



Amirkabir University of Technology
(Tehran Polytechnic)

Natural Language Processing

Lecture 18: Spell Correction

Amirkabir University of Technology

Dr Momtazi

Outline

- **Introduction**
- Definition of Minimum Edit Distance
- Computing Minimum Edit Distance
- Backtrace for Computing Alignments

Who cares about spelling?

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mtttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

(See for the story behind this supposed research report.)

Detection vs. Correction

- Two distinct tasks:
 - Error detection = simply find the misspelled words
 - Error correction = correct the misspelled words
- It might be easy to tell that “ater” is a misspelled word, but what is the correct word?

water? later? after?

What causes errors?

- Keyboard mistyping Space bar
- Keyboard proximity
- Similarity of shape
- Phonetic errors
- Knowledge problems

Keyboard Space Bar and Keyboard Proximity

- Space bar issues
 - Run-on errors = two separate words become one
 - e.g., the fuzz becomes thefuzz
 - Split errors = one word becomes two separate words
 - e.g., equalization becomes equali zation
- Keyboard proximity
 - e.g., Jack becomes Hack since h, j are next to each other on a typical American keyboard

Similarity of Shape

- Physical similarity
 - Similarity of shape, e.g., mistaking two physically similar letters when typing up something handwritten
 - e.g., tight for fight

Phonetic Errors

- Errors based on the sounds of a language (not necessarily on the letters)
 - Homophones = two words which sound the same
 - e.g., red/ read (past tense), cite/ site/ sight, they're/ their/ there
 - Spoonerisms = switching two letters/sounds around
 - e.g., It's a ravy(vary) grain with biscuit wheels.
 - Letter substitution = replacing a letter (or sequence of letters) with a similar-sounding one
 - e.g., John kracked his nuckles. instead of John cracked his knuckles
 - e.g., I study sikologee

Knowledge Problems

- Cases of not knowing how to spell:
 - Not knowing a word and guessing its spelling (can be phonetic)
 - e.g., sientist
 - Not knowing a rule and guessing it
 - e.g., Do we double a consonant for ing words? jog -> jogging ?

Spell Checking Approaches

- Minimum edit distance
- Language Modeling
- Sequence to Sequence Models
-

Outline

- Introduction
- **Definition of Minimum Edit Distance**
- Computing Minimum Edit Distance
- Backtrace for Computing Alignments

How similar are two strings?

- Spell correction
 - The user typed “graffe”
Which is closest?
 - graf
 - graft
 - grail
 - giraffe

- Computational Biology

- Align two sequences of nucleotides

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGGTCGATTGCCCGAC
```

- Resulting alignment:

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTGCCCGAC
```

- Also for Machine Translation, Information Extraction, Speech Recognition

Edit Distance

- The minimum edit distance between two strings is the minimum number of editing operations
 - Insertion
 - Deletion
 - Substitution
- Needed to transform one into the other

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

Alignment in Computational Biology

- Given a sequence of bases

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC  
TAGCTATCACGACCGCGGGTCGATTGCCCGAC
```

- An alignment:

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---  
TAG-CTATCAC--GACCGC--GGTCGATTGCCCGAC
```

- Given two sequences, align each letter to a letter or gap

Other uses of Edit Distance in NLP

- Evaluating Machine Translation and speech recognition

R Spokesman confirms senior government adviser was appointed

H Spokesman said the senior adviser was appointed

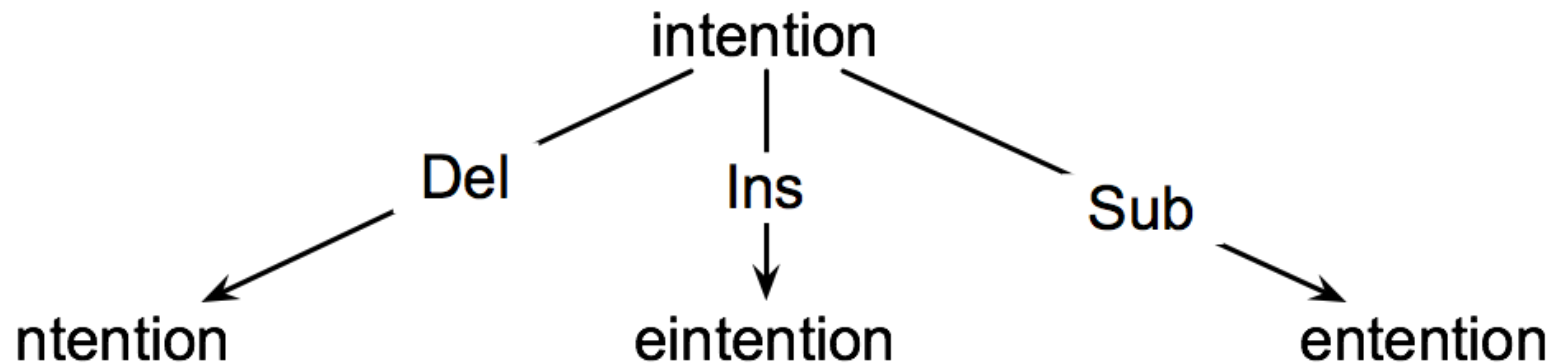
S

I

D

How to find the Min Edit Distance?

- 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 naively
 - Lots of distinct paths wind up at the same state.
 - We don't have to keep track of all of them
 - Just the shortest path to each of those revisited states.

Outline

- Introduction
- Definition of Minimum Edit Distance
- **Computing Minimum Edit Distance**
- Backtrace for Computing Alignments

Dynamic Programming for Minimum Edit Distance

- **Dynamic programming:** A tabular computation of $D(n, m)$
- Solving problems by combining solutions to subproblems.
- Bottom-up
 - We compute $D(i, j)$ for small i, j
 - And compute larger $D(i, j)$ based on previously computed smaller values
 - i.e., compute $D(i, j)$ for all $i (0 < i < n)$ and $j (0 < j < m)$

Defining Min Edit Distance (Levenshtein)

- Initialization

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

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

- Recurrence Relation:

For each $i = 1 \dots M$

For each $j = 1 \dots N$

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

- 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

Outline

- Introduction
- Definition of Minimum Edit Distance
- Computing Minimum Edit Distance
- **Backtrace for Computing Alignments**

Computing alignments

- Edit distance isn't sufficient
 - We often need to **align** each character 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

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

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

MinEdit 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 Distance

- Base conditions:

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

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

Termination:

$$D(N, M) \text{ is distance}$$

- Recurrence Relation:

For each $i = 1 \dots M$

For each $j = 1 \dots N$

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

Result of Backtrace

- Two strings and their **alignment**:

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

Performance

- Time:
 $O(nm)$
- Space:
 $O(nm)$
- Backtrace
 $O(n+m)$

Much More Space to Work

- Contextual information
- Language models
- Sources of errors
- Domain knowledge
- Common mistakes

Further Reading

- Speech and Language Processing (3rd ed. draft)
 - Chapter 2 + Appendix B