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

n元语法

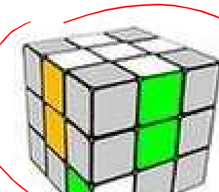
length

- An n-gram model is a probability distribution over sequences of events (grams/units/items) 一系列events发生的概率分布
- 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 通过过去发生的一系列事件，推断下一个事件几率
- 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?

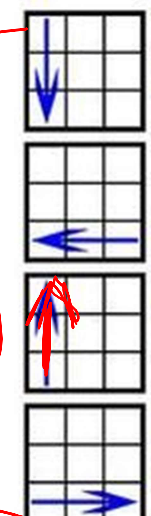
-4 -3 -2 -1 t

下个词是you

next base pair based on past DNA sequence

AGCTTCG ... A? G? C? T?

t-3 t-2 t-1



t

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

What's a Language Model?

language model是建立在字符串上的n-gram model

- A Language model is a n-gram model over word/character sequences

events =words或events=char都行

- 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

like a coffee几率远大于hike a toffee

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: $S^* = \underset{S\in L}{\operatorname{argmax}} P(S\,|\,O)$
- We can use Bayes rule to rewrite this as: $S^* = \underset{S\in L}{\operatorname{argmax}} \dfrac{P(O\,|\,S)P(S)}{P(O)}$
- Since denominator is the same for each candidate S, we can ignore it for the argmax:

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

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

我们关心的是language model也就是候选句的可能性

不能省略但我们并不关心他

In Speech Recognition

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

Acoustic model

Language model

LM

在有信号的情况下说出这句话最大的纪律等于

$$\underset{\text{word sequence}}{\operatorname{argmax}}\ P(\text{word sequence} \mid \text{acoustic signal})$$

S   O

S   P

O

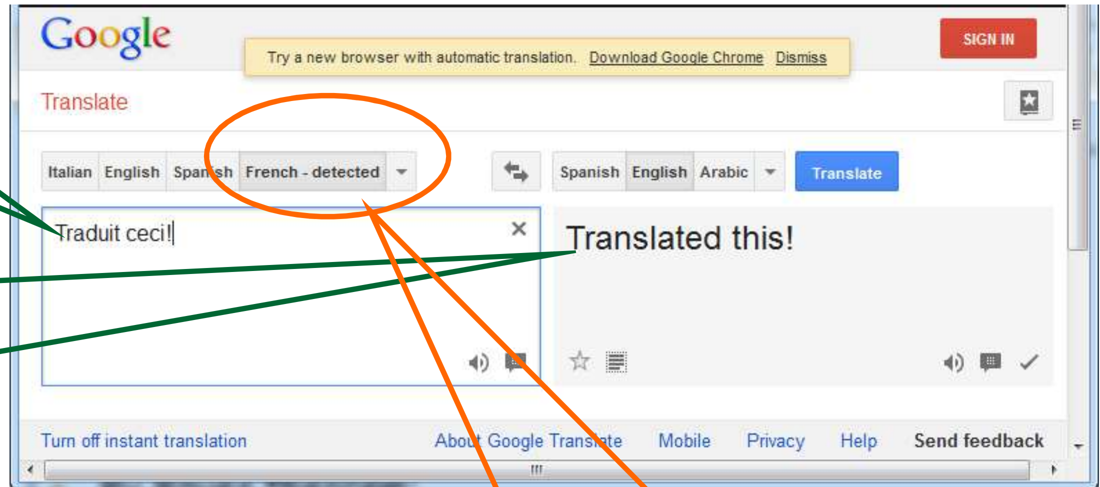
$$\underset{\text{wordsequence}}{\operatorname{argmax}}\ \frac{P(\text{acoustic signal} \mid \text{word sequence}) \times P(\text{word sequence})}{P(\text{acoustic signal})}$$

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

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

$$en^* = \underset{en}{\operatorname{argmax}}\ \underbrace{P(fr\,|\,en)}_{\text{Translation model}} \times \underbrace{P(en)}_{\text{Language model}}$$

le bug de l'an 2000

en1 The bug of year 2000

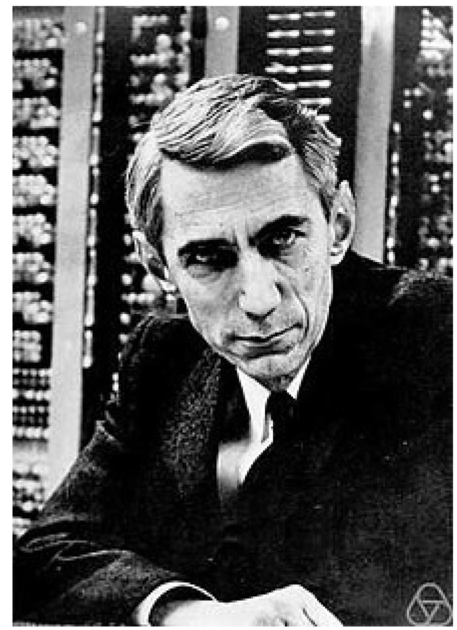
en2 Y2K bug

“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.

不看前文的，你任意选一个预测，叫做N-1,雏形



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

1st 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 $\frac{1}{100000}$

所有的都同几率

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

the比rabbit频繁太多

2nd approximation: unigrams

$n=1$

n还是等于1，意思是不考虑word-order

unigrams

- take into account the frequency of the word in some training corpus 语料库

  考虑频率，把每个word几率算出来

  - at any given point, "*the*" is more probable than "*rabbit*"

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

    the比rabbit更有可能

- 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) 取决于history

$$P(w_n \mid \boxed{w_1 w_2 \dots w_{n-1}})$$

n 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.

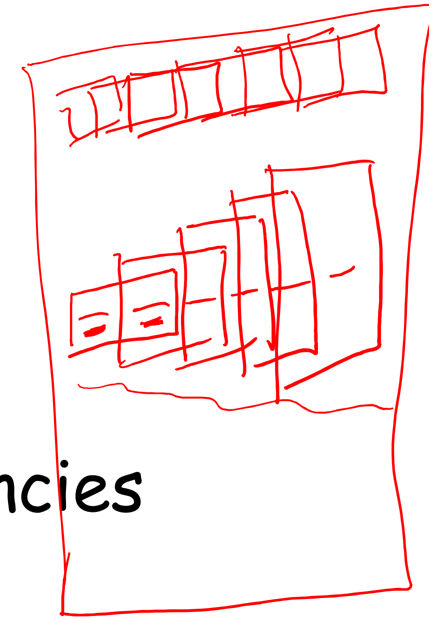
信息肯定是越多越好的，但是这样也慢，我们怎样取一个合适的值呢

bigram,能看前面的

Bigrams

$n = 2$

一阶Markov model



- 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

V= 100 000

1st word

	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	x 1 0
aback	26	1	6	0	…	12	2
…	…	…	…	…	…	…	…
zoophyte	0	0	0	1	…	0	0
zucchini	0	0	0	3	…	0	0

列一张表格，前面是x，后面的纪律

Trigrams n=3

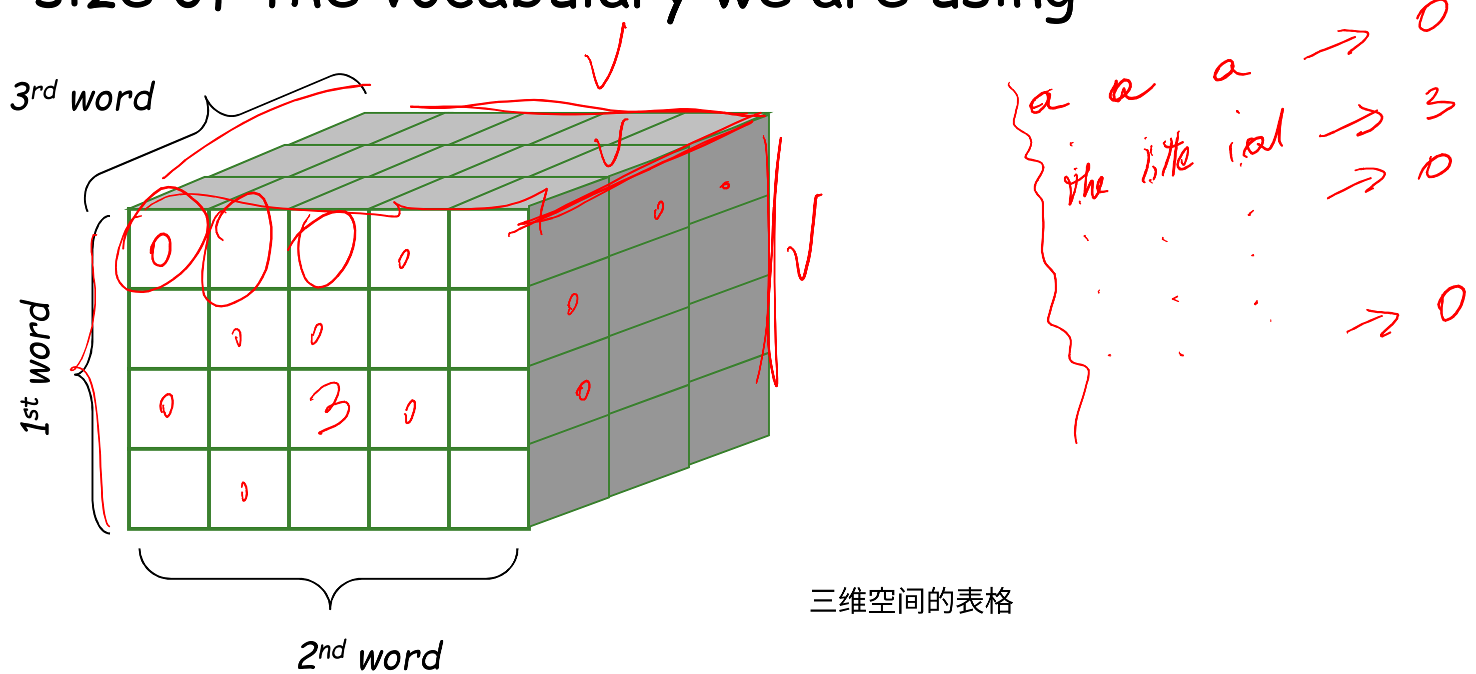
二阶Markov model

- second-order Markov models

history=2

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

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



三维空间的表格

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

bi tri已经要很大的数据了

Building n-gram Models

larger collection of document Wikipedia dump.

1. Data preparation:
   - Decide on training corpus i.e.

     准备训练用的语料库
   - 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>

       去除符号，留下空格
       - (<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 $w_1 \ldots w_n$

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

history + w_n

history

统计，我们要的combo的总次数除以前面那一串的总次数

3. Smooth your model (see later)

smooth

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*"

例如我们有10个come across
8个后面加as
1个后面加more
1个后面加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: $0.01 \times 0.02 \times 0.05 \times \ldots$
- so instead, we add the log of the probs
  - $\text{argmax}_{cj}\ \cancel{\log(P(c_j))} + \sum \log(P(w_i|c))$
  - ex: $\log(0.01) + \log(0.02) + \log(0.05) + \ldots$

45

Some Adjustments

- product of probabilities… numerical underflow for long sentences

连乘会导致飞快下降

- so instead of multiplying the probs, we add the log of the probs

用log也不错

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

数据稀疏

如果训练语料库里没出现过，那么对于没见过的情况，都是0

- 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!

sequences

- Solution: smoothing

解决方法，加入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:

$\lambda = 0.5$

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

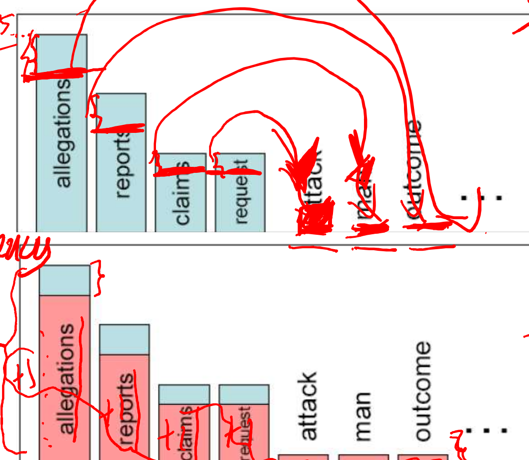
44

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



Before raw prob. distribution

After

frequency

+1

Add-one: Example

raw

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:

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

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

$P(w_2 \mid w_1)$

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

+1…+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

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

总共例子

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

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

+1 .5 .25

.5

  - most widely used value for δ = 0.5

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

Factors of Training Corpus

- Size: size越大越好，但是过一会儿后效果就停下来了
  - the more, the better
  - but after a while, not much improvement…
    - bigrams (characters) after 100’s million words
    - trigrams (characters) after some billions of words

A … Z V=26

- Genre (adaptation): 依赖原语料，你training了一个食谱，预测手工说明书就很憨
  - 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

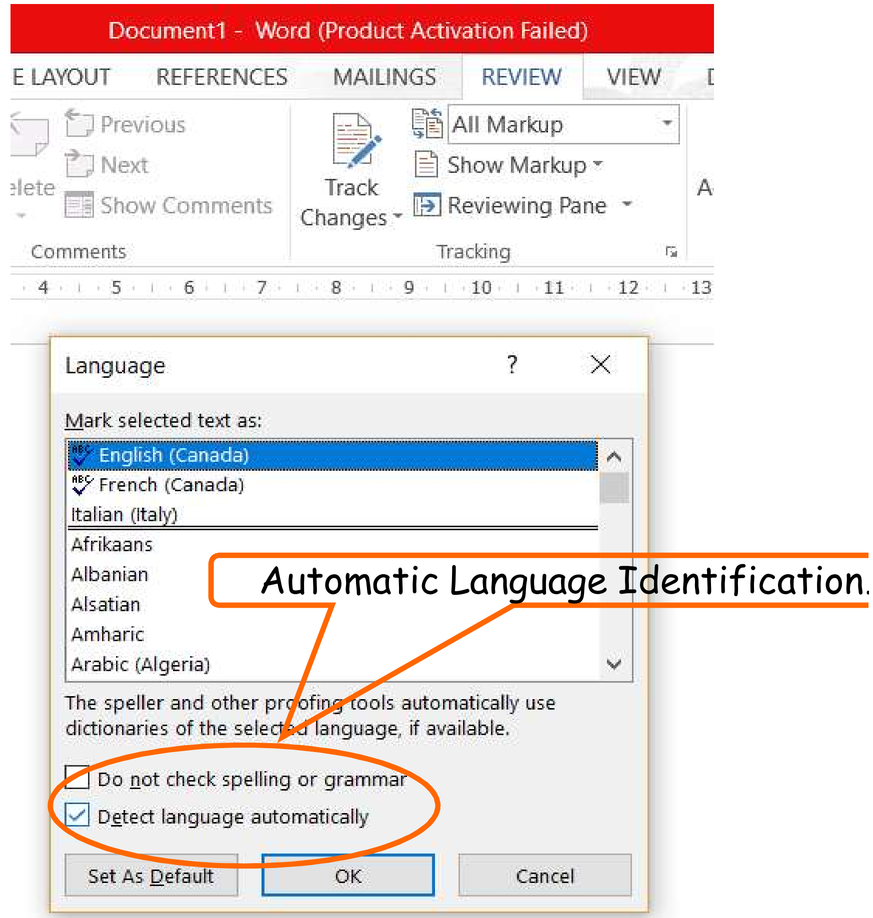
I am going to the d {dentist daycare}

Jack - 30 yr daddy

Example: Language Identification

识别是啥语言，
假设，如果是同样的语言，那么就有着同样的word sequence

- 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 例如英语里面ing用的更多
- Training phase:
  - construction of the language model 提前建立好model
  - with pre-classified documents (known language/author)
- Testing phase:
  - apply language model to unknown text 直接带入



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

126

bigram

italian

vs

Spanish

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

3. Given a unknown sentence "che bella cosa" is it in Italian or in Spanish?

P("che bella cosa") with the Italian LM $p(h|c) \times p(e|h) \dots \quad p(a|s) =$

P("che bella cosa") with the Spanish LM $p(h|c) \times p(e|h) \dots$

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

自然语言不是线性的

- Natural language is not linear ....
- there may be *long-distance dependencies*.

有可能依赖语句会变得很长

  - Syntactic dependencies
    - *The man next to the large oak tree near ... is tall.*
    - *The men next to the large oak tree near ... are tall.*
  - *Semantic dependencies*
    - *The bird next to the large oak tree near ... flies rapidly.*
    - *The man next to the large oak tree near ... talks rapidly.*
  - *World knowledge*
    - *Michael Jackson, who was featured in ..., is buried in California.*
    - *Michael Bublé, who was featured in ..., is living 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