

SIT330-770: Natural Language Processing

Week 3 - Text processing

Regular Expressions, Text Normalization, Edit Distance

Dr. Mohamed Reda Bouadjene

School of Information Technology, Faculty of Sci Eng & Built Env

reda.bouadjene@deakin.edu.au



DEAKIN UNIVERSITY


1

SIT330-770: Natural Language Processing

Week 3.1 - Regular Expressions

Dr. Mohamed Reda Bouadjene

School of Information Technology, Faculty of Sci Eng & Built Env




DEAKIN UNIVERSITY

2

Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
 - woodchuck
 - woodchucks
 - Woodchuck
 - Woodchucks



DEAKIN UNIVERSITY

3

Regular Expressions: Disjunctions

- Letters inside square brackets []

Pattern	Matches
[wW]oodchuck	Woodchuck, woodchuck
[1234567890]	Any digit

- Ranges [A-Z]

Pattern	Matches
[A-Z]	An upper case letter
[a-z]	A lower case letter
[0-9]	A single digit

DEAKIN UNIVERSITY

4

Regular Expressions: Negation in Disjunction

- Negations [^ss]
 - Carat means negation only when first in []

Pattern	Matches
[^A-Z]	Not an upper case letter
[^Ss]	Neither 'S' nor 's'
[^e^]	Neither e nor ^
a^b	The pattern a carat b


DEAKIN UNIVERSITY

5

Regular Expressions: More Disjunction

- Woodchuck is another name for groundhog!
- The pipe | for disjunction

Pattern	Matches
groundhog woodchuck	woodchuck
yours mine	yours
a b c	= [abc]
[gG]roundhog [wW]oodchuck	Woodchuck



DEAKIN UNIVERSITY

6

Regular Expressions: **? * + .**

Pattern	Matches
<code>colou?r</code>	Optional previous char <u>color</u> <u>colour</u>
<code>oo*h!</code>	0 or more of previous char <u>oh!</u> <u>ooh!</u> <u>oooh!</u> <u>ooooh!</u>
<code>o+h!</code>	1 or more of previous char <u>oh!</u> <u>ooh!</u> <u>oooh!</u> <u>ooooh!</u>
<code>baa+</code>	<u>baa</u> <u>baaa</u> <u>baaaa</u> <u>baaaaa</u>
<code>beg.n</code>	<u>begin</u> <u>begun</u> <u>begun</u> <u>beg3n</u>

Stephen C Kleene
Kleene *, Kleene +

7

7

Regular Expressions: Anchors **^ \$**

Pattern	Matches
<code>^[A-Z]</code>	<u>P</u> alo Alto
<code>^[^A-Za-z]</code>	<u>"</u> 1 "Hello"
<code>\.\$</code>	The end <u>.</u>
<code>.\$</code>	The end <u>?</u> The end! <u>.</u>

8

8

Example

- Find me all instances of the word "the" in a text.

the

Misses capitalized examples

[tT]he

Incorrectly returns other or theology

[^a-zA-Z][tT]he[^a-zA-Z]

9

9

Errors

- The process we just went through was based on fixing two kinds of errors:
 - Matching strings that we should not have matched (there, then, other)
False positives (Type I errors)
 - Not matching things that we should have matched (The)
False negatives (Type II errors)

10

10

Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two antagonistic efforts:
 - Increasing accuracy or precision (minimizing false positives)
 - Increasing coverage or recall (minimizing false negatives).

11

11

Summary

- Regular expressions play a surprisingly large role
 - Sophisticated sequences of regular expressions are often the first model for any text processing
- For hard tasks, we use machine learning classifiers
 - But regular expressions are still used for pre-processing, or as features in the classifiers
 - Can be very useful in capturing generalizations

12

12

SIT330-770: Natural Language Processing

Week 3.2- More Regular Expressions: Substitutions and ELIZA

Dr. Mohamed Reda Bouadjene

School of Information Technology,
Faculty of Sci Eng & Built Env



13

Substitutions

- Substitution in Python and UNIX commands:


```
s/regexp1/pattern/
```

 e.g.:


```
s/colour/color/
```

14

Capture Groups

- Say we want to put angles around all numbers:


```
the 35 boxes → the <35> boxes
```
- Use parens () to "capture" a pattern into a numbered register (1, 2, 3...)
- Use \1 to refer to the contents of the register


```
s/([0-9]+)/<\1>/
```

15

Capture groups: multiple registers

- /the (.*?)er they (.*), the \1er we \2/
- Matches


```
the faster they ran, the faster we ran
```
- But not


```
the faster they ran, the faster we ate
```

16

But suppose we don't want to capture?

- Parentheses have a double function: grouping terms, and capturing
- Non-capturing groups: add a ?: after paren:
- E.g.: /(?:some|a few) (people|cats) like some \1/
 - matches
 - some cats like some cats
 - but not
 - some cats like some some

17

Lookahead assertions

- (?= pattern) is true if pattern matches, but is **zero-width**; **doesn't advance character pointer**
- (?! pattern) true if a pattern does not match
- How to match, at the beginning of a line, any single word that doesn't start with "Volcano":


```
o/^(?!Volcano)[A-Za-z]+/
```

18

Simple Application: ELIZA

- Early NLP system that imitated a Rogerian psychotherapist
 - Joseph Weizenbaum, 1966.
- Uses pattern matching to match, e.g.,:
 - "I need X"
 and translates them into, e.g.
 - "What would it mean to you if you got X?"

19

19

Simple Application: ELIZA

Men are all alike.
 IN WHAT WAY
 They're always bugging us about something or other. CAN YOU THINK OF A SPECIFIC EXAMPLE
 Well, my boyfriend made me come here.
 YOUR BOYFRIEND MADE YOU COME HERE
 He says I'm depressed much of the time.
 I AM SORRY TO HEAR YOU ARE DEPRESSED

20

20

How ELIZA works

- s/. * I'M (depressed|sad) . */ I AM SORRY TO HEAR YOU ARE \1/
- s/. * I AM (depressed|sad) . */ WHY DO YOU THINