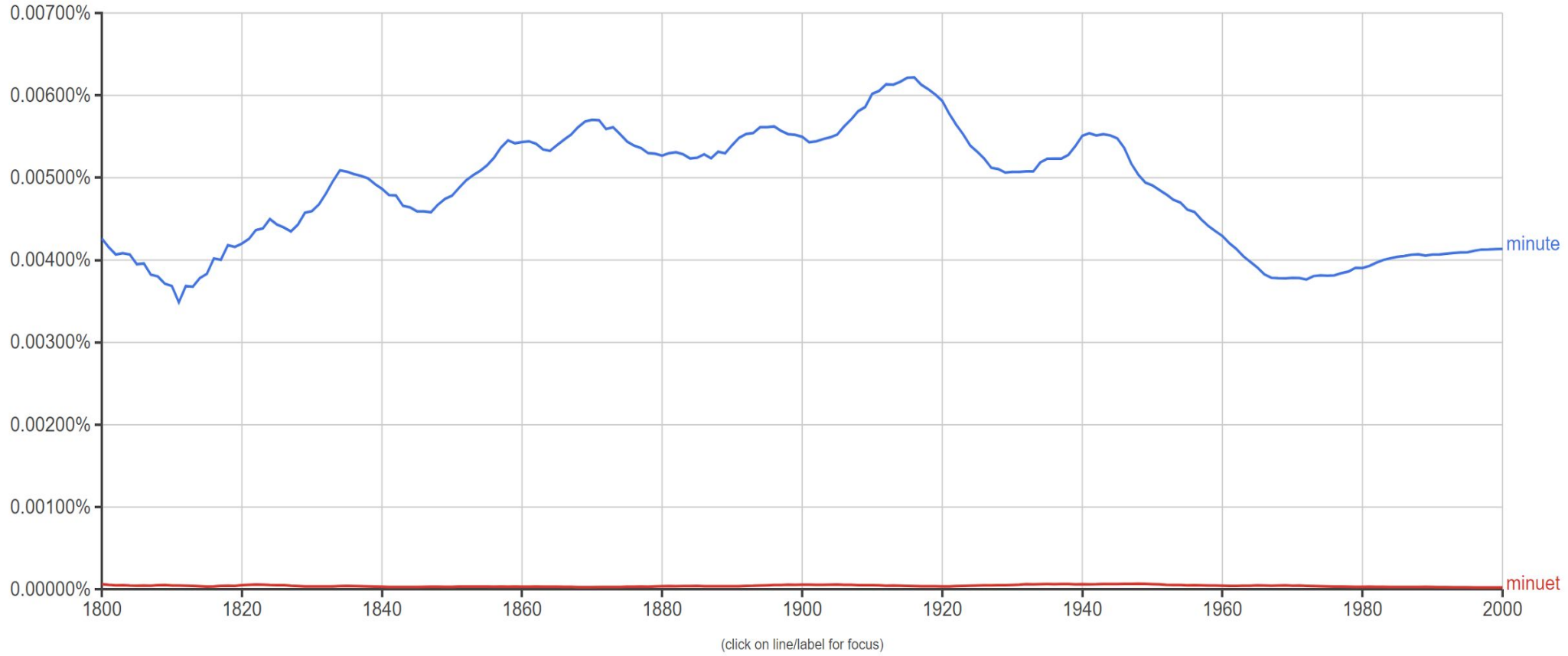
What does this mean?

Spelling correction

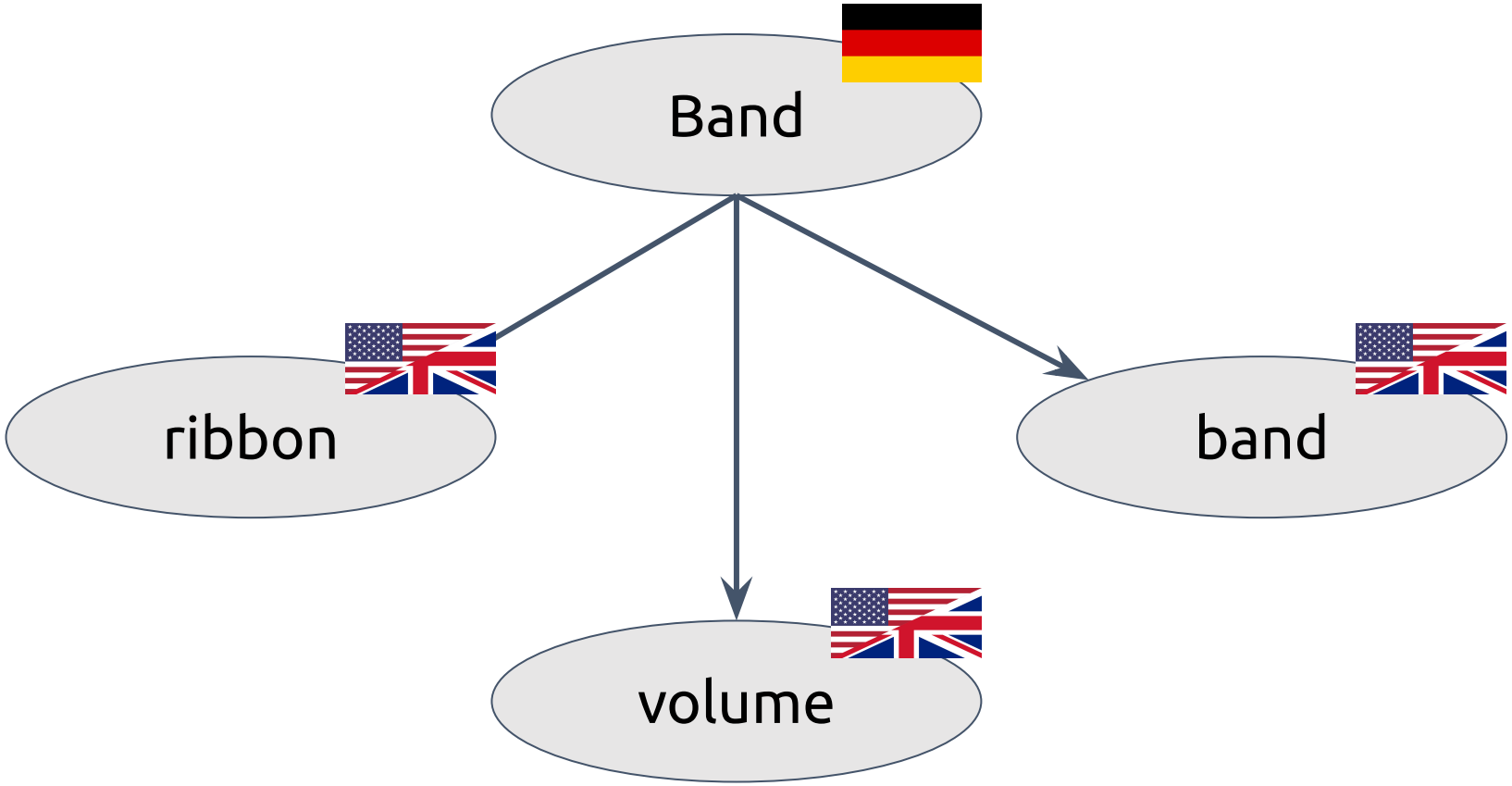
"I will be there in five minuts"



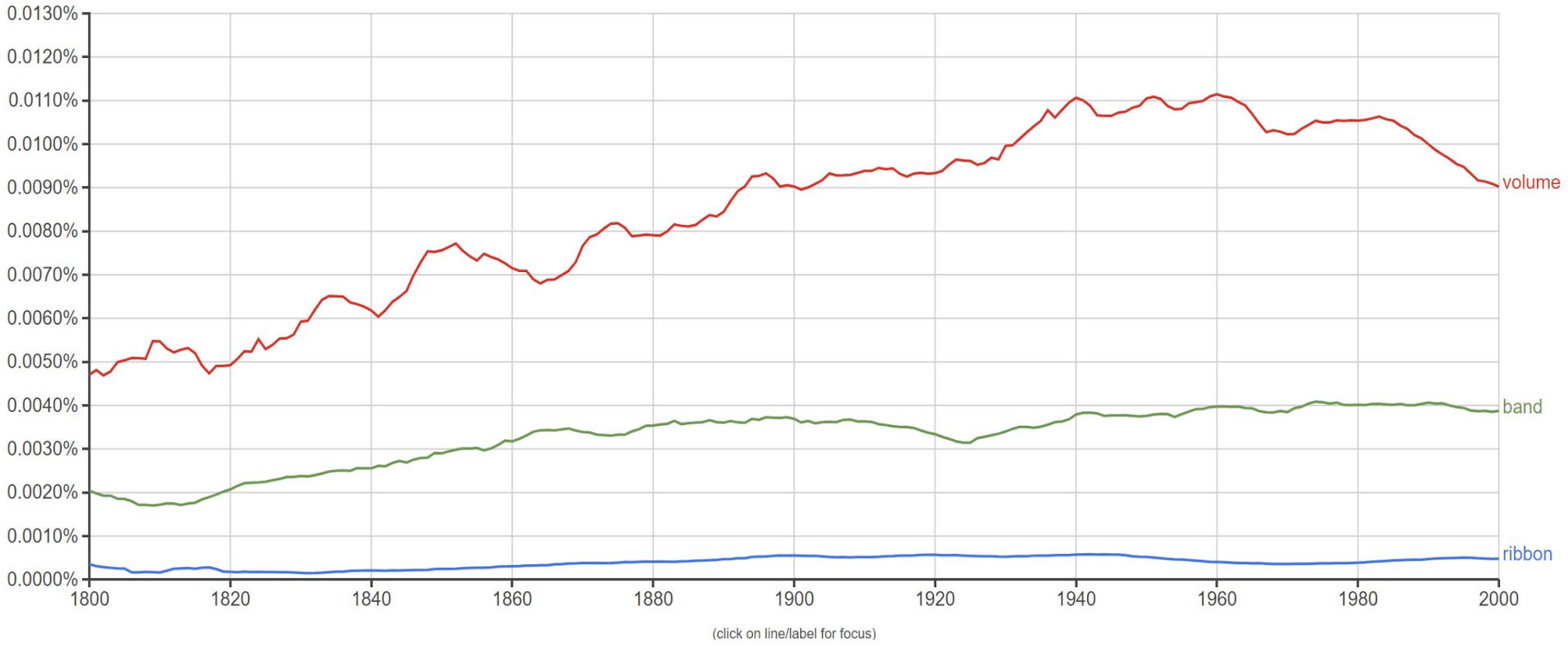
Spelling correction



Translation probability



Translation Probability



Conditional Probability

What we really need is:

$$p(\text{“volume”@en} \mid \text{“Band”@de})$$

Definition of conditional probability

$$p(A|B)p(B) = p(A \cap B)$$

Bayes Law

If $$p(A|B)p(B) = p(A \cap B)$$

Then also $$p(B|A)p(A) = p(B \cap A)$$

Thus $$\frac{p(A|B)p(B)}{p(B|A)p(A)} = \frac{p(A \cap B)}{p(B \cap A)} = 1$$

$$p(A|B) = p(B|A)\frac{p(A)}{p(B)}$$

Noisy Channel Model

We can treat p(Band) as a constant

$$p(\text{volume}|\text{Band}) = p(\text{Band}|\text{volume})\frac{p(\text{volume})}{p(\text{Band})}$$

$$p(\text{ribbon}|\text{Band}) = p(\text{Band}|\text{ribbon})\frac{p(\text{ribbon})}{p(\text{Band})}$$

$$p(\text{volume}|\text{Band}) \propto p(\text{Band}|\text{volume})p(\text{volume})$$

Overview

Refresher on Probability

Estimating probabilities

n-grams

Language Model Smoothing

Evaluation of Language Models

Summary

Estimating probabilities

For a vocabulary *V*, the most likely estimation of the probability of a word, *w*, given a count function, *c*, is:

$$p(w) = \frac{c(w)}{\sum_{v \in W} c(v)}$$

Example corpus

Do you like green eggs and ham?
I do not like them, Sam-I-am.
I do not like green eggs and ham.
Would you like them here or there?
I would not like them here or there.
I would not like them anywhere.
I do not like green eggs and ham.
I do not like them Sam-I-am.

Frequency counts

and	3	*like*	8
anywhere	1	*not*	6
do	5	*or*	2
eggs	3	*Sam-I-am*	2
green	3	*them*	5
ham	3	*there*	2
here	2	*would*	3
I	6	*you*	2

sum=56

Probabilities

and	0.05	*like*	0.14
anywhere	0.02	*not*	0.11
do	0.09	*or*	0.04
eggs	0.05	*Sam-I-am*	0.04
green	0.05	*them*	0.09
ham	0.05	*there*	0.04
here	0.04	*would*	0.05
I	0.11	*you*	0.04

Probability of a sentence (unigram)

$p(\text{“I like green eggs and ham”})$

$\cong p(\text{“I”}) \times p(\text{“like”}) \times p(\text{“green”}) \times p(\text{“eggs”}) \times p(\text{“and”}) \times p(\text{“ham”})$

$= 0.11 \times 0.14 \times 0.05 \times 0.05 \times 0.05 \times 0.05$

$= 0.00000013$

Out-of-vocabulary Problem

$p(\text{“Would you like them in a house”})$

$\cong p(\text{“would”}) \times p(\text{“you”}) \times p(\text{“like”}) \times p(\text{“them”}) \times p(\text{“in”}) \times p(\text{“a”}) \times p(\text{“house”})$

$= 0.05 \times 0.04 \times 0.13 \times 0.09 \times ??? \times ??? \times ???$

$= 0.05 \times 0.04 \times 0.13 \times 0.09 \times 0 \times 0 \times 0$

$= 0$

Solutions to data sparsity

Get a bigger corpus!

- Not always possible
- More complex probabilities require exponentially larger corpora

Add-one smoothing

Simple idea: add one to every count

$$p(w) = \frac{c(w) + 1}{\sum_{v \in V} c(v) + |V|}$$

Probabilities

and	0.06	*like*	0.13
anywhere	0.03	*not*	0.10
do	0.08	*or*	0.04
eggs	0.06	*Sam-I-am*	0.04
green	0.06	*them*	0.08
ham	0.06	*there*	0.04
here	0.04	*would*	0.06
I	0.10	*you*	0.04
OOV	0.01		

OOV with Add-One Smoothing

p(“Would you like them in a house”)

≅ p(“would”) × p(“you”) × p(“like”) × p(“them”) × p(“in”) × p(“a”) × p(“house”)

= 0.06 × 0.06 × 0.13 × 0.08 × 0.01 × 0.01 × 0.01

= 0.000000000065

Problems with add one smoothing

Not a probability distribution with OOV as does not sum to one

Overestimates low probabilities and underestimates frequent probabilities

Laplace smoothing (or add alpha) has two parameters:

- α, the amount to add
- N, the expected number of OOV

$$p(w) = \frac{c(w) + \alpha}{\sum_{v \in V} c(v) + \alpha(|V| + N)}$$

It is not clear how to find α or N

Probability Mass

Some of the probability of should be assigned to unseen events

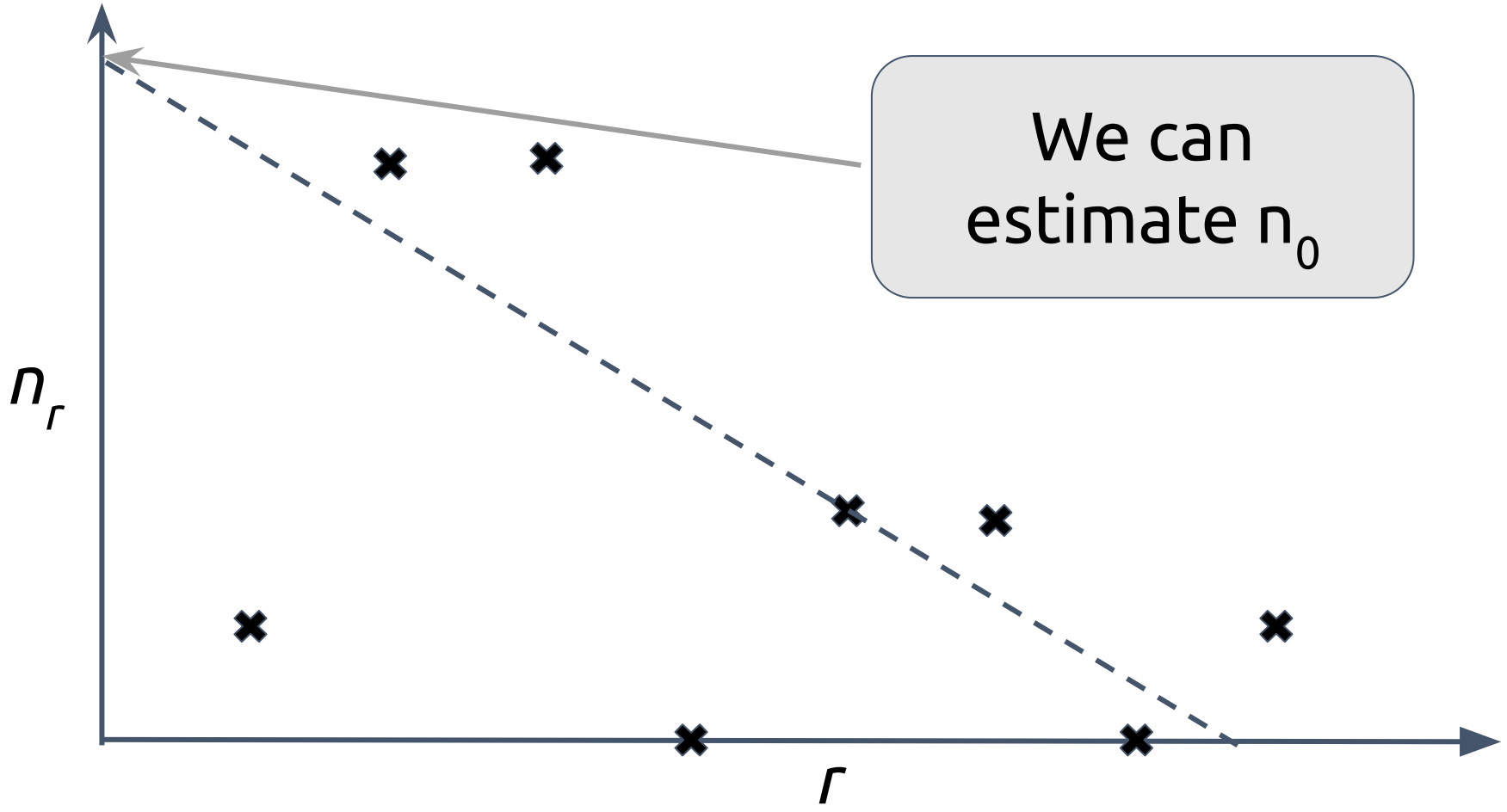
$$p(\text{seen}) = \frac{\sum_{w \in W} c(w) + \alpha|V|}{\sum_{w \in W} c(w) + \alpha(|V| + N)}$$

$$p(\text{unseen}) = \frac{\alpha N}{\sum_{w \in W} c(w) + \alpha(|V| + N)}$$

Count of counts

n_1	*anywhere*	1
n_2	*Sam-I-am, you, here, or there*	5
n_3	*green, eggs, and, ham, would*	5
n_4		0
n_5	*do, them*	2
n_6	*I, not*	2
n_7		0
n_8	*like*	1

Linear fit on count-of-counts



Good-Turing Smoothing

Idea: Assign probability mass of words that occur *r + 1* times to words that occur *r* times

$$p(\text{words occuring } r \text{ times}) = \frac{rn_r}{\sum_{r'=0,1,\ldots} r'n_{r'}}$$

Thus p of a word is

$$p(w) = \frac{c^*(w)}{\sum_{v \in V} c^*(v)}$$

$$c^*(w) = (c(w) + 1)\frac{n_{c(w)+1}}{n_{c(w)}}$$

Good-Turing Smoothing

n_r is difficult to calculate as

- Unknown for n_0
- Often zero for high *r*
- Or poorly estimated for high *r*

Implementations of Good-Turing Smoothing thus use linear regression to estimate n_r

Overview

Refresher on Probability

Estimating probabilities

n-grams

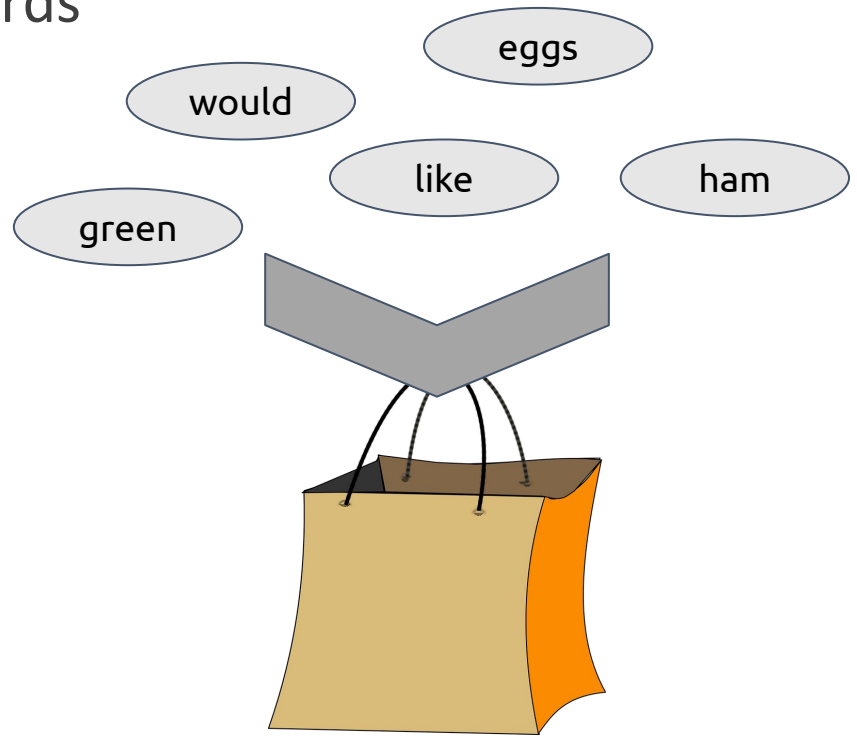
Language Model Smoothing

Evaluation of Language Models

Summary

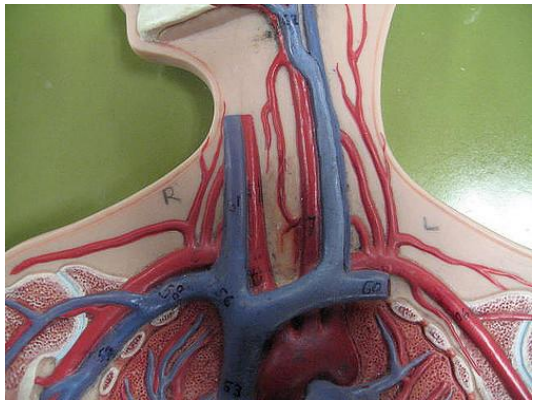
Bag-of-words

Up until now we have treated words as disconnected units.
This is called ‘bag-of-words’



Context

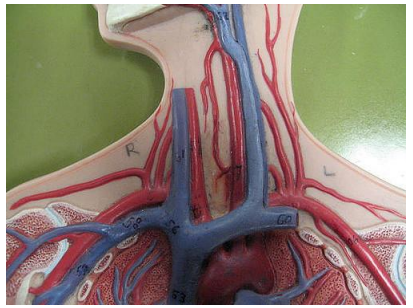
vessel



Context

fishing vessel

blood vessel



copper vessel

Probability of a sentence (language modelling)

Recalling the definition of conditional probability

$$p(w_1 w_2 \dots w_n) = p(w_n | w_1 \dots w_{n-1}) p(w_1 \dots w_{n-1})$$

Hence by induction

$$p(w_1 w_2 \dots w_n) = \prod_{k=1,\dots,n} p(w_k | w_1 \dots w_{k-1})$$

IMPORTANT: Note the order here. e.g., $p(w_4|w_1 w_2 w_3)$

N-gram approximation

As seen already ‘data sparsity’ is a major issue, this is nearly impossible for long-sequences of words (n-grams)

Solution: Approximate using a fixed window of previous words

$$p(w_1 w_2 \dots w_n) = \prod_{k=1,\dots,n} p(w_k | w_{k-m+1} \dots w_{k-1})$$

1-gram = unigram
2-gram = bigram
3-gram = trigram
4-gram = four-gram
etc.

Look back only *m - 1* words

Example corpus

Do you like green eggs and ham?
I do not like them, Sam-I-am.
I do not like green eggs and ham.
Would you like them here or there?
I would not like them here or there.
I would not like them anywhere.
I do not like green eggs and ham.
I do not like them Sam-I-am.

Bigram counts

and there	1	*here or*	1	*them Sam-I-am*	2
and ham	3	*I do*	4	*them here*	2
do you	1	*I would*	2	*them anywhere*	1
do not	4	*like green*	3	*would you*	1
eggs and	3	*like them*	5	*would not*	2
green eggs	3	*not like*	6	*you like*	2
here and	1	*or there*	1		

Bigram probabilities

$$p(\text{ham}|\text{and}) = \frac{c(\text{and ham})}{c(\text{and ham}) + c(\text{and there})}$$

Do not do the following. It is not correct. Why?

$$p(\text{ham}|\text{and}) \neq \frac{c(\text{and ham})}{c(\text{and})}$$

Bigram probabilities

p(there|and)	0.25	**p(or|here)**	0.50	**p(Sam-I-am|them)**	0.40
p(ham|and)	0.75	**p(do|I)**	0.67	**p(here|them)**	0.40
p(you|do)	0.20	**p(would|I)**	0.33	**p(anywhere|them)**	0.20
p(not|do)	0.80	**p(green|like)**	0.38	**p(you|would)**	0.33
p(and|eggs)	1.00	**p(them|like)**	0.62	**p(not|would)**	0.67
p(eggs|green)	1.00	**p(like|not)**	1.00	**p(like|you)**	1.00
p(and|here)	0.50	**p(there|or)**	1.00		

Applying bigram probabilities

$p(\text{“would you like green eggs and ham”})$

$\cong p(\text{“would”}) \times p(\text{“you”}|\text{”would”}) \times p(\text{“like”}|\text{”you”}) \times p(\text{“green”}|\text{”like”}) \times p(\text{“eggs”}|\text{”green”}) \times p(\text{“and”}|\text{”eggs”}) \times p(\text{“ham”}|\text{”and”})$

$= 0.05 \times 0.33 \times 1.0 \times 0.38 \times 1.0 \times 1.0 \times 0.75$

$= 0.0047$

Machine Translation

For machine translation we wish to choose the best translation

$$p(\text{“see the band live”} \mid \text{“die Band live sehen”})$$

vs.

$$p(\text{“see the volume live”} \mid \text{“die Band live sehen”})$$

Thus, we wish to find

$$\operatorname{argmax} p(\mathbf{t} \mid \mathbf{f})$$

where **t** is translation, **f** is foreign text.

Machine Translation

By Bayes' Law:

$$p(\mathbf{t}_1|\mathbf{f}) = p(\mathbf{f} \mid \mathbf{t}_1) \times p(\mathbf{t}_1) \div p(\mathbf{f})$$

$$p(\mathbf{t}_2|\mathbf{f}) = p(\mathbf{f} \mid \mathbf{t}_2) \times p(\mathbf{t}_2) \div p(\mathbf{f})$$

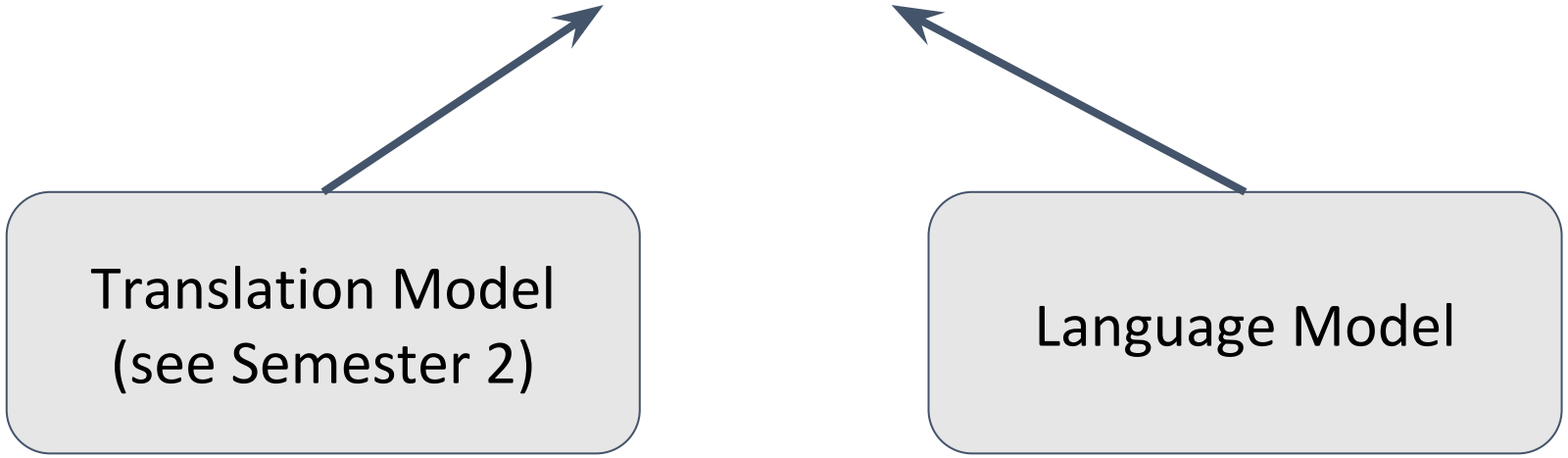
It follows that $p(\mathbf{t}_1|\mathbf{f}) > p(\mathbf{t}_2|\mathbf{f})$ iff

$$p(\mathbf{f} \mid \mathbf{t}_1) \times p(\mathbf{t}_1) > p(\mathbf{f} \mid \mathbf{t}_2) \times p(\mathbf{t}_2)$$

Noisy channel model for MT

Thus find the maximum of $p(\boldsymbol{t}|\boldsymbol{f})$ is equivalent to finding:

$$\text{argmax } p(\boldsymbol{f}|\boldsymbol{t})\ p(\boldsymbol{t})$$



Language Models for Machine Translation

Say we have a foreign sentence

“die Band live sehen”

Given our noisy channel model we can say:

$$p(\text{“see the band live”} \mid \text{”die Band live sehen”}) \propto$$
$$p(\text{“die Band live sehen”} \mid \text{“see the band live”}) \times p(\text{“see the band live”})$$

Language models for Machine Translation

Let's say the translation model generates to candidate translations

p("die Band live sehen"|"see the band live") = 0.005

p("die Band live sehen"|"see the volume live") = 0.008

Under a bigram model

p("see the band live") ≅ p("see") × p("the"|"see") × p("band"|"the") × p("live"|"band")

p("see the volume live") ≅ p("see") × p("the"|"see") × p("volume"|"the") × p("live"|"volume")

Probabilities (based on Wikipedia)

the •	120,998,336	*band* •	663,338	*volume* •	107,180
the band	309,897	*band live*	235	*volume live*	2
the volume	12,147				

$p(\text{“band”}|\text{”the”}) \times p(\text{“live”}|\text{”band”}) = 0.0026 \times 0.00035 = 1.2 \times 10^{-6}$
$p(\text{“volume”}|\text{”the”}) \times p(\text{“live”}|\text{”volume”}) = 0.00010 \times 0.000019 = 1.9 \times 10^{-9}$

$p(\text{”see the band live”}|\text{“die Band live sehen”}) \propto 0.005 \times 1.2 \times 10^{-6} = 5.9 \times 10^{-9}$
$p(\text{”see the volume live”}|\text{“die Band live sehen”}) \propto 0.008 \times 1.9 \times 10^{-9} = 1.5 \times 10^{-11}$

Overview

Refresher on Probability

Estimating probabilities

n-grams

Language Model Smoothing

Evaluation of Language Models

Summary

Higher-order n-gram smoothing

Consider we have two bigrams which don't occur in the corpus, e.g.,

$$c(\text{“I ham”}) = 0$$

$$c(\text{“I like”}) = 0$$

Then (also with Good-Turing smoothing) it follows

$$p(\text{“ham”}|\text{”I”}) = p(\text{“like”}|\text{”I”})$$

However, p(“like”) > p(“ham”) therefore it follows that we should have

$$p(\text{“ham”}|\text{”I”}) < p(\text{“like”}|\text{”I”})$$

Back-off

Simple solution: if $c(w_1 w_2) = 0$ use $c(w_2)$

However, now we do not have a true probability distribution (and overestimate some unlikely n-grams)

So we must discount the counts

$$c_{\text{back-off}}(w_1 w_2) = \begin{cases} d_r c(w_1 w_2) & c(w_1 w_2) > 0 \\ \alpha c(w_2) & c(w_1 w_2) = 0 \end{cases}$$

Where d_r is the probability mass of seen examples and α is such that

$$\sum_{w \in V} c_{\text{back-off}}(w_1 w) = \sum_{w \in V} c(w_1 w)$$

Interpolation

We simply interpolate the unigram probability into the bigram probability as follows

$$p_{JM}(w_n|w_{n-1}) = (1-\lambda_1)p(w_n|w_{n-1}) + \lambda_1 p(w_n)$$

Generalizes to higher n-grams as

$$\begin{aligned} & p_{JM}(w_n|w_{n-k+1}\cdots w_{n-1}) = \\ & (1-\lambda_{k-1})p(w_n|w_{n-k+1}\cdots w_{n-1}) + \\ & \lambda_{k-1}p(w_n|w_{n-k+2}\cdots w_{n-1}) \end{aligned}$$

Witten-Bell Smoothing

Some bigrams nearly occur almost as much as the unigram, e.g., *“San Francisco”, “helter skelter”*

p(“helter skelter”) ≅ p(“skelter”)

We should therefore have that

p(“UNK|helter”) ≪ p(“skelter|helter”)

We achieve this with the ‘diversity of history’

$$N_{1+}(w_1\bullet) = |\{w_2 : c(w_1 w_2) > 0\}|$$

$$\lambda = \frac{N_{1+}(w_1\bullet)}{N_{1+}(w_1\bullet) + c(w_1\bullet)}$$

Witten-Bell Interpolation Example

$$\lambda_{helter} = \frac{20}{20 + 459} = 0.042$$

$$\lambda_{skelter} = \frac{60}{60 + 430} = 0.122$$

Probability mass for unknown following words

Overview

Refresher on Probability

Estimating probabilities

n-grams

Language Model Smoothing

Evaluation of Language Models

Summary

What makes a good language model?

Should produce high probabilities for valid (English) sentences. Should produce low probabilities for ungrammatical sentences

Evaluation normally looks only at first condition. If language model produces valid probability distribution

Perplexity is the most widely used evaluation metric

$$H(p_{LM}) = -\frac{1}{n}\log p_{LM}(w_1, \ldots, w_n)$$

$$PP = 2^{H(p_{LM})}$$

Perplexity Example

<s> I do so like green eggs and ham! </s>
<s> Thank you </s>
<s> Thank you Sam-I-am </s>

Add start and end tokens to show sentence boundaries

$$H(p) = \frac{1}{19}(p(\mathrm{I}|<\mathrm{s}>) + \ldots + p(</\mathrm{s}>|\mathrm{Sam\text{-}I\text{-}am})$$

State-of-the-Art for Language Models

Method	Perplexity
Add-one	383.2
Add-alpha	113.2
Good-Turing	112.9

Koehn (2009)

Method	Perplexity
Kneser-Ney 5-gram	125.7
Recurrent Neural Network	101.0
RNN + KN5	92.9

Mikolov (2012)

Method	2-gram	3-gram	4-gram
Good-Turing	96.2	62.9	59.9
Interpolation (Witten-Bell)	97.1	63.8	60.4
Back-off (Modified Kneser-Ney)	95.4	61.6	58.6
Interpolated back-off	94.5	59.3	54.0

Koehn (2009)

Statistical Machine Translation, P. Koehn (2009).
Statistical language models based on neural networks. T. Mikolov (2012)

Overview

Refresher on Probability

Estimating probabilities

n-grams

Language Model Smoothing

Evaluation of Language Models

Summary

Key points

- Probability of words is used to choose most-likely correct solution
- Noisy Channel Model is frequently used
  - $p(A|B) \propto p(B|A)\, p(A)$
- Probabilities can be estimated from corpora
  - Data sparsity makes this hard
  - Out-of-vocabulary words must also be handled in applications
- Smoothing is technique to assign probabilities to unseen and low probability events
  - Assumes there is a ‘probability mass’ associated with unseen events
  - This can be estimated by linear count-of-counts
  - Good-Turing is a method that assigns probability mass of higher counts to lower counts

Key points

Context is vital for understanding language

We can use n-grams to model this n-gram language model, e.g. bigram model over three words

$$p(w_1 w_2 w_3) \cong p(w_3|w_2) \times p(w_2|w_1) \times p(w_1)$$

n-gram language models are used to choose most likely output in machine translation, spelling correction, etc.

Higher-order n-gram probabilities can be smoothed by using lower-order probabilities

- Back-off
- Interpolation

Lab of this Week

Exercises on word frequencies, n-grams, probabilistic modelling for text classification.

QA