COMP 472: Artificial Intelligence
Natural Language Processing part 5
n-gram Model video 3

- Russell & Norvig: Section 23.1.3, 23.1.4

Today

1. Introduction
2. Bag of word model
3. n-gram models 

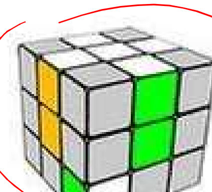
4. Deep Learning for NLP
    1. Word Embeddings
    2. Recurrent Neural Networks

n-gram Model

length

- An n-gram model is a probability distribution over sequences of events (grams/units/items)
- models the order of the events
- Used when the past sequence of events is a good indicator of the next event to occur in the sequence
- i.e. To predict the next event in a sequence of event

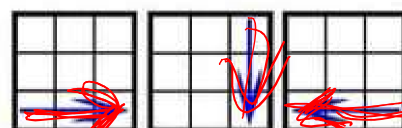
Eg:



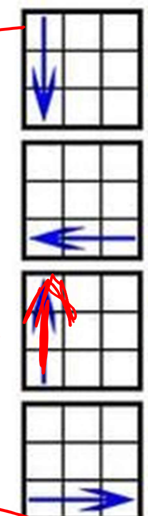
next move of player based on past moves
    left right right up ... up? down? left? right?

NL

next word based on past words
    Hi dear, how are ... pencil? laptop? you? magic?
    -4 -3 -2 -1 t



t-3 t-2 t-1



t

next base pair based on past DNA sequence
    AGCTTCG ... A? G? C? T?

https://www.pinterest.com.au/pin/241716704985206767/

What's a Language Model?

- A Language model is a n-gram model over word/character sequences
- ie: events = words or events = character or grams = larger syntactic constituent

  grams grams

- P("I'd like a coffee with 2 sugars and milk") ≈ 0.001
- P("I'd hike a toffee with 2 sugars and silk") ≈ 0.000000001

Applications of LM

Rule-based ≡ Systran
Statistical ML ≡ Google
Neural ML = Google Today

- Speech Recognition
- Statistical Machine Translation
- Language Identification
- Word Prediction
- Spelling Correction
  - *He is trying to <u>fine</u> out.*
  - *He is trying to <u>find</u> out.*
- Optical character recognition
- Handwriting recognition
- Natural Language Generation
- …

In Speech Recognition

Given: Observed sound - O

Find: The most likely word/sentence – S*

S1: How to recognize speech. ?

S2: How to wreck a nice beach. ?

S3: …

- Goal: find most likely sentence (S*) given the observed sound (O) …
- ie. pick the sentence with the highest probability: $S^* = \underset{S \in L}{\operatorname{argmax}} P(S \mid O)$
- We can use Bayes rule to rewrite this as: $S^* = \underset{S \in L}{\operatorname{argmax}} \frac{P(O \mid S)P(S)}{\cancel{P(O)}}$
- Since denominator is the same for each candidate S, we can ignore it for the argmax:

$$S^* = \underset{S \in L}{\operatorname{argmax}} \boxed{P(O \mid S)} \times \boxed{P(S)}$$

Acoustic model -- Probability of the possible phonemes in the language + Probability of ≠ pronunciations

Language model -- P(a sentence) Probability of the candidate sentence in the language

In Speech Recognition

$$S^* = \operatorname*{argmax}_{S \in L} P(O \mid S) \times P(S)$$

Acoustic model

Language model

LM

$$\operatorname*{argmax}_{\text{word sequence}} P(\overset{S}{\text{word sequence}} \mid \overset{O}{\text{acoustic signal}})$$

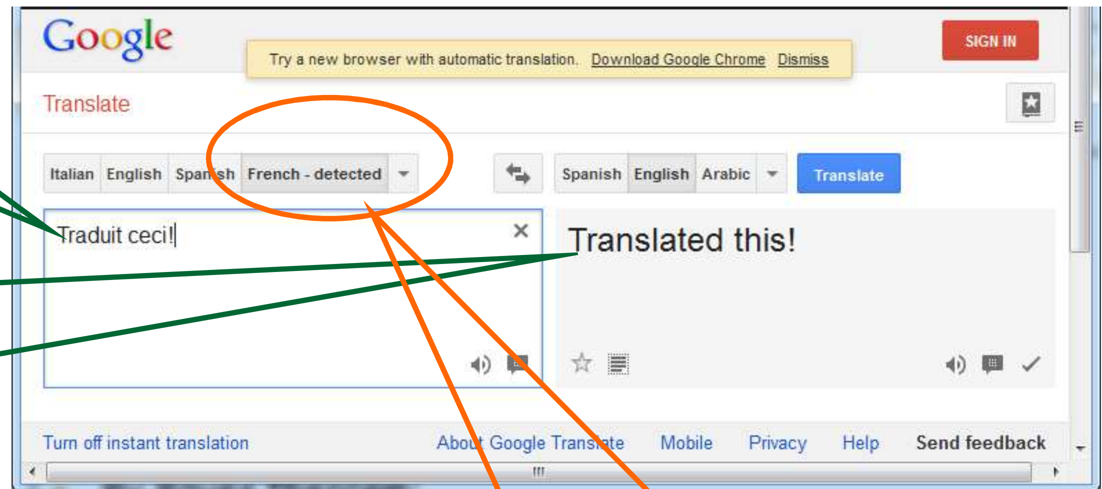
$$\operatorname*{argmax}_{\text{wordsequence}} \frac{P(\overset{O}{\text{acoustic signal}} \mid \text{word sequence}) \times P(\text{word sequence})}{P(\text{acoustic signal})}$$

$$\operatorname*{argmax}_{\text{wordsequence}} P(\text{acoustic signal} \mid \text{word sequence}) \times P(\boxed{\text{word sequence}})$$

In Statistical Machine Translation

- Assume we translate from fr[foreign] to English i.e: (en|fr)

Given: Foreign sentence - fr



Find: The most likely English sentence – en*

S1: Translate that!

S2: Translate this!

S3: Eat your soup!

S4…

Automatic Language Identification… guess how that's done?

faithful × fluent

$$en^* = \underset{en}{\operatorname{argmax}}\, P(fr \mid en) \times P(en)$$

Translation model

Language model

le bug de l'an 2000

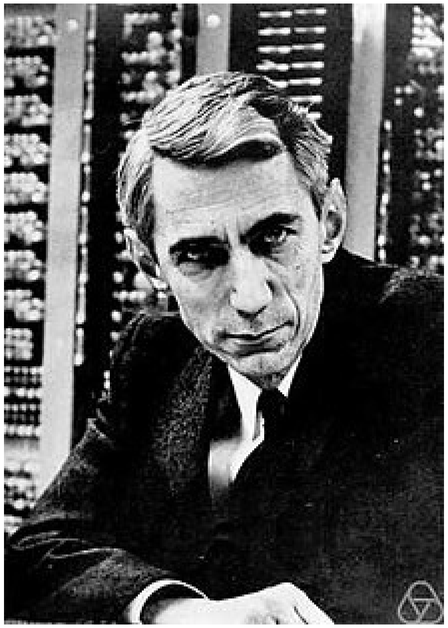
en$_1$ The bug of year 2000

en$_2$ Y2K bug

"Shannon Game" (Shannon, 1951)

When are you and Max going to tie the ___ ? knot

- Predict the next word/character given the *n-1* previous words/characters.



https://en.wikipedia.org/wiki/Claude_Shannon

1^{st} approximation

- each word has an equal probability to follow any other
  - with 100,000 words, the probability of each word at any given point is .00001 $\frac{1}{100000}$

- problem…
  - some words are more frequent than others…
  - *eg*. "*the*" appears many more times, than "*rabbit*"

