CS4442/9542b
Artificial Intelligence II
prof. Olga Veksler

Lecture 10

Natural Language Processing

Spelling Correction

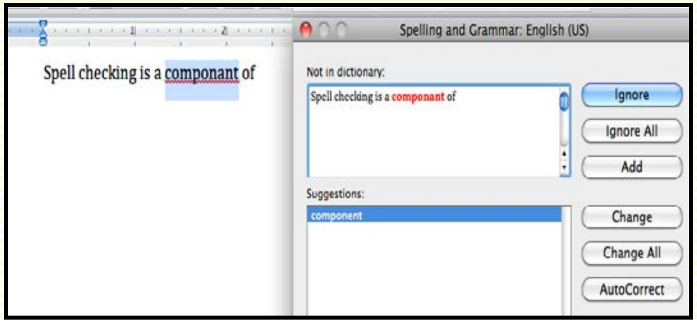
Many slides from: D. Jurafsky, C. Manning

Outline

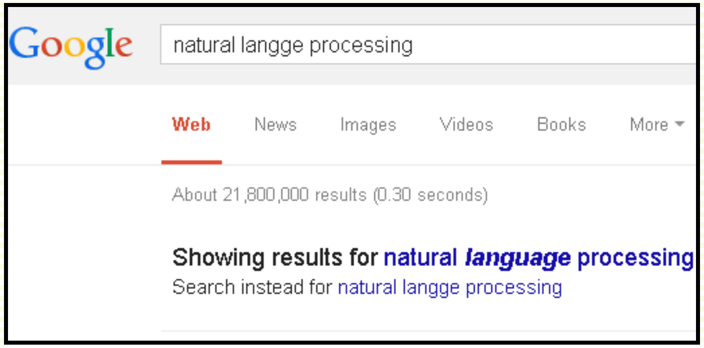
- Intro to spelling correction
- Types of spelling errors
  1. non word spelling errors
  2. real world spelling errors
- Spelling tasks:
  1. Detecting errors
     1. non word spelling errors: dictionary + context
     2. real word spelling errors: from context
  2. Correcting Errors
     - edit distance
       - Dynamic programming (DP) for computing minimum edit distance
     - noisy channel model

Applications for Spelling Correction

- Word processing



- Web search



- Mobile devices, etc.

Types of Spelling Errors

1. **Non-word errors** (not in dictionary)
    - *graffe* → *giraffe*
2. **Real-word errors** (in dictionary)
    - typographical errors
        - *three* → *there*
    - cognitive errors (homophones)
        - *piece*→*peace*
        - *too* → *two*

Non-Word Spelling Errors Detection/Correction

- Non-word spelling error detection
  - any word not in ***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

Candidate Generation

- Words with similar spelling
- The user typed graffe
- Which is closest
  - graf, graft, grail, giraffe?
- Use **edit distance** to compute **d**(graffe,graf)

Damerau-Levenshtein Edit Distance

- The **minimum edit distance** between two strings is the minimum number of editing operations needed to transform one string into the other
- Editing operations:
  - insertion
  - deletion
  - substitution
  - for spelling, also want to include transposition of two letters

Minimum Edit Distance

- Two strings and their **alignment**:

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```

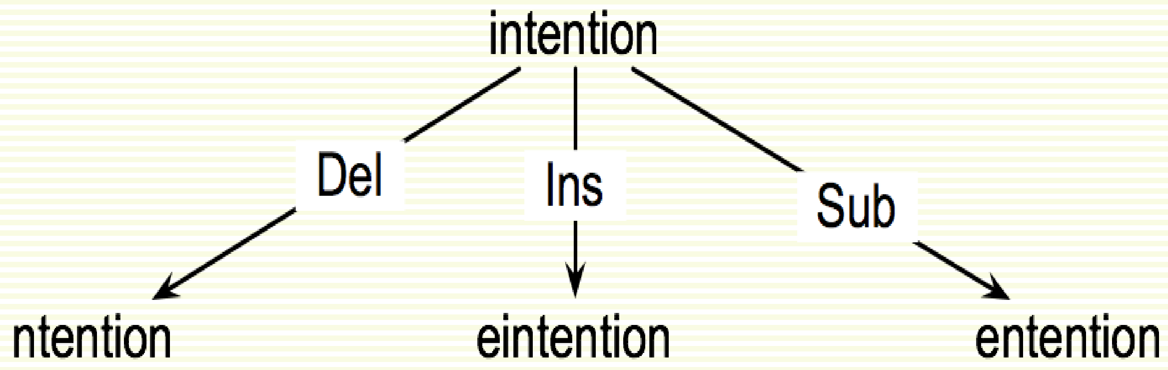
Minimum Edit Distance

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

- If each operation has cost of 1
  - distance between these is 5
- If substitutions cost 2 (Levenshtein)
  - distance between them is 8

How to Find Min Edit Distance?

- Naïve approach
- Searching for a path (sequence of edits) from the start string to the final string:
  - **Initial state**: the word we're transforming
  - **Operators**: insert, delete, substitute
  - **Goal state**: the word we're trying to get to
  - **Path cost**: what we want to minimize: the number of edits



Minimum Edit as Search

- But the space of all edit sequences is huge!
  - we can’t afford to navigate naïvely
  - many distinct paths wind up at the same state
  - **dynamic programming** for efficiency

Defining Min Edit Distance

- For two strings
  - **X** of length **n**
  - **Y** of length **m**
- Define **`D(i,j)`**
  - the edit distance between **X**[**1**...**i**] and **Y**[**1**...**j**]
    - i.e., the first **i** characters of **X** and the first **j** characters of **Y**
  - the edit distance between **X** and **Y** is thus **`D(n,m)`**

Dynamic Programming for Min Edit Distance

- Tabular computation of **`D(n,m)`**
- Solve problems by combining solutions to subproblems
- Bottom-up
  - Initialize: compute **`D(i,j)`** for small **`i`**, **`j`**
  - Iterate: compute larger **`D(i,j)`** based on previously computed smaller values

Defining Min Edit Distance (Levenshtein)

- Initialization

```
D(i,0) = i
D(0,j) = j
```

D(3,0) = 3

intention

↓

ø execution

D(0,5) = 5

ø intention

↑

execution

Defining Min Edit Distance (Levenshtein)

- Initialization

```
D(i,0) = i
D(0,j) = j
```

- Recurrence Relation (iteration)

```
for each  i = 1...m
     for each  j = 1...n
```

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2 \text{ if } X(i) \neq Y(j) \\ 0 \text{ if } X(i) = Y(j) \end{cases} \end{cases}$$

Defining Min Edit Distance (Levenshtein)

- The smallest of:

```
D(int,exec)= del[t] + D(in,exec)
D(int,exec)= D(int,exe) + ins[c]
D(int,exec)= substitute[t,c] + D(in,exe)
```

- Recurrence Relation (iteration)

```
for each  i = 1...m
      for each  j = 1...n
```

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2 \text{ if } X(i) \neq Y(j) \\ 0 \text{ if } X(i) = Y(j) \end{cases} \end{cases}$$

Defining Min Edit Distance (Levenshtein)

- Initialization

$$D(i,0) = i$$
$$D(0,j) = j$$

- Recurrence Relation (iteration)

```
for each  i = 1...m
     for each  j = 1...n
```

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2 \text{ if } X(i) \neq Y(j) \\ 0 \text{ if } X(i) = Y(j) \end{cases} \end{cases}$$

- Termination

$D(n,m)$ is distance

Edit Distance Table: Initialization

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Edit Distance Table: Iterations

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2 \text{ if } X(i) \neq Y(j) \\ 0 \text{ if } X(i) = Y(j) \end{cases} \end{cases}$$

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Edit Distance Table: Iterations

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2 \text{ if } X(i) \neq Y(j) \\ 0 \text{ if } X(i) = Y(j) \end{cases} \end{cases}$$

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1	2								
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Edit Distance Table: Iterations

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2 \text{ if } X(i) \neq Y(j) \\ 0 \text{ if } X(i) = Y(j) \end{cases} \end{cases}$$

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1	2								
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Edit Distance Table: Iterations

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1	2	**3**							
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Edit Distance Table: Termination

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2 \text{ if } X(i) \neq Y(j) \\ 0 \text{ if } X(i) = Y(j) \end{cases} \end{cases}$$

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Edit Distance Table: Termination

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2	**3**								
I	1	2	3	4	5	6	7	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Edit Distance Table: Termination

N	9	8	9	10	11	12	11	10	9	**8**
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

MinEdit with Backtrace

- Keep pointer to extract the optimal alignment

n	9	↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙←↓ 12	↓ 11	↓ 10	↓ 9	↙ **8**
o	8	↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↓ 10	↓ 9	↙ **8**	← 9
i	7	↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	↙ **8**	← 9	← 10
t	6	↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙ **8**	← 9	← 10	←↓ 11
n	5	↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ **8**	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙↓ 10
e	4	↙ 3	← 4	↙← **5**	← **6**	← 7	←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9
t	3	↙←↓ 4	↙←↓ **5**	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙ 7	←↓ 8	↙←↓ 9	↓ 8
n	2	↙←↓ **3**	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↓ 7	↙←↓ 8	↙ 7
i	**1**	↙←↓ 2	↙←↓ 3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙ 6	← 7	← 8
#	**0**	1	2	3	4	5	6	7	8	9
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**

Performance

- Time: O(nm)
- Space: O(nm)

Weighted Edit Distance

- Why add weights to computation?
- Some letters are more likely to be mistyped than others
- Collect statistics on how often
  - one letter is substituted with another
  - on deletions
  - on insertions
  - on transpositions

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

Confusion Matrix for Spelling Errors

sub[X, Y] = Sub

X	a	b	c	d	e	f	g	h	i
a	0	0	7	1	342	0	0	2	118
b	0	0	9	9	2	2	3	1	0
c	6	5	0	16	0	9	5	0	0
d	1	10	13	0	12	0	5	5	0
e	388	0	3	11	0	2	2	0	89
f	0	15	0	3	1	0	5	2	0
g	4	1	11	11	9	2	0	0	0
h	1	8	0	3	0	0	0	0	0

v	w	x	y	z
0	1	0	5	0
0	8	0	0	0
3	7	1	1	0
0	4	0	2	0
0	1	0	18	0
0	2	0	0	0
0	1	0	3	0
0	2	0	0	0
0	2	1	15	0
0	0	0	0	0
0	4	0	0	3
0	0	0	0	0
3	2	2	3	0
0	1	2	0	2
0	0	0	18	0
4	1	0	0	0
0	0	0	0	0
0	0	1	0	0
0	5	3	20	1
2	19	0	7	6
0	2	0	8	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

Weighted Min Edit Distance

- Initialization

$$D(0,0) = 0$$

$$D(i,0) = D(i-1,0) + del[x(i)] \qquad 1 < i \leq n$$

$$D(0,j) = D(0,j-1) + ins[y(j)] \qquad 1 < j \leq m$$

- Recurrence Relation

$$D(i,j) = \min \begin{cases} D(i-1,j) & + \; del[x(i)] \\ D(i,j-1) & + \; ins[y(j)] \\ D(i-1,j-1) & + \; sub[x(i),y(j)] \end{cases}$$

- Termination

$D(n,m)$ is distance

Noisy Channel Model: Intuition

send **w** $\xrightarrow{\text{noisy channel}}$ receive **x**

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

prob of **w** given misspelled **x**

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

best guess

search over vocabulary **V**

How to Model **P**(**w**|**x**)?

- **P**(**w**|**x**) is not easy to model

send **w** $\xrightarrow{\text{noisy channel}}$ receive **x**

- But **P**(**x**|**w**) is easier to model
- Can study the properties of noisy channel
  - how often one intends to write **w** but writes **x** instead?
  - example: how often want to write “the” but write “hta”

Noisy Channel

- Since **P**(**x**|**w**) is easier to model, rewrite:

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

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

$$= \underset{\mathbf{w} \in \mathbf{V}}{\operatorname{argmax}} \underbrace{\mathbf{P}(\mathbf{x} \mid \mathbf{w})}_{\text{noisy channel model}} \underbrace{\mathbf{P}(\mathbf{w})}_{\text{language model}}$$

How to Model **P(w)**?

- Use one of the language models we have learned
- Unigram probabilities if **w** is just a word
- If modeling error in context, use bigram or trigram model
  - Sentence: Like ther books
  - Non-word = ther
  - Both there and their have edit distance of 1 to ther
  - Both **P**(their) and **P**(there), have high probability
  - Add context: **P**(like there books) vs. **P** (like their books)
- With smoothing if necessary
  - Add-Delta, Good-Turing, etc

Modeling Noisy Channel **P(x|w)**

- Also called error probability model
- Build model from **marked** training data, much like for modeling language
  - need texts with marked mistakes/corrections
- Assume one mistake per word for simplicity
  - Misspelled word **x** has characters $\mathbf{x}_1\ \mathbf{x}_2 \ldots \mathbf{x}_m$
  - Dictionary word **w** has characters $\mathbf{w}_1\ \mathbf{w}_2 \ldots \mathbf{w}_n$
  - Can get from **w** to **x** with either one insertion, or one deletion, etc
- **P**(**x**|**w**) is probability of this one mistake

Modeling Noisy Channel **P(x|w)**

- Get counts from marked data
- Here **`y`** and **`z`** are characters

```
del[z,y]:    count(zy typed as z)
ins[z,y]:    count(z typed as zy)
sub[z,y]:    count(z typed as y)
trans[z,y]:  count(zy typed as yz)
```

- Generate confusion matrix from marked data
  - Peter Norvig, etc.

Confusion Matrix for Modeling **P(x|w)**

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

Modeling Noisy Channel **P(x|w)**

- Assume can get from **x** to **w** with one error

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

Non-Word Spelling Error Example

“acress”

Non-Word Spelling Error Example

"acress"

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2
- Can also allow insertion of **space** or **hyphen**

```
thisidea → this idea
inlaw    → in-law
```

Words within distance 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

Unigram Probability P(**w**)

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

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

Noisy Channel Model P(**x**|**w**) for acress

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)
actress	t	-	c|ct	**.000117**
cress	-	a	a|#	.00000144
caress	ca	ac	ac|ca	.00000164
access	c	r	r|c	.000000209
across	o	e	e|o	.0000093
acres	-	s	es|e	.0000321
acres	-	s	ss|s	.0000342

Combining P(w) and P(w|x) for acress

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)	P(word)	10^9 *P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	0.00078
caress	ca	ac	ac|ca	.00000164	.00000170	0.0028
access	c	r	r|c	.000000209	.0000916	0.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

Adding Context

"a stellar and versatile **acress** whose combination of sass and glamour"

- Adding context to misspelled word really helps
- Use bigram language model
- Counts from CCAE with add-1 smoothing

P(actress|versatile) = .000021

P(whose|actress) = .0010

P(versatile actress whose) = .000021*.0010 = 210 x10^{-10}

P(across|versatile) = .000021

P(whose|across) = .000006

P(versatile across whose) = .000021*.000006 = 1 x10^{-10}

Using Bigram Language Model

- Now multiply by noisy channel probabilities

P(acress | actress) = .000117

P(acress | across) = .0000093

- To finally get

P(versatile actress whose)P(acress|actress) = $.000117 \times 210 \times 10^{-10}$

$= .002457 \times 10^{-10}$

P(versatile across whose)P(acress|across) = $.0000093 \times 1 \times 10^{-10}$

$= .0000093 \times 10^{-10}$

Real-Word Spelling Errors

```
leaving in about fifteen minuets to go to her house.
The design an construction of the system.
Can they lave him my messages?
The study was conducted mainly be John Black.
```

- Spelling errors that are words in dictionary
- 25-40% of spelling errors are real words
  - from [Kukich 1992]
- Use context of in this case

