Spelling Correction: Edit Distance

Niloy Ganguly

CSE, IITKGP

Week 2

Spelling Correction

Spelling Correction

I am writing this email on behaf of ...

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
- ....

Spelling Correction

I am writing this email on behaf of ...
The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
- ....

Isolated word error correction

- Pick the one that is closest to ‘behaf’

Spelling Correction

I am writing this email on behaf of ...
The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
- ....

Isolated word error correction

- Pick the one that is closest to ‘behaf’
- How to define ‘closest’?

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
- ....

Isolated word error correction

- Pick the one that is closest to ‘behaf’
- How to define ‘closest’?
- Need a **distance metric**

Spelling Correction

I am writing this email on behaf of ...
The user typed 'behaf'.

Which are some close words?

- behalf
- behave
- ....

Isolated word error correction

- Pick the one that is closest to 'behaf'
- How to define 'closest'?
- Need a **distance metric**
- The simplest metric: **edit distance**

Edit Distance

- The minimum edit distance between two strings

Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations

Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
  - Insertion
  - Deletion
  - Substitution

Minimum Edit Distance

Example

Edit distance from ‘intention’ to ‘execution’

Minimum Edit Distance

Example

Edit distance from ‘intention’ to ‘execution’

```
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
```

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 a cost of 1 (Levenshtein)
    - Distance between these is 5

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 a cost of 1 (Levenshtein)
  - Distance between these is 5
- If substitution costs 2 (alternate version)
  - Distance between these is 8

How to find the Minimum Edit Distance?

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to

How to find the Minimum Edit Distance?

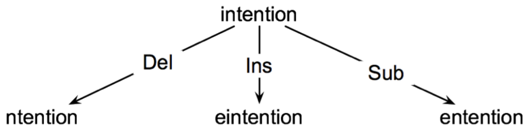
Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to
- **Path cost:** what we want to minimize: the number of edits

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to
- **Path cost:** what we want to minimize: the number of edits



Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge

Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state

Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state
- Don’t have to keep track of all of them

Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state
- Don't have to keep track of all of them
- Keep track of the shortest path to each state

Defining Minimum Edit Distance Matrix

For two strings

- X of length n
- Y of length m

Defining Minimum Edit Distance Matrix

For two strings

- X of length n
- Y of length m

We 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

Defining Minimum Edit Distance Matrix

For two strings

- X of length n
- Y of length m

We 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

Thus, the edit distance between X and Y is $D(n,m)$

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n, m)$

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems
- Bottom-up

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems
- Bottom-up
  - Compute $D(i,j)$ for small i,j

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems
- Bottom-up
  - Compute $D(i,j)$ for small i,j
  - Compute larger $D(i,j)$ based on previously computed smaller values

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems
- Bottom-up
  - Compute $D(i,j)$ for small i,j
  - Compute larger $D(i,j)$ based on previously computed smaller values
  - Compute $D(i,j)$ for all i and j till you get to $D(n,m)$

Dynamic Programming Algorithm

Initialization

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

Recurrence Relation:

```
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
```

The Edit Distance Table

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

The Edit Distance Table

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

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

The Edit Distance Table

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

Computing Alignments

Computing Alignments

- Computing edit distance may not be sufficient for some applications

Computing Alignments

- Computing edit distance may not be sufficient for some applications
    - We often need to align characters of the two strings to each other

Computing Alignments

- Computing edit distance may not be sufficient for some applications
  - We often need to align characters of the two strings to each other
- We do this by keeping a “backtrace”

Computing Alignments

- Computing edit distance may not be sufficient for some applications
  - We often need to align characters of the two strings to each other
- We do this by keeping a “backtrace”
- Every time we enter a cell, remember where we came from

Computing Alignments

- Computing edit distance may not be sufficient for some applications
  - We often need to align characters of the two strings to each other
- We do this by keeping a “backtrace”
- Every time we enter a cell, remember where we came from
- When we reach the end,

Computing Alignments

- Computing edit distance may not be sufficient for some applications
  - We often need to align characters of the two strings to each other
- We do this by keeping a “backtrace”
- Every time we enter a cell, remember where we came from
- When we reach the end,
  - Trace back the path from the upper right corner to read off the alignment

The Edit Distance Table

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

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

The Edit Distance Table

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

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

Minimum Edit with Backtrace

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**	

Adding Backtrace to Minimum Edit

Base conditions:

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

Termination:

`D(N,M) is distance`

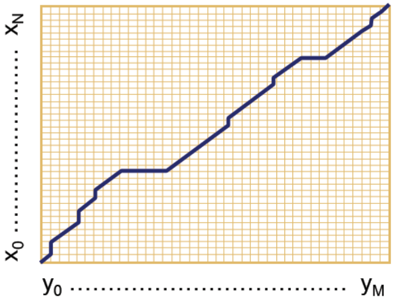
Recurrence Relation:

```
For each  i = 1…M
     For each  j = 1…N
```

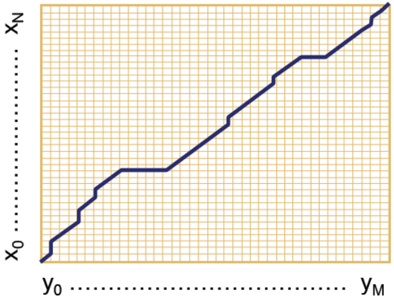
$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{deletion} \\ D(i,j-1) + 1 & \text{insertion} \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases} & \text{substitution} \end{cases}$$

$$ptr(i,j) = \begin{cases} \text{LEFT} & \text{insertion} \\ \text{DOWN} & \text{deletion} \\ \text{DIAG} & \text{substitution} \end{cases}$$

The distance matrix

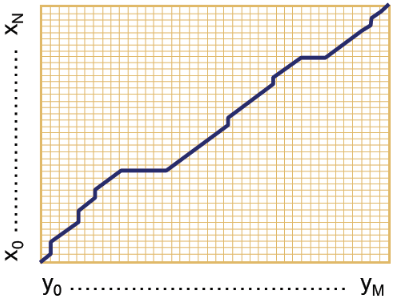


The distance matrix



Every non-decreasing path from (0,0) to (M,N) corresponds to an alignment of two sequences.

The distance matrix



Every non-decreasing path from (0,0) to (M,N) corresponds to an alignment of two sequences.

An optimal alignment is composed of optimal sub-alignments.

Result of Backtrace

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

Performance

Time

Performance

Time

$O(nm)$

Space

Performance

Time

$O(nm)$

Space

$O(nm)$

Backtrace

Performance

Time

$O(nm)$

Space

$O(nm)$

Backtrace

$O(n+m)$

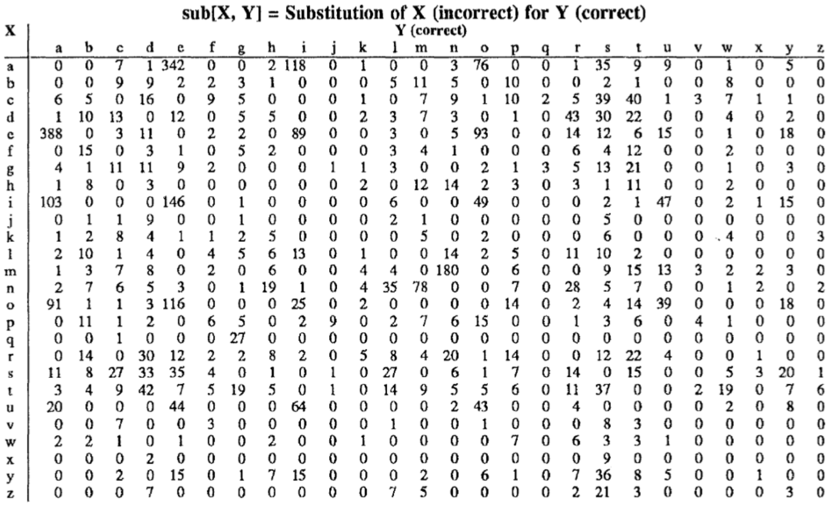
Weighted Edit Distance, Other variations

Weighted Edit Distance

Why to add weights to the computation?

- Some letters are more likely to be mistyped.

Confusion Matrix for Spelling Errors

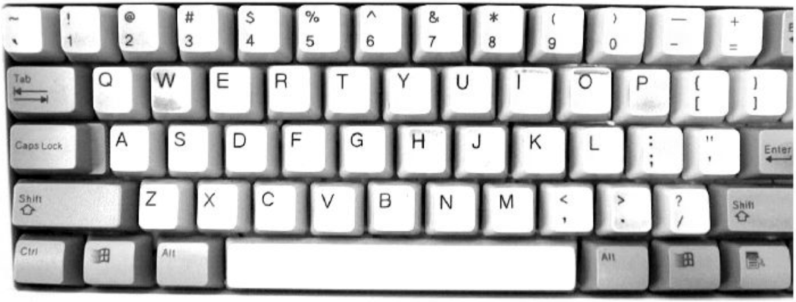


sub[X, Y] = Substitution of X (incorrect) for Y (correct)

Y (correct)

X	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

Keyboard Design



Weighted Minimum Edit Distance

Initialization:

```
D(0,0) = 0
D(i,0) = D(i-1,0) + del[x(i)];     1 < i ≤ N
D(0,j) = D(0,j-1) + ins[y(j)];     1 < j ≤ M
```

Recurrence Relation:

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

Termination:

```
D(N,M) is distance
```

How to modify the algorithm with transpose?

Transpose

- $transpose(x, y) = (y, x)$
- Also known as metathesis

How to modify the algorithm with transpose?

Transpose

- $transpose(x,y) = (y,x)$
- Also known as metathesis

Modification to the dynamic programmic algorithm

$$D[i][j] = min \begin{cases} D(i-1,j)+1 & (deletion) \\ D(i,j-1)+1 & (insertion) \\ D(i-1,j-1)+ & \begin{cases} 2 & if(x[i] \neq y[j])(substitution) \\ 0 & otherwise \end{cases} \\ D(i-2,j-2)+1 & (x[i] = y[j-1] \text{ and } x[i-1] = y[j] \\ & (transposition) \end{cases}$$

How to find dictionary entries with smallest edit distance?