2nd approximation: unigrams

n = 1

- take into account the frequency of the word in some training corpus
  - at any given point, "*the*" is more probable than "*rabbit*"

    $\frac{100}{100000}$ $\frac{5}{100\,000}$

- problem...
  - does not take word order into account.
  - this is the bag of word approach.
  - "~~*Just then, the white*~~ ..."

    the

- solution...
  - the probability of a word should depend on the previous words (the history)

$$P(\underset{n}{w_n} \mid \underbrace{\boxed{w_1 w_2 \ldots w_{n-1}}}_{\text{history}})$$

n-1 words

What size should *n* be?

n → size of n-gram
history → n-1

- *Examples*
  - *the large green ________.* n = 4
    - *mountain? tree?*
  - *Sue swallowed the large green ________.* n = 6
    - ~~*mountain? tree?*~~ *pill? broccoli?*

V = 100 000

large n
⇒ 1) large size of model
2) model is unreliable

- Knowing that Sue "*swallowed*" helps narrow down possibilities
- ie. Going back 3 words before helps
- But, how far back do we look?

V = 26 char.

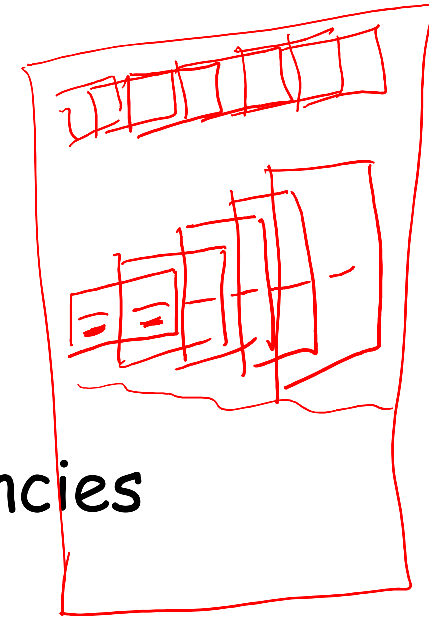
Bigrams

n = 2

- first-order Markov models

$$P(w_n | w_{n-1})$$

history = 1 (i.e. n-1)



- N-by-N matrix of probabilities/frequencies
- N = size of the vocabulary we are using

2nd word V = 100 000

1st word V = 100 000

	a	aardvark	aardwolf	aback	...	zoophyte	zucchini
a	0	0	0	0	...	8	5
aardvark	0	0	0	0	...	0	0
aardwolf	0	0	0	+1 0	...	0	x1 0
aback	26	1	6	0	...	12	2
...	...	...	...	...	...	...	...
zoophyte	0	0	0	1	...	0	0
zucchini	0	0	0	3	...	0	0

Trigrams

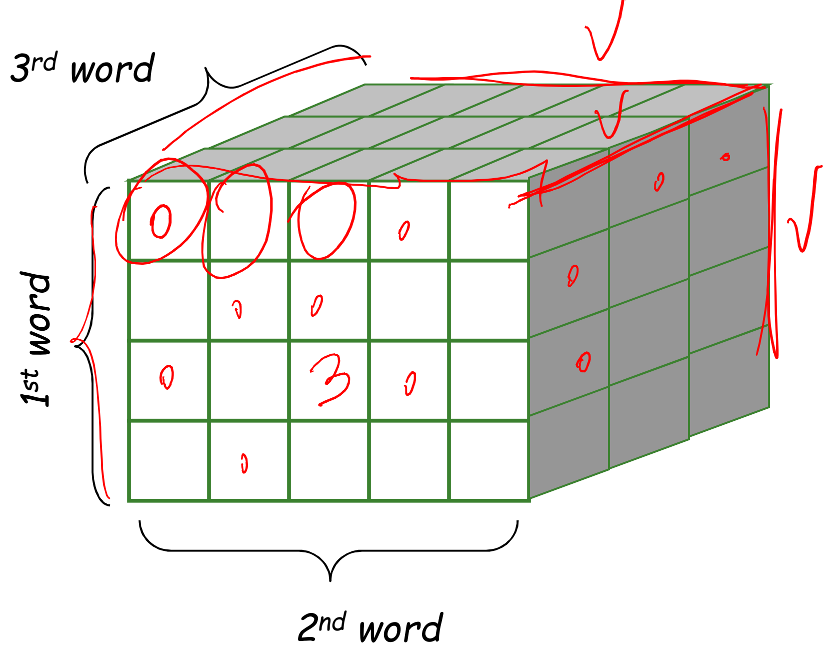
n = 3

- second-order Markov models

$$P(w_n | w_{n-1} w_{n-2})$$

history = 2

- N-by-N-by-N matrix of probabilities/frequencies
- N = size of the vocabulary we are using



a a a → 0

the little cat → 3

→ 0

→ 0

Why use only bi- or tri-grams?

- Markov approximation is still costly

  with a 20 000 word vocabulary:
  - bigram needs to store 400 million parameters
  - trigram needs to store 8 trillion parameters
  - using a language model > trigram is impractical

Building n-gram Models

i.e. larger collection of document Wikipedia dump.

1. Data preparation:
   - Decide on training corpus
   - Clean and tokenize
   - How do we deal with sentence boundaries?
     - I eat. I sleep.
       - (I eat) (eat I) (I sleep)
     - <s>I eat <s> I sleep <s>
       - (<s> I) (I eat) (eat <s>) (<s> I) (I sleep) (sleep <s>)

decide and be consistent

V = {100 000}

V = {100 000 + punctuation + ...}

Building n-gram Models

2. Count words and build model
   - Let $C(w_1 \ldots w_n)$ be the frequency of n-gram $w_1 \ldots w_n$

$$\boxed{P(w_n \mid w_1 \ldots w_{n-1}) = \frac{C(w_1 \ldots w_n)}{C(w_1 \ldots w_{n-1})}}$$

$\dfrac{\text{history} + w_n}{\text{history}}$

3. Smooth your model (see later)

Example 1:

- in a training corpus, we have 10 instances of "*come across*"
  - 8 times, followed by "*as*"
  - 1 time, followed by "*more*"
  - 1 time, followed by "*a*"
- so we have:
  - $$P(as \mid come\ across) = \frac{C(come\ across\ as)}{\underbrace{C(come\ across)}_{\text{history}}} = \frac{8}{10}$$ trigram model
  - P(more | come across) = 0.1
  - P(a | come across) = 0.1
  - P(X | come across) = 0 where X ≠ "*as*", "*more*", "*a*"

Example 2:

bi-gram

P(on|eat) = .16	P(want|I) = .32	P(eat|to) = .26
P(some|eat) = .06	P(would|I) = .29	P(have|to) = .14
P(British|eat) = .001	P(don't|I) = .08	P(spend|to)= .09
…	…	…
P(I|<s>) = .25	P(to|want) = .65	P(food|British) = .6
P(I'd|<s>) = .06	P(a|want) = .5	P(restaurant|British) = .15
…	…	…

P(*I want to eat British food*)
= P(I|<s>) x P(want|I) x P(to|want) x P(eat|to) x P(British|eat) x P(food|British)
= .25 x .32 x .65 x .26 x .001 x .6
= .000008

P(sleep)

<

Remember this slide...

Be Careful: Use Logs

- if we really do the product of probabilities...
  - $\text{argmax}_{cj}\ P(c_j) \prod P(w_i|c_j)$
  - we soon have numerical underflow...
  - ex: 0.01 × 0.02 × 0.05 × ...
- so instead, we add the log of the probs
  - $\text{argmax}_{cj}\ \cancel{\log(P(c_j))} + \sum \log(P(w_i|c))$
  - ex: log(0.01) + log(0.02) + log(0.05) + ...

