UPES

UNIVERSITY WITH A PURPOSE

Computational Linguistic and Natural Language Processing

UNIVERSITY WITH A

Language Modeling (Introduction to N-grams)

- **Language modeling** (LM) is the use of various statistical and probabilistic techniques to determine the probability of a given sequence of words occurring in a sentence.

Uses and examples of language modeling

- Language models are the backbone of natural language processing (NLP). Below are some NLP tasks that use language modeling, what they mean, and some applications of those tasks
- Speech recognition
- Machine translation
- Parts-of-speech tagging
- Parsing
- Sentiment analysis
- Optical character recognition

Language Modeling (Introduction to N-grams)

- **Probabilistic Language Models**

Today's goal: assign a probability to a sentence.

- Machine Translation: • P(**high** winds tonight) > P(**large** winds tonight) (distinguish between good and bad translation).
- Spell Correction: The office is about fifteen **minuets** from my house

"P(about fifteen **minutes** from) > P(about fifteen **minuets** from)"

- Speech Recognition.. • P(I saw a van) >> P(eyes awe of an)
- + Summarization, question--answering, etc.!!

Language Modeling (Introduction to N-grams)

- **Probabilistic Language Models**
  - Goal: compute the probability of a sentence or sequence of words:

    $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$
  - Related task: probability of an upcoming word:

    $P(w_5|w_1,w_2,w_3,w_4)$
  - A model that computes either of these:

    $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.
  - Better: **the grammar** But **language model** or **LM** is standard

UP
UNIVERSITY WITH A

The Chain Rule

- **How to compute P(W)**
- How to compute this joint probability:
- P(its, water, is, so, transparent, that)
- Intuition: let's rely on the Chain Rule of Probability
- Recall the definition of conditional probabilities

Rewriting:

- More variables:

  $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$

- The Chain Rule in General

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

The Chain Rule

- **The Chain Rule applied to compute joint probability of words in sentence**

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

The Chain Rule

How to estimate these probabilities

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption

- Simplifying assumption:

P(the | its water is so transparent that) " P(the | that)

- Or maybe

P(the | its water is so transparent that) " P(the | transparent that)

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

N-gram Language Model

An N-gram is a sequence of N tokens (or words).

Let’s understand N-gram with an example. Consider the following sentence:

“I love reading blogs about data science.”

A 1-gram (or unigram) is a one-word sequence. For the above sentence, the unigrams would simply be: “I”, “love”, “reading”, “blogs”, “about”, “data”, “science”.

A 2-gram (or bigram) is a two-word sequence of words, like “I love”, “love reading”, “reading blogs”, “blogs about”, “about data”, “data science”. And a 3-gram (or trigram a three-word sequence of words like “I love reading”, “about data science”..

Simplest case: Unigram model

The simplest case of markov model is called as Uni-gram model. In the unigram model, we simply estimate the probability of the whole sequence of words by the product of the probability of the individual word.

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

Bigram model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november

N-gram models

- We can extend to trigrams, 4-grams, 5-grams.
- In general this is an insufficient model of language

  Because language has **long---distance dependencies**:

"The computer which I had just put into the machine room on the fifth floor crashed."

- But we can often get away with N-gram model

Estimating N-gram Probabilities

- **Estimating bigram probabili1es**
- The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Examples

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$$

More examples: Berkeley Restaurant Project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i’m looking for a good place to eat breakfast
- When is caffe venezia open during the day

Raw bigram counts

- Out of 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Raw bigram probabili1es

- Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Bigram estimates of sentence probabili1es

P(<s> I want English food </s>) =

P(I|<s>)

×P(want|I)

×P(english|want)

×P(food|english)

×P(</s>|food)

= .000031

What kinds of knowledge?

- P(English | want) = .0011
- P(Chinese | want) = .0065
- P(to | want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (I | <s>) = .25

Practical Issues

- We do everything in log space
- Avoid underflow
- (also adding is faster than multiplying)
- $\log(p1 * p2 * p3 * p4) = \log p1 + \log p2 + \log p3 + \log p4$

Language Modeling Toolkits

- SRILM
- http://www.speech.sri.com/projects/srilm/

Google N--Gram Release, August 2006 …

AUG
3

All Our N-gram are Belong to You

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,

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.

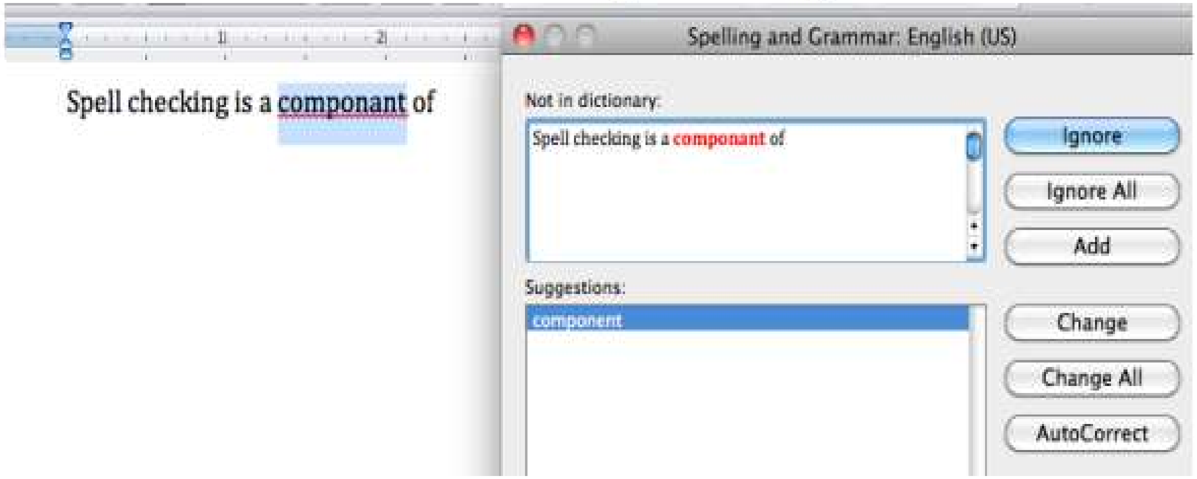
- `serve as the incoming 92`
- `serve as the incubator 99`
- `serve as the independent 794`
- `serve as the index 223`
- `serve as the indication 72`
- `serve as the indicator 120`
- `serve as the indicators 45`
- `serve as the indispensable 111`
- `serve as the indispensible 40`
- `serve as the individual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

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

Spelling Correction

Applications for spelling correction

Word processing



Phones

Web search

Spelling task

Spelling Error Detection

Spelling Error Correction:

- Autocorrect
- hte → the
- Suggest a correction
- Suggestion lists

Types of spelling errors

- Non-word Errors
  - graffe → giraffe
- Real word Errors
  - Typo graphical errors
    - three → there
  - Cognitive Errors (homophones)
    - piece → peace
    - too → two

Non word spelling errors

- Non-word spelling error detection:
    - Any word not in a ***dictionary*** is an error
    - The larger the dictionary the better
- Non-word spelling error correction:
    - Generate ***candidates***: real words that are similar to error
    - Choose the one which is best:
        - Shortest weighted edit distance
        - Highest noisy channel probability

Real-word spelling errors

For each word w, generate candidate set:

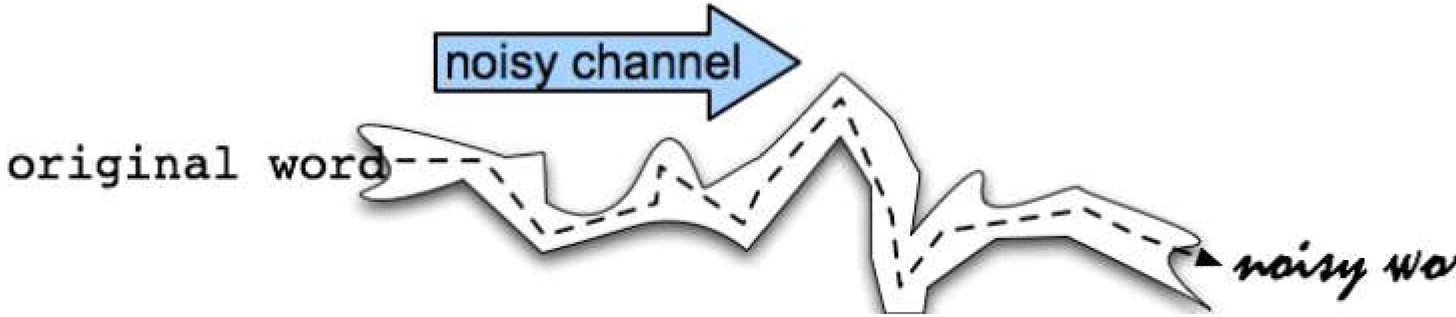
- Find candidate words with similar pronunciations
- Find candidate words with similar spelling
- Include w in candidate set

Choose best candidate

- Noisy Channel
- Classifier

The Spelling Correction Task (The Noisy Channel Model of Spelling)

Noisy Channel Intuition:



Noisy Channel: It is a probabilistic model.

- We see an observation x of a misspelled word
- Find the correct word w

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w \mid x)$$

$$= \underset{w \in V}{\operatorname{argmax}} \frac{P(x \mid w)P(w)}{P(x)}$$

$$= \underset{w \in V}{\operatorname{argmax}} P(x \mid w)P(w)$$

Non word spelling error example

acress

Candidate generation

- Words with similar spelling
    - Small edit distance to error
- Words with similar pronunciation
    - Small edit distance of pronunciation to error
- Minimal edit distance between two strings, where edits are:
    - Insertion
    - Deletion
    - Substitution
    - Transposition of two adjacent letters

Words within 1 of acress

Error	Candidate Correction	Correct Letter	Error Letter	Type
acress	actress	t	–	deletion
acress	cress	–	a	insertion
acress	caress	ca	ac	transposition
acress	access	c	r	substitution
acress	across	o	e	substitution
acress	acres	–	s	insertion
acress	acres	–	s	insertion

Candidate generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2

- Also allow insertion of **space** or **hyphen**
  - `thisidea` → `this idea`
  - `inlaw` → `in-law`

Language Model

- Use any of the language modeling algorithms we've learned
- Unigram, bigram, trigram
- Web-scale spelling correction
  - Stupid backoff

Unigram Prior probability

Counts from 404,253,213 words in Corpus of Contemporary English (COCA)

word	Frequency of word	P(word)
actress	9,321	.0000230573
cress	220	.0000005442
caress	686	.0000016969
access	37,038	.0000916207
across	120,844	.0002989314
acres	12,874	.0000318463

Channel model probability

- **Error model probability, Edit probability**
- *Kernighan, Church, Gale 1990*

- *Misspelled word* $x = x_1, x_2, x_3 \ldots x_m$
- *Correct word* $w = w_1, w_2, w_3, \ldots, w_n$

- $P(x|w)$ = probability of the edit
  - (deletion/insertion/substitution/transposition)

Computing error probability: confusion matrix

```
del[x,y]:    count(xy typed as x)
ins[x,y]:    count(x typed as xy)
sub[x,y]:    count(x typed as y)
trans[x,y]:  count(xy typed as yx)
```

Insertion and deletion conditioned on previous character

Confusion matrix for spelling errors

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X													Y (correct)													
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Channel model

$$P(x|w) = \begin{cases} \dfrac{\mathrm{del}[w_{i-1}, w_i]}{\mathrm{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \dfrac{\mathrm{ins}[w_{i-1}, x_i]}{\mathrm{count}[w_{i-1}]}, & \text{if insertion} \\ \dfrac{\mathrm{sub}[x_i, w_i]}{\mathrm{count}[w_i]}, & \text{if substitution} \\ \dfrac{\mathrm{trans}[w_i, w_{i+1}]}{\mathrm{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$