How to find dictionary entries with smallest edit distance?

Naïve Method

Compute edit ditance from the query term to each dictionary term – an exhaustive search

How to find dictionary entries with smallest edit distance?

Naïve Method

Compute edit ditance from the query term to each dictionary term – an exhaustive search

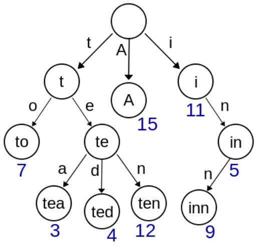
Can be made efficient if we do it over a trie structure

How to find dictionary entries with smallest edit distance?

Naïve Method

Compute edit ditance from the query term to each dictionary term – an exhaustive search

Can be made efficient if we do it over a trie structure



How to find dictionary entries with smallest edit distance?

How to find dictionary entries with smallest edit distance?

- Generate all possible terms with an edit distance <=2 (deletion + transpose + substitution + insertion) from the query term and search them in the dictionary.

How to find dictionary entries with smallest edit distance?

- Generate all possible terms with an edit distance <=2 (deletion + transpose + substitution + insertion) from the query term and search them in the dictionary.
- For a word of length 9, alphabet of size 36, this will lead to 114,324 terms to search for

How to find dictionary entries with smallest edit distance?

- Generate all possible terms with an edit distance <=2 (deletion + transpose + substitution + insertion) from the query term and search them in the dictionary.
- For a word of length 9, alphabet of size 36, this will lead to 114,324 terms to search for
- For Chinese alphabet size is 70,000 (Unicode Han Characters)

How to find dictionary entries with smallest edit distance?

Symmetric Delete Spelling Correction

- Generate terms with an edit distance ≤ 2 (deletes) from each dictionary term (offline)
- Generate terms with an edit distance ≤ 2 (deletes) from the input terms and search in dictionary

How to find dictionary entries with smallest edit distance?

Symmetric Delete Spelling Correction

- Generate terms with an edit distance ≤ 2 (deletes) from each dictionary term (offline)
- Generate terms with an edit distance ≤ 2 (deletes) from the input terms and search in dictionary

Number of deletes within edit distance ≤ 2 for a word of length 9 will be 45

How to find dictionary entries with smallest edit distance?

Symmetric Delete Spelling Correction

- Generate terms with an edit distance ≤ 2 (deletes) from each dictionary term (offline)
- Generate terms with an edit distance ≤ 2 (deletes) from the input terms and search in dictionary

Number of deletes within edit distance ≤ 2 for a word of length 9 will be 45

A further check is required to remove the false positives

Spelling Correction

Spelling Correction

Types of spelling errors: Non-word Errors

- behaf $\rightarrow$ behalf

Spelling Correction

Types of spelling errors: Non-word Errors

- behaf $\rightarrow$ behalf

Types of spelling errors: Real-word Errors

- **Typographical errors:** three $\rightarrow$ there
- **Cognitive errors (homophones):** piece $\rightarrow$ peace, too $\rightarrow$ 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 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 the error word
- Choose the best one:
  - Shortest weighted edit distance
  - Highest noisy channel probabliity

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

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

Choosing best candidate

- Noisy Channel

Noisy Channel

We see an observation x of the misspelled word

Find the correct word w

$$\hat{w} = \underset{w \in V}{\arg\max} P(w|x)$$

Noisy Channel

We see an observation x of the misspelled word

Find the correct word w

$$\hat{w} = \underset{w \in V}{\arg\max} P(w|x)$$

$$= \underset{w \in V}{\arg\max} \frac{P(x|w)P(w)}{P(x)}$$

Noisy Channel

We see an observation x of the misspelled word

Find the correct word w

$$\hat{w} = \underset{w \in V}{\arg\max} P(w|x)$$

$$= \underset{w \in V}{\arg\max} \frac{P(x|w)P(w)}{P(x)}$$

$$= \underset{w \in V}{\arg\max} P(x|w)P(w)$$

Non-word spelling error: acress

Words with similar spelling

Small edit distance to error

Words with similar pronuncitation

Small edit distance of pronunciation to error

Non-word spelling error: acress

Words with similar spelling

Small edit distance to error

Words with similar pronuncitation

Small edit distance of pronunciation to error

Damerau-Levenshtein edit distance

Minimum edit distance, where edits are:

Non-word spelling error: acress

Words with similar spelling

Small edit distance to error

Words with similar pronuncitation

Small edit distance of pronunciation to error

Damerau-Levenshtein edit distance

Minimum edit distance, where edits are:
Insertion, Deletion, Substitution,

Non-word spelling error: acress

Words with similar spelling

Small edit distance to error

Words with similar pronuncitation

Small edit distance of pronunciation to error

Damerau-Levenshtein edit distance

Minimum edit distance, where edits are:
Insertion, Deletion, Substitution,
Transposition of two adjacent letters

Words within edit 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

Candidate generation

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

Candidate generation

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

Allow deletion of space or hyphen

- thisidea $\rightarrow$ this idea
- inlaw $\rightarrow$ in-law

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)

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 are conditioned on previous character

Channel model

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

Channel model 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

Noisy channel probability 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	.00078
caress	ca	ac	ac|ca	.00000164	.00000170	.0028
access	c	r	r|c	.000000209	.0000916	.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

Using a bigram language model

- " ... versatile acress whose ..."