45

Some Adjustments

- product of probabilities… numerical underflow for long sentences
- so instead of multiplying the probs, we add the log of the probs

P(*I want to eat British food)*
= log(P(I|<s>)) + log(P(want|I)) + log(P(to|want)) + log(P(eat|to)) + log(P(British|eat)) + log(P(food|British))
= log(.25) + log(.32) + log(.65) + log (.26) + log(.001) + log(.6)

Problem: Data Sparseness

- What if a sequence never appears in training corpus? P(X)=0
  - *"come across the men" --> prob = 0*
  - *"come across some men" --> prob = 0*
  - *"come across 3 men" --> prob = 0*

P(...... come across some candy ..)
= 0

- The model assigns a probability of zero to unseen events …
- probability of an n-gram involving unseen words will be zero!

sequence

- Solution: smoothing
  - decrease the probability of previously seen events
  - so that there is a little bit of probability mass left over for previously unseen events

Remember this other slide...

Be Careful: Smooth Probabilities

- normally: $P(w_i \mid c_j) = \dfrac{(\text{frequency of } w_i \text{ in } c_j)}{\text{total number of words in } c_j}$
- what if we have a $P(w_i|c_j) = 0$...?
  - ex. the word "dumbo" never appeared in the class SPAM?
  - then P("dumbo"| SPAM) = 0
- so if a text contains the word "dumbo", the class SPAM is completely ruled out !
- to solve this: we assume that every word always appears at least once (or a smaller value)
  - ex: add-1 smoothing:

$\lambda = 0.5$

$$P(w_i \mid c_j) = \frac{(\text{frequency of } w_i \text{ in } c_j) + 1}{\text{total number of words in } c_j + \text{size of vocabulary}}$$

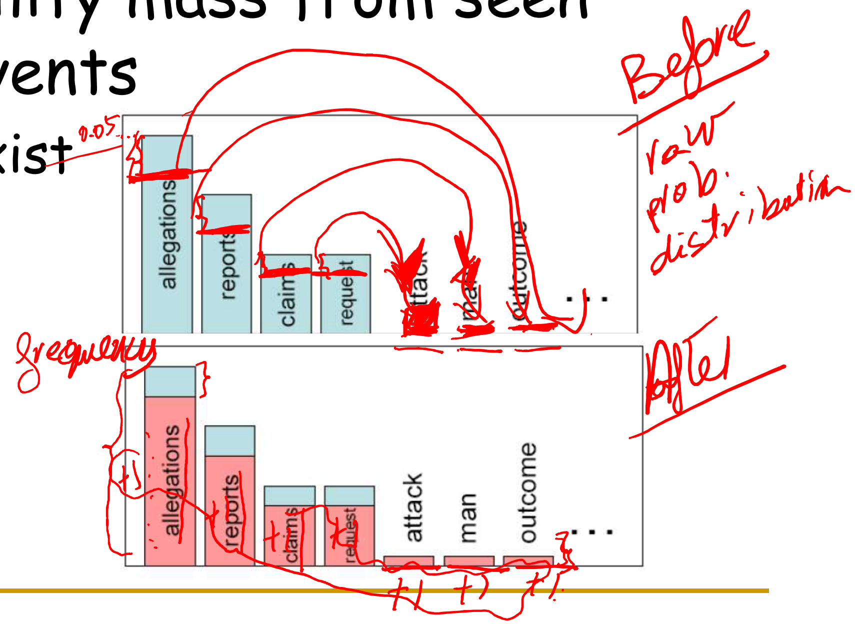
44

Add-one Smoothing

- Pretend we have seen every n-gram one more time than we actually did

  newCount(n-gram) = oldCount(n-gram) + 1

- The idea is to “steal” probability mass from seen events to give it to unseen events
  - various smoothing techniques exist
  - depending on:
    - how you steal from the rich
    - how you redistribute to the poor



https://disney.fandom.com/wiki/Robin_Hood_(character)
https://www.usna.edu/Users/cs/nchamber/courses/nlp/f13/slides/set4-smoothing.pdf

Add-one: Example

unsmoothed bigram counts (frequencies):

2nd word

1st word

	I	want	to	eat	Chinese	food	lunch	…	Total
I	8	1087	0	13	0	0	0		C(I)=3437
want	3	0	786	0	6	8	6		C(want)=1215
to	3	0	10	860	3	0	12		C(to)=3256
eat	0	0	2	0	19	2	52		C(eat)=938
Chinese	2	0	0	0	0	120	1		C(Chinese)=213
food	19	0	17	0	0	0	0		C(food)=1506
lunch	4	0	0	0	0	1	0		C(lunch)=459
…									…
									…
									N=10,000

- Assume a vocabulary of V= 1616 (different) words
  - V = {a, aardvark, aardwolf, aback, … , I, …, want,… to, …, eat, Chinese, …, food, …, lunch, …, zoophyte, zucchini}
  - |V| = 1616 words
- And a total of N = 10,000 bigrams (~word instances) in the training corpus

Add-one: Example

unsmoothed bigram counts: 2nd word

1st word

	I	want	to	eat	Chinese	food	lunch	…	**Total**
I	8	1087	0	13	0	0	0		C(I)=3437
want	3	0	786	0	6	8	6		C(want)=1215
to	3	0	10	860	3	0	12		C(to)=3256
eat	0	0	2	0	19	2	52		C(eat)=938
Chinese	2	0	0	0	0	120	1		C(Chinese)=213
food	19	0	17	0	0	0	0		C(food)=1506
lunch	4	0	0	0	0	1	0		C(lunch)=459
…									
									N=10,000

n-gram prob. $P(w_1, w_2)$

unsmoothed bigram conditional probabilities:

	I	want	eat			Σ
I	8/3437 +	1087/3437 +	. .	+	+	1
want	3/1215	0	0			1
						1

1616

note:

$P(II) = \frac{8}{10\,000}$

$P(I \mid I) = \frac{8}{3\,437}$

$P(w_2 \mid w_1)$

Add-one: Example (con't)

1616

add-one smoothed bigram counts:

	I	want	to	eat	Chinese	food	lunch	…	Total
I	~~8~~ +1 9	~~1087~~ +1 ≈1088	1	14	1	1	1		~~3437~~ C(I) + |V| = 5053
want	~~3~~ +1 4	0+1 ≈ 1	787	1	7	9	7		C(want) + |V| = 2831
to	4	1	**11**	**861**	**4**	**1**	**13**		C(to) + |V| = 4872
eat	1	1	23	1	20	3	53		C(eat) + |V| = 2554
Chinese	3	1	1	1	1	121	2		C(Chinese) + |V| = 1829
food	20	1	18	1	1	1	1		C(food) + |V| = 3122
lunch	5	1	1	1	1	2	1		C(lunch) + |V| = 2075
…									~~total = 10,000~~ $N+\|V\|^2 = 10{,}000 + (1616)^2 = 2{,}621{,}456$

1616

+V

V^2

+1

5000 000 real corpus

+|V| … 1

add-one bigram conditional probabilities:

	I	want	to	eat	Chinese	food	lunch	…
I	.0018 (9/5053)	.215	.00019	.0028	.00019	.00019	.00019	
want	.0014	.00035	.278	.00035	.0025	.0031	.00247	
to	.00082	.0002	.00226	.1767	.00082	.0002	.00267	
eat	.00039	.00039	.0009	.00039	.0078	.0012	.0208	
…								

Add-one, more formally

$$P_{Add1}(w_1\, w_2 \ldots w_n) = \frac{C(w_1\, w_2 \ldots w_n) + 1}{N + B}$$

N: size of the corpus

i.e. nb of n-gram tokens in training corpus

B: number of "bins"

i.e. nb of <u>different</u> n-gram types

i.e. nb of cells in the matrix

e.g. for bigrams, it's (size of the vocabulary)^2

Add-delta Smoothing

- problem with add-1 smoothing:
  - every previously unseen n-gram is given a low probability
  - but there are so many of them that too much probability mass is given to unseen events

- solution:
  - instead of adding 1, add some other (smaller) positive value δ

~~+1~~ .5 .25

$$P_{AddD}(w_1\, w_2 \ldots w_n) = \frac{C\,(w_1\, w_2 \ldots w_n) + \delta}{N + \delta B}$$

.5

  - most widely used value for δ = 0.5

- better than add-one, but still not great…

Factors of Training Corpus

- Size:
  - the more, the better
  - but after a while, not much improvement…
    - bigrams (characters) after $\boxed{\text{100's million words}}$
    - trigrams (characters) after some billions of words

A … Z V=26

- Genre (adaptation):
  - training on cooking recipes and testing on aircraft maintenance manuals

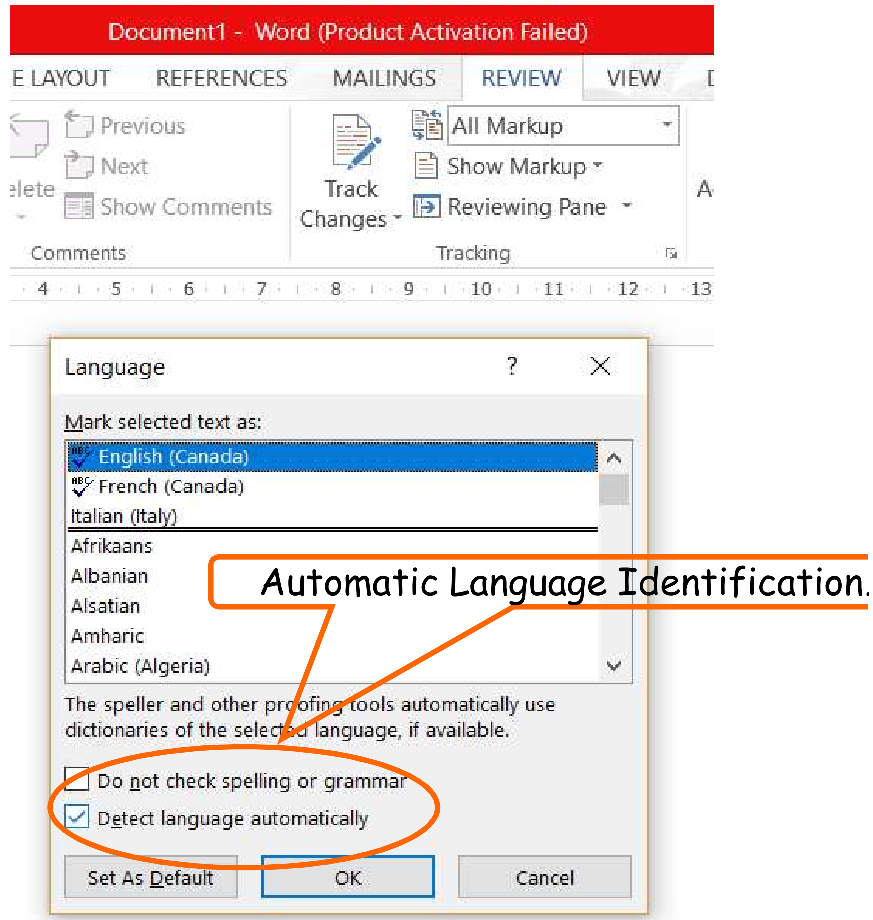
in a b g … z

I am going to the dentist
d …
Susan - 20 yr student

I am going to the d { dentist daycare
Jack - 30 yr daddy

Example: Language Identification

- hypothesis: texts that resemble each other (same author, same language) share similar character/word sequences
  - In English character sequence "*ing*" is more probable than in French
- Training phase:
  - construction of the language model
  - with pre-classified documents (known language/author)
- Testing phase:
  - apply language model to unknown text