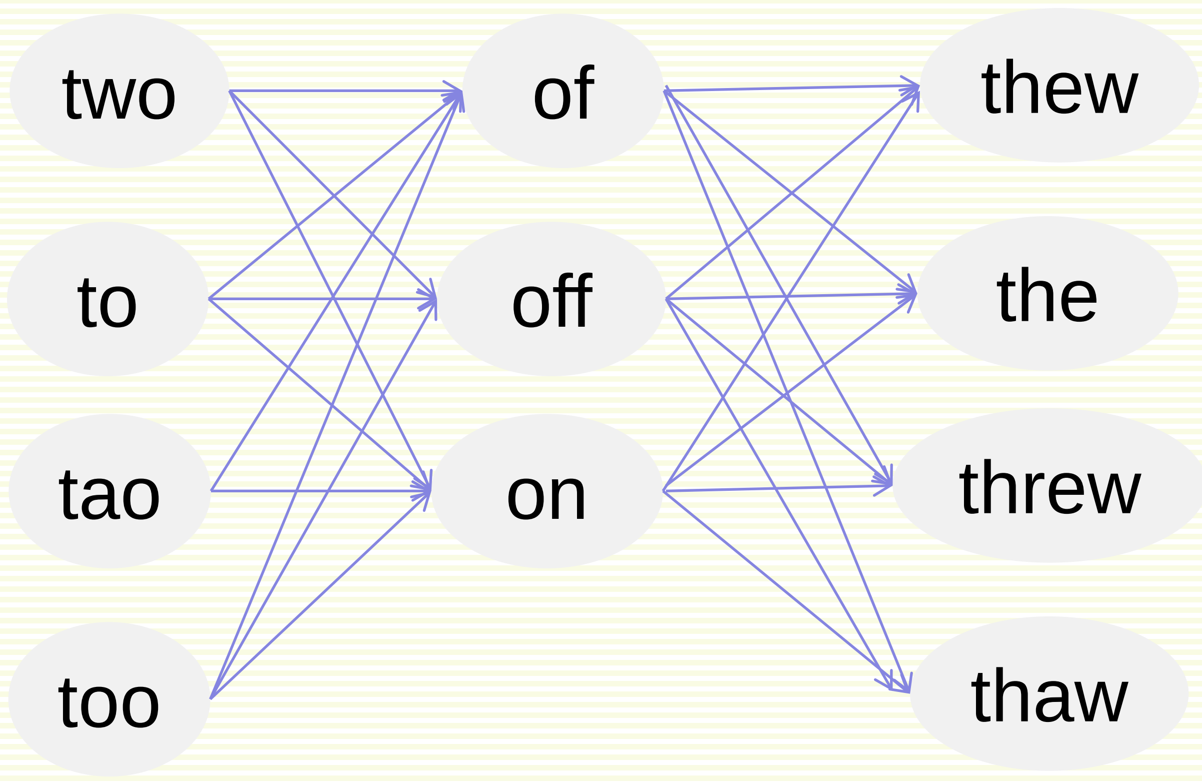
Solving Real-World Spelling Errors

- For each word **w** in the sentence
  - Generate *candidate set*
    - the word itself
    - all single-letter edits that are English words
    - homophone words (similar pronunciations)
- Choose best combination word candidates
  - noisy channel model
  - task-specific classifier

Example of Real-Word Spell Correction [P.Norvig]

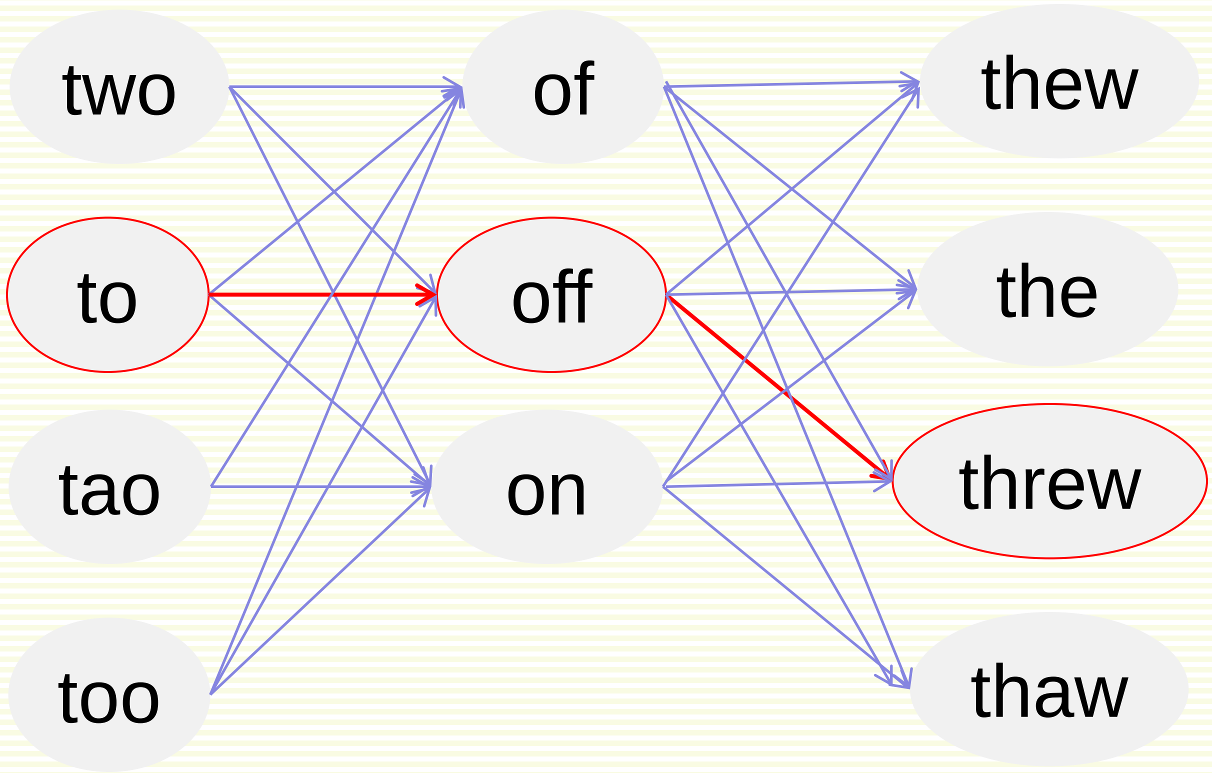
- Example: "two of thew"
  - Candidate (two) = {two, to, tao, too}
  - Candidate (of) = {of, off, on}
  - Candidate (thew) = {thew, the, threw, thaw}

Graph of all Possible Candidate Combinations



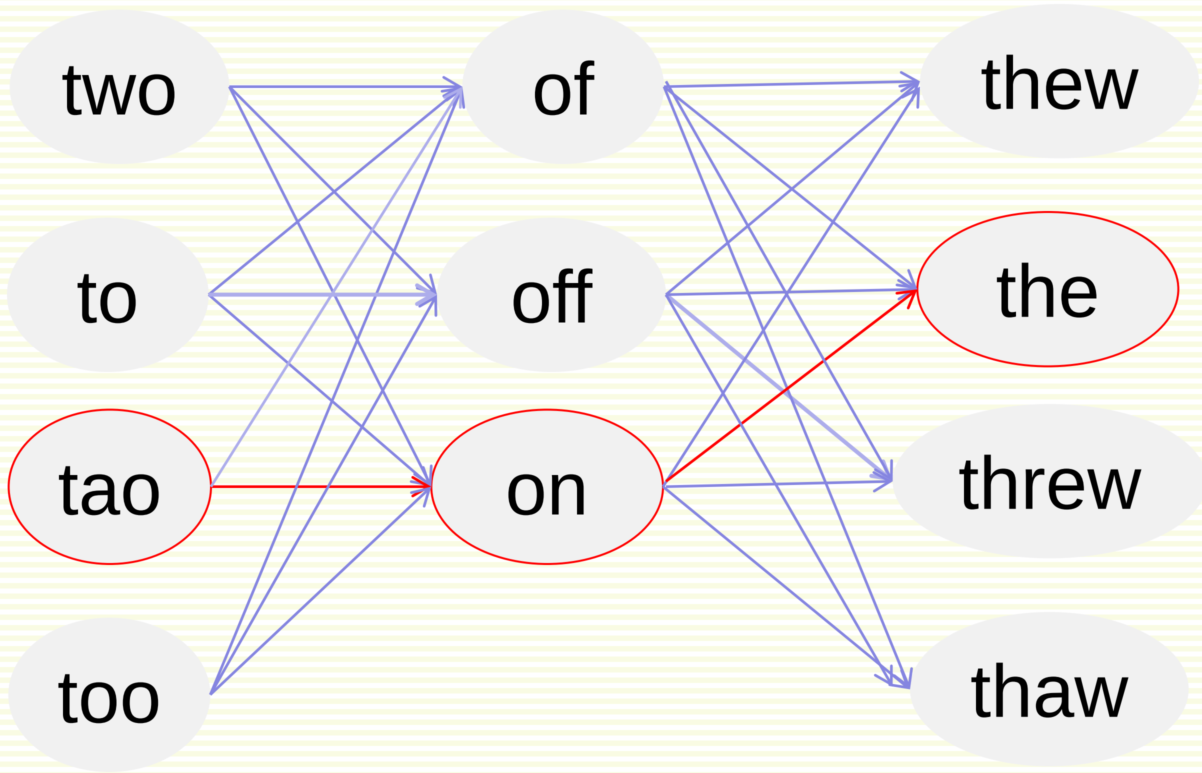
- Each path corresponds to a possible sentence

Graph of all Possible Candidate Combinations



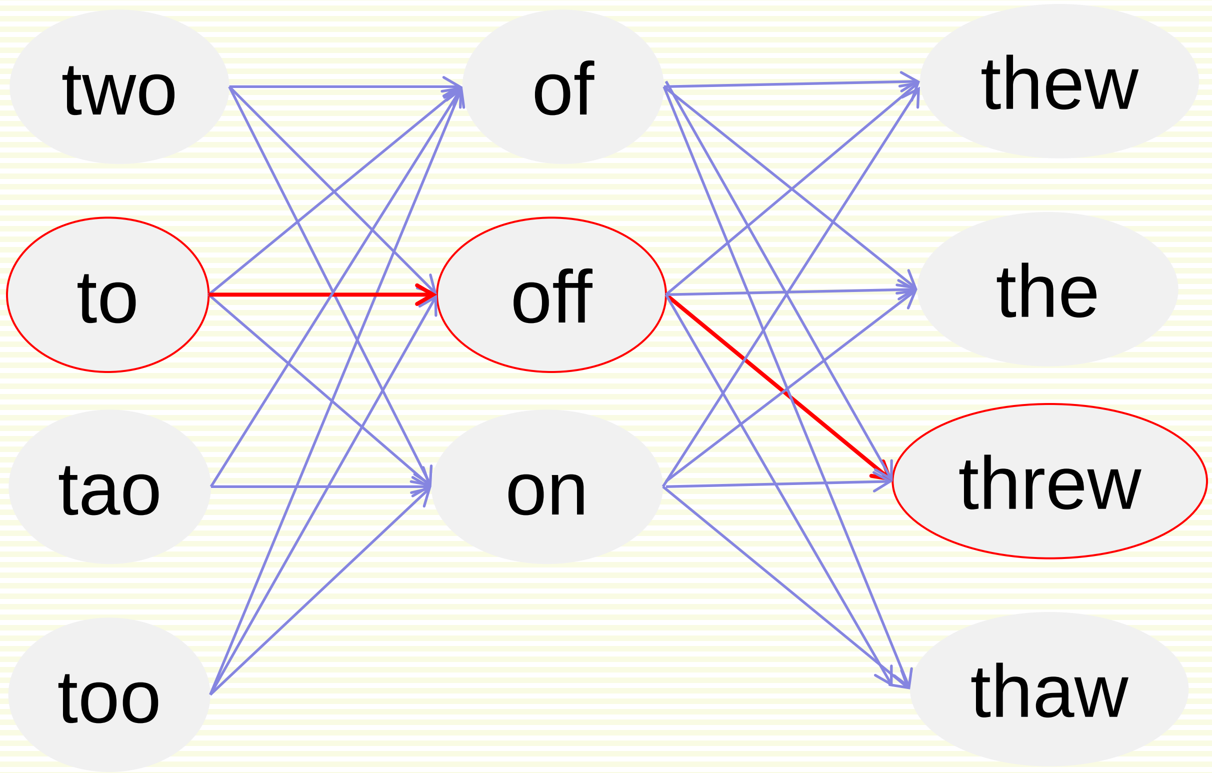
- One possible path (sentence)