Using a bigram language model

- " ... versatile acress whose ..."
- Counts from the Corpus of Contemporary American English with add-1 smoothing

Using a bigram language model

- " ... versatile acress whose ..."
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile) = 0.000021, P(across|versatile) = 0.000021

Using a bigram language model

- " ... versatile acress whose ..."
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile) = 0.000021, P(across|versatile) = 0.000021
- P(whose|actress) = 0.0010, P(whose|across) = 0.000006

Using a bigram language model

- " ... versatile acress whose ..."
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile) = 0.000021, P(across|versatile) = 0.000021
- P(whose|actress) = 0.0010, P(whose|across) = 0.000006
- P("versatile actress whose") = 0.000021 * 0.0010 = 210 x 10^{-10}

Using a bigram language model

- " ... versatile acress whose ..."
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile) = 0.000021, P(across|versatile) = 0.000021
- P(whose|actress) = 0.0010, P(whose|across) = 0.000006
- P("versatile actress whose") = 0.000021 * 0.0010 = 210 x 10^{-10}
- P("versatile across whose") = 0.000021 * 0.000006 = 1 x10^{-10}

Real-word spelling errors

- The study was conducted mainly **be** John Black
- The design **an** construction of the system ...

Real-word spelling errors

- The study was conducted mainly **be** John Black
- The design **an** construction of the system ...

25-40% of spelling errors are real words

Noisy channel for real-word spell correction

Given a sentence $X = w_1, w_2, w_3 \ldots, w_n$

- Candidate (w_1) = $\{w_1, w'_1, w''_1, w'''_1, \ldots\}$
- Candidate (w_2) = $\{w_2, w'_2, w''_2, w'''_2, \ldots\}$
- Candidate (w_3) = $\{w_3, w'_3, w''_3, w'''_3, \ldots\}$

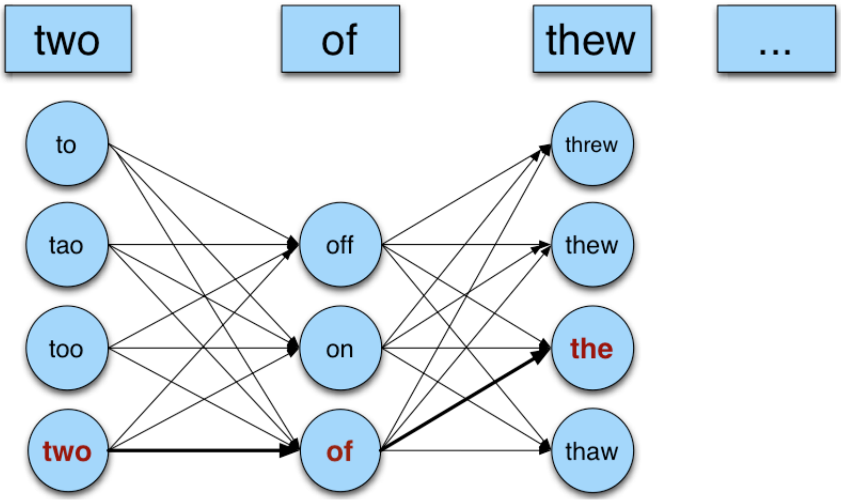
Noisy channel for real-word spell correction

Given a sentence $X = w_1, w_2, w_3 \ldots, w_n$

- Candidate (w_1) = $\{w_1, w'_1, w''_1, w'''_1, \ldots\}$
- Candidate (w_2) = $\{w_2, w'_2, w''_2, w'''_2, \ldots\}$
- Candidate (w_3) = $\{w_3, w'_3, w''_3, w'''_3, \ldots\}$

Choose the sequence W that maximizes $P(W|X)$

Noisy channel for real-world spell correction



Simplification: One error per sentence

Choose among all possible sentences with one word replaced

two of thew

- w_1, w''_2, w_3 two **off** thew
- w_1, w_2, w'_3 two of **the**
- w'''_1, w_2, w_3 **too** of thew

Simplification: One error per sentence

Choose among all possible sentences with one word replaced

two of thew

- w_1, w''_2, w_3 two **off** thew
- w_1, w_2, w'_3 two of **the**
- w'''_1, w_2, w_3 **too** of thew

Choose the sequence W that maximizes $P(W|X)$

Getting the probability values

Noisy Channel

$$\hat{W} = \arg\max_{W \in S} P(W|X)$$

where X is the observed sentence and S is the set of all the possible sequences from the candidate set

Getting the probability values

Noisy Channel

$$\hat{W} = \arg\max_{W \in S} P(W|X)$$

where X is the observed sentence and S is the set of all the possible sequences from the candidate set

$$= \arg\max_{W \in S} P(X|W)P(W)$$

Getting the probability values

Noisy Channel

$$\hat{W} = \arg\max_{W \in S} P(W|X)$$

where X is the observed sentence and S is the set of all the possible sequences from the candidate set

$$= \arg\max_{W \in S} P(X|W)P(W)$$

P(X|W)

- Same as for non-word spelling correction

Getting the probability values

Noisy Channel

$$\hat{W} = \arg\max_{W \in S} P(W|X)$$

where X is the observed sentence and S is the set of all the possible sequences from the candidate set

$$= \arg\max_{W \in S} P(X|W)P(W)$$

P(X|W)

- Same as for non-word spelling correction
- Also require proabability for no error $P(w|w)$

Probability of no error

What is the probability for a correctly typed word? P(“the”|“the”)

Probability of no error

What is the probability for a correctly typed word? P(“the”|“the”)

It may depend on the source text under consideration

- 1 error in 10 words $\rightarrow$ 0.9
- 1 error in 100 words $\rightarrow$ 0.99

Computing $P(W)$

Use Language Model

- Unigram
- Bigram
- ...

Context Sensitive Spelling Correction

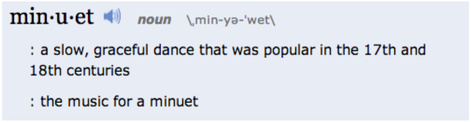
Context Sensitive Spelling Correction

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house



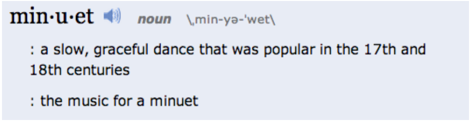
min·u·et noun \ˌmin-yə-ˈwet\

: a slow, graceful dance that was popular in the 17th and 18th centuries

: the music for a minuet

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house


min·u·et noun \ˌmin-yə-ˈwet\

: a slow, graceful dance that was popular in the 17th and 18th centuries

: the music for a minuet

Use a Language Model

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

Probablilistic Language Models: Applications

Speech Recognition

- P(I saw a van) $>>$ P(eyes awe of an)

Probablilistic Language Models: Applications

Speech Recognition

- P(I saw a van) $>>$ P(eyes awe of an)

Machine Translation

Which sentence is more plausible in the target language?

- P(**high** winds) > P(**large** winds)

Probablilistic Language Models: Applications

Speech Recognition

- P(I saw a van) $>>$ P(eyes awe of an)

Machine Translation

Which sentence is more plausible in the target language?

- P(**high** winds) > P(**large** winds)

Other Applications

- Context Sensitive Spelling Correction
- Natural Language Generation
- ...

Completion Prediction

- Language model also supports predicting the completion of a sentence.
  - Please turn off your cell ...
  - Your program does not ...

Completion Prediction

- Language model also supports predicting the completion of a sentence.
  - Please turn off your cell ...
  - Your program does not ...
- *Predictive text input* systems can guess what you are typing and give choices on how to complete it.

Probabilistic Language Modeling

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$$

Probabilistic Language Modeling

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$$

- **Related Task:** probability of an upcoming word:

$$P(w_4 | w_1, w_2, w_3)$$

Probabilistic Language Modeling

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$$

- **Related Task:** probability of an upcoming word:

$$P(w_4 | w_1, w_2, w_3)$$

- A model that computes either of these is called a **language model**

Computing $P(W)$

How to compute the joint probability

P(about, fifteen, minutes, from)

Computing $P(W)$

How to compute the joint probability

P(about, fifteen, minutes, from)

Basic Idea

Rely on the Chain Rule of Probability

The Chain Rule

Conditional Probabilities

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

The Chain Rule

Conditional Probabilities

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

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

The Chain Rule

Conditional Probabilities

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

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

More Variables

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

The Chain Rule

Conditional Probabilities

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

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

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

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

Probability of words in sentences

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

P("about fifteen minutes from") =

Probability of words in sentences

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

P("about fifteen minutes from") =

P(about) x P(fifteen | about) x P(minutes | about fifteen) x P(from | about fifteen minutes)

Estimating These Probability Values

Count and divide

$$P(\text{office} \mid \text{about fifteen minutes from}) = \frac{\mathit{Count}\,(\text{about fifteen minutes from office})}{\mathit{Count}\,(\text{about fifteen minutes from})}$$

Estimating These Probability Values

Count and divide

$$P(\text{office} \mid \text{about fifteen minutes from}) = \frac{\textit{Count}\ (\text{about fifteen minutes from office})}{\textit{Count}\ (\text{about fifteen minutes from})}$$

What is the problem

We may never see enough data for estimating these

Markov Assumption

Simplifying Assumption: Use only the previous word

P(office | about fifteen minutes from) $\approx$ P(office | from)

Markov Assumption

Simplifying Assumption: Use only the previous word

P(office | about fifteen minutes from) $\approx$ P(office | from)

Or the couple previous words

P(office | about fifteen minutes from) $\approx$ P(office | minutes from)

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

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

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

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

Using Markov Assumption: only k previous words

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

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

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

Using Markov Assumption: only k previous words

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

We approximate each component in the product

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

N-Gram Models

P(office | about fifteen minutes from)

An N-gram model uses only $N-1$ words of prior context.

N-Gram Models

P(office | about fifteen minutes from)

An N-gram model uses only $N-1$ words of prior context.

- Unigram: P(office)
- Bigram: P(office | from)
- Trigram: P(office | minutes from)

N-Gram Models

P(office | about fifteen minutes from)

An N-gram model uses only $N-1$ words of prior context.

- Unigram: P(office)
- Bigram: P(office | from)
- Trigram: P(office | minutes from)

Markov model and Language Model

N-Gram Models

P(office | about fifteen minutes from)

An N-gram model uses only $N-1$ words of prior context.

- Unigram: P(office)
- Bigram: P(office | from)
- Trigram: P(office | minutes from)

Markov model and Language Model

An N-gram model is an $N-1$-order Markov Model

N-Gram Models

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

N-Gram Models

- We can extend to trigrams, 4-grams, 5-grams
- In general, an insufficient model of language:
  language has long-distance dependencies:
  “The computer which I had just put into the machine room on the fifth floor **crashed**.”

N-Gram Models

- We can extend to trigrams, 4-grams, 5-grams
- In general, an insufficient model of language:
  language has long-distance dependencies:
  "The computer which I had just put into the machine room on the fifth floor **crashed**."
- In most of the applications, we can get away with N-gram models

Estimating N-grams probabilities

Estimating N-grams probabilities

Maximum Likelihood Estimate

Value that makes the observed data the "most probable"

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

Estimating N-grams probabilities

Maximum Likelihood Estimate

Value that makes the observed data the "most probable"

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

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

An Example

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

<s>I am here </s>
<s>who am I </s>
<s>I would like to know </s>

An Example

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

<s>I am here </s>
<s>who am I </s>
<s>I would like to know </s>

Estimating bigrams

P(I|<s>) =
P(</s>|here) =
P(would | I) =
P(here | am) =
P(know | like) =

An Example

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

<s>I am here </s>
<s>who am I </s>
<s>I would like to know </s>

Estimating bigrams

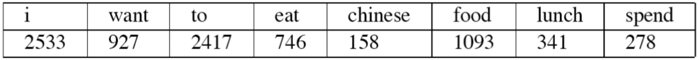
P(I|<s>) = 2/3
P(</s>|here) =1
P(would | I) = 1/3
P(here | am) = 1/2
P(know | like) = 0

Bigram counts from 9222 Restaurant 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

Computing bigram probabilities

Normlize by unigrams



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

Computing bigram probabilities

Normlize by unigrams

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

Bigram Probabilities

	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

Computing Sentence Probabilities

P(<s> I want english food </s>)

= P(I | <s>) x P(want | I) x P(english | want) x P(food | english) x P(</s> | food)

Computing Sentence Probabilities

P(<s> I want english food </s>)

= P(I | <s>) x P(want | I) x P(english | want) x P(food | english) x P(</s> | food)
= 0.000031

What knowledge does n-gram represent?

- 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

Everything in log space

- Avoids underflow

Practical Issues

Everything in log space

- Avoids underflow
- Adding is faster than multiplying

Practical Issues

Everything in log space

- Avoids underflow
- Adding is faster than multiplying

$$log(p_1 \times p_2 \times p_3 \times p_4) = log p_1 + log p_2 + log p_3 + log p_4$$

Handling zeros

Use smoothing

Language Modeling Toolkit

SRILM

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

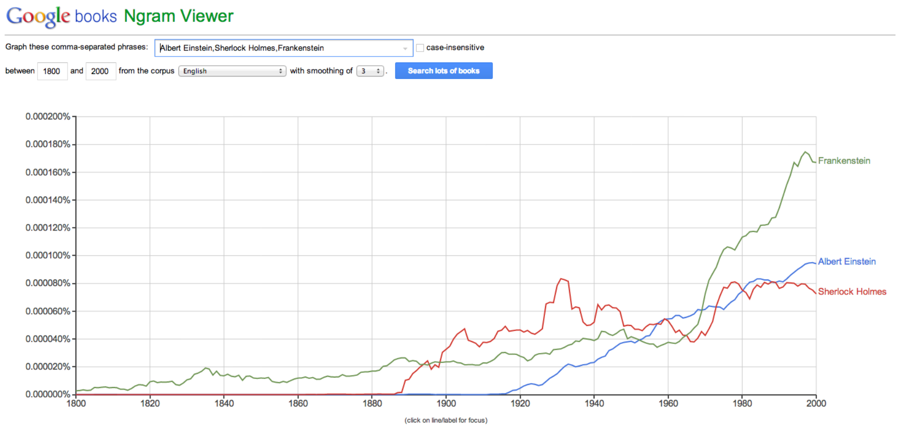
Google N-grams

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

Example from the 4-gram data

serve as the inspector	66
serve as the inspiration	1390
serve as the installation	136
serve as the institute	187
serve as the institution	279
serve as the institutional	461

Google books Ngram Data


Google books Ngram Viewer

Graph these comma-separated phrases: Albert Einstein,Sherlock Holmes,Frankenstein ☐ case-insensitive

between 1800 and 2000 from the corpus English with smoothing of 3 .

Evaluating Language Model: Basic Smoothing

Evaluating Language Model

Does it prefer good sentences to bad sentences?

Assign higher probability to real (or frequently observed) sentences than ungrammatical (or rarely observed) ones

Evaluating Language Model

Does it prefer good sentences to bad sentences?

Assign higher probability to real (or frequently observed) sentences than ungrammatical (or rarely observed) ones

Training and Test Corpora

- Parameters of the model are trained on a large corpus of text, called **training set**.
- Performance is tested on a disjoint (held-out) **test data** using an **evaluation metric**

Extrinsic evaluation of N-grams models

Comparison of two models, A and B

- Use each model for one or more tasks: *spelling corrector, speech recognizer, machine translation*
- Get accuracy values for A and B
- Compare accuracy for A and B

Intrinsic evaluation: Perplexity

Intuition: The Shannon Game

How well can we predict the next word?

Intrinsic evaluation: Perplexity

Intuition: The Shannon Game

How well can we predict the next word?

- I always order pizza with cheese and ...
- The president of India is ...
- I wrote a ...

Intrinsic evaluation: Perplexity

Intuition: The Shannon Game

How well can we predict the next word?

- I always order pizza with cheese and . . .
- The president of India is . . .
- I wrote a . . .

Unigram model doesn’t work for this game.

Intrinsic evaluation: Perplexity

Intuition: The Shannon Game

How well can we predict the next word?

- I always order pizza with cheese and . . .
- The president of India is . . .
- I wrote a . . .

Unigram model doesn't work for this game.

A better model of text

is one which assigns a higher probability to the actual word

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity ($PP(W)$)

Perplexity is the inverse probability of the test data, normalized by the number of words:

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity ($PP(W)$)

Perplexity is the inverse probability of the test data, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity ($PP(W)$)

Perplexity is the inverse probability of the test data, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

Applying chain Rule

$$PP(W) = \left(\prod \frac{1}{P(w_i | w_1 \dots w_{i-1})} \right)^{\frac{1}{N}}$$

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity ($PP(W)$)

Perplexity is the inverse probability of the test data, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

Applying chain Rule

$$PP(W) = \left(\prod \frac{1}{P(w_i|w_1 \ldots w_{i-1})}\right)^{\frac{1}{N}}$$

For bigrams

$$PP(W) = \left(\prod \frac{1}{P(w_i|w_{i-1})}\right)^{\frac{1}{N}}$$

Example: A Simple Scenario

- Consider a sentence consisting of N random digits

Example: A Simple Scenario

- Consider a sentence consisting of N random digits
- Find the perplexity of this sentence as per a model that assigns a probability $p = 1/10$ to each digit.

Example: A Simple Scenario

- Consider a sentence consisting of N random digits
- Find the perplexity of this sentence as per a model that assigns a probability $p = 1/10$ to each digit.

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

Example: A Simple Scenario

- Consider a sentence consisting of N random digits
- Find the perplexity of this sentence as per a model that assigns a probability $p = 1/10$ to each digit.

$$\begin{aligned} PP(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \left(\left(\frac{1}{10}\right)^N\right)^{-\frac{1}{N}} \end{aligned}$$

Example: A Simple Scenario

- Consider a sentence consisting of N random digits
- Find the perplexity of this sentence as per a model that assigns a probability $p = 1/10$ to each digit.

$$\begin{aligned} PP(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \left(\left(\frac{1}{10}\right)^N\right)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}\right)^{-1} \\ &= 10 \end{aligned}$$

Lower perplexity = better model

WSJ Corpus

Training: 38 million words

Test: 1.5 million words

Lower perplexity = better model

WSJ Corpus

Training: 38 million words

Test: 1.5 million words

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lower perplexity = better model

WSJ Corpus

Training: 38 million words

Test: 1.5 million words

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Unigram perplexity: 962?

The model is as confused on test data as if it had to choose uniformly and independently among 962 possibilities for each word.