Example: Language Identification

- bigram of characters
  - characters = 26 letters (case insensitive)
  - possible variations: case sensitivity, punctuation, beginning/end of sentence marker, …

Example: Language Identification

$V = 26$

1. Train a character-based language model for Italian:

	A	B	C	D	...	Y	Z
A	0.0014	0.0014	0.0014	0.0014	...	0.0014	0.0014
B	0.0014	0.0014	0.0014	0.0014	...	0.0014	0.0014
C	0.0014	0.0014	0.0014	0.0014	...	0.0014	0.0014
D	0.0042	0.0014	0.0014	0.0014	...	0.0014	0.0014
E	0.0097	0.0014	0.0014	0.0014	...	0.0014	0.0014
...	...	...	...	...	...	...	0.0014
Y	0.0014	0.0014	0.0014	0.0014	...	0.0014	0.0014
Z	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014

$V = 26$

Σ 1 1 1

$|26$

bigram

italian

vs

2. Train a character-based language model for Spanish:

	A	B	C	D	...	Y	Z
A	0.0014	0.0014	0.0014	0.0014	...	0.0014	0.0014
B	0.0014	0.0014	0.0014	0.0014	...	0.0014	0.0014
C	0.0014	0.0014	0.0014	0.0014	...	0.0014	0.0014
D	0.0042	0.0014	0.0014	0.0014	...	0.0014	0.0014
E	0.0097	0.0014	0.0014	0.0014	...	0.0014	0.0014
...	...	...	...	...	...	...	0.0014
Y	0.0014	0.0014	0.0014	0.0014	...	0.0014	0.0014
Z	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014	0.0014

Spanish

3. Given a unknown sentence "che bella cosa" is it in Italian or in Spanish?
   - P("che bella cosa") with the Italian LM $\quad p(h|c) \times p(e|h) \cdots p(a|s) =$
   - P("che bella cosa") with the Spanish LM $\quad p(h|c) \times p(e|h) \cdots =$
4. Highest probability -->language of sentence

$\Rightarrow$ Italian !

Google's Web 1T 5-gram model

- 5-grams
- generated from 1 trillion words
- 24 GB compressed
  - Number of tokens: 1,024,908,267,229
  - Number of sentences: 95,119,665,584
  - Number of unigrams: 13,588,391
  - Number of bigrams: 314,843,401
  - Number of trigrams: 977,069,902
  - Number of fourgrams: 1,313,818,354
  - Number of fivegrams: 1,176,470,663
- See discussion: http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html
- See Google Ngram Viewer: http://en.wikipedia.org/wiki/Google_Ngram_Viewer

Problem with n-grams

- Natural language is not linear ....
- there may be *long-distance dependencies.*
  - Syntactic dependencies
    - *The <u>man</u> next to the large oak tree near … <u>is</u> <u>tall</u>.*
    - *The <u>men</u> next to the large oak tree near … <u>are</u> tall.*
  - *Semantic dependencies*
    - *The <u>bird</u> next to the large oak tree near … <u>flies</u> rapidly.*
    - *The <u>man</u> next to the large oak tree near … <u>talks</u> rapidly.*
  - *World knowledge*
    - *<u>Michael Jackson</u>, who was featured in …, is <u>buried</u> in California.*
    - *<u>Michael Bublé</u>, who was featured in …, is <u>living</u> in California.*
  - …
- More complex models of language are needed to handle such dependencies.

Today

1. Introduction
2. Bag of word model
3. n-gram models
4. Deep Learning for NLP
   1. Word Embeddings
   2. Recurrent Neural Networks

Up Next

1. Introduction
2. Bag of word model
3. n-gram models
4. Deep Learning for NLP
    1. Word Embeddings
    2. Recurrent Neural Networks