Graph of all Possible Candidate Combinations



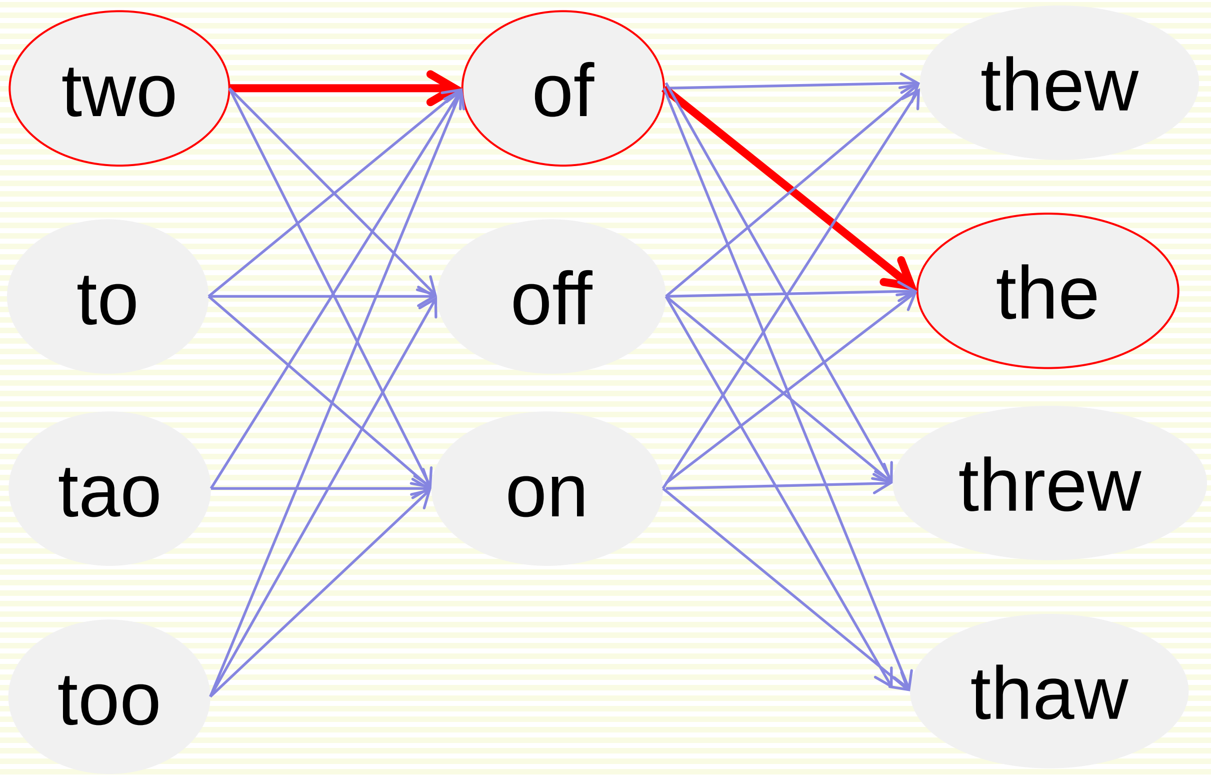
- Another possible path (sentence)

Graph of all Possible Candidate Combinations



- Compute probability of each path (sentence)
- Choose the path (sentence) of highest probability

Noisy Channel for Real-Word Spell Correction



- The best probability path (sentence), hopefully
- Can use DP (dynamic programming) to speed up computation

Simplification: One Error per Sentence

- Or simplify to allow only one word to be replaced
  - assumes one error per sentence
- Example: “two of thew”
  - C(two) = {two, to, tao, too}
  - C(of) = {of, off, on}
  - C (thew) = {thew, the, threw, thaw}
- Check only 11 sentences:

Two of threw	Two of thew	Two of thew
To of thew	Two off thew	Two of the
Tao of thew	Two on thew	Two of threw
Too of thew		Two of thaw

Where to Get Probabilities

- Language model **P(w)**
  - Same as before
    - Bigram or trigram
      - should not use unigram as it gives no context
- Noisy Channel Model **P(w|x)**
  - same as for non-word spelling correction
  - plus need probability for no error, **P(w|w)**
- Choose sentence that maximizes **P(w|x)P(w)**

Probability of No Error

- Noisy channel probability for correctly typed word?
  - **P**(the|the)
  - i.e. probability of no error
- Depends on the application
  - .90 (1 error in 10 words)
    - *for Olga*
  - .95 (1 error in 20 words)
  - .99 (1 error in 100 words)
  - .995 (1 error in 200 words)
    - *for English literature major*

Peter Norvig's Example Continued

x	w	x|w	P(x|w)	P(w)	10^9 P(x|w)P(w)
thew	the	ew|e	0.000007	0.02	144
thew	thew		0.95	0.00000009	90
thew	thaw	e|a	0.001	0.0000007	0.7
thew	threw	h|hr	0.000008	0.000004	0.03
thew	thwe	ew|we	0.000003	0.00000004	0.0001

HCI issues in spelling

- Autocorrect, f very confident in correction
  - Very common mistake: hte→the
- Less confident
  - Give one best correction
- Less confident
  - Give a correction list
- Unconfident
  - Just flag as an error

State of the art noisy channel

- Never just multiply the prior and the error model
- Independence assumptions→probabilities not commensurate
- Instead: weigh them

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

- Learn λ from a validation set

Classifier for Real-Word Spelling Correction

- Instead of just channel model and language model
- Use many features in a classifier
- Build a classifier for a specific pair like:

  whether/weather

  - “cloudy” within +- 10 words
  - ___ to VERB
  - ___ or not

Evaluation

- Some spelling error test sets
  - Wikipedia's list of common English misspelling
  - Aspell filtered version of that list
  - Birkbeck spelling error corpus
  - Peter Norvig's list of errors (includes Wikipedia and Birkbeck, for training or testing)