The Shannon Visualization Method

Use the language model to generate word sequences

The Shannon Visualization Method

Use the language model to generate word sequences

- Choose a random bigram (<s>,w) as per its probability

The Shannon Visualization Method

Use the language model to generate word sequences

- Choose a random bigram (<s>,w) as per its probability
- Choose a random bigram (w,x) as per its probability

The Shannon Visualization Method

Use the language model to generate word sequences

- Choose a random bigram (<s>,w) as per its probability
- Choose a random bigram (w,x) as per its probability
- And so on until we choose </s>

The Shannon Visualization Method

Use the language model to generate word sequences

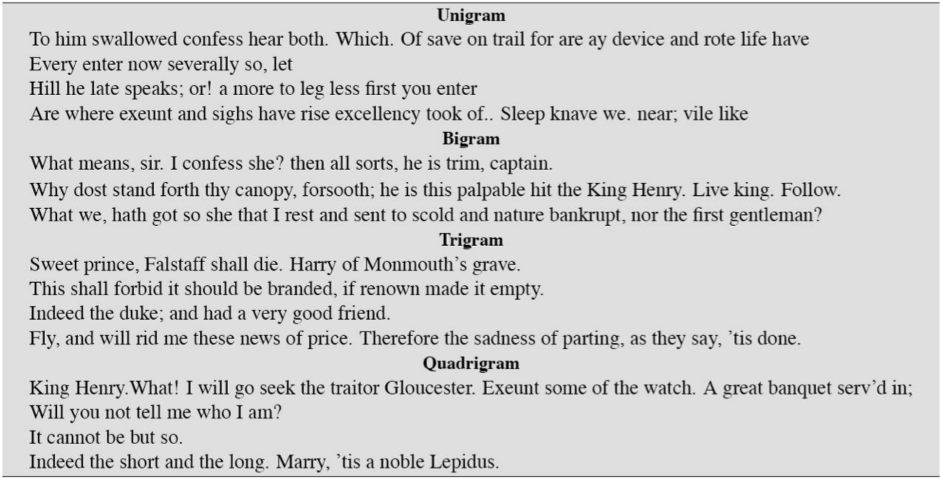
- Choose a random bigram (<s>,w) as per its probability
- Choose a random bigram (w,x) as per its probability
- And so on until we choose </s>

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

Shakespeare as Corpus

- N = 884,647 tokens, V = 29,066
- Shakespeare produced 300,000 bigram types out of $V^2 = 844$ million possible bigrams.

Approximating Shakespeare



Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.

Problems with simple MLE estimate: zeros

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

- ... denied the offer
- ... denied the loan

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

- ... denied the offer
- ... denied the loan

Zero probability n-grams

- P(offer | denied the) = 0

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

- ... denied the offer
- ... denied the loan

Zero probability n-grams

- P(offer | denied the) = 0
- The test set will be assigned a probability 0

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

- ... denied the offer
- ... denied the loan

Zero probability n-grams

- P(offer | denied the) = 0
- The test set will be assigned a probability 0
- And the perplexity can't be computed

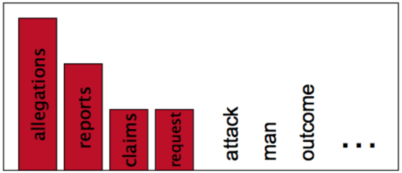
Language Modeling: Smoothing

Language Modeling: Smoothing

With sparse statistics

P(w | denied the)

3	allegations
2	reports
1	claims
1	request
7	total

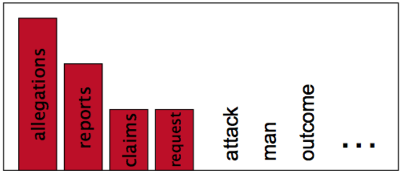


Language Modeling: Smoothing

With sparse statistics

P(w | denied the)
3 allegations
2 reports
1 claims
1 request

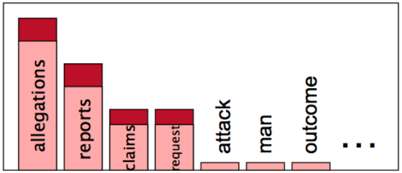
7 total



Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other

7 total



Laplace Smoothing (Add-one estimation)

- Pretend as if we saw each word (N-gram) one more time that we actually did

Laplace Smoothing (Add-one estimation)

- Pretend as if we saw each word (N-gram) one more time that we actually did
- Just add one to all the counts!

Laplace Smoothing (Add-one estimation)

- Pretend as if we saw each word (N-gram) one more time that we actually did
- Just add one to all the counts!
- MLE estimate for bigram: $P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)}{c(w_{i-1})}$

Laplace Smoothing (Add-one estimation)

- Pretend as if we saw each word (N-gram) one more time that we actually did
- Just add one to all the counts!
- MLE estimate for bigram: $P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)}{c(w_{i-1})}$
- Add-1 estimate: $P_{Add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)+1}{c(w_{i-1})+V}$

Reconstituted counts as effect of smoothing

Effective bigram count ($c^*(w_{n-1}w_n)$)

$$\frac{c^*(w_{n-1}w_n)}{c(w_{n-1})} = \frac{c(w_{n-1}w_n)+1}{c(w_{n-1})+V}$$

Comparing with bigrams: Restaurant corpus

	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

Comparing with bigrams: Restaurant corpus

	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

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

More general formulations: Add-k

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

More general formulations: Add-k

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

More general formulations: Add-k

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

Unigram prior smoothing:

$$P_{UnigramPrior}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + mP(w_i)}{c(w_{i-1}) + m}$$

More general formulations: Add-k

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

Unigram prior smoothing:

$$P_{UnigramPrior}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + mP(w_i)}{c(w_{i-1}) + m}$$

A good value of k or m?

Can be optimized on held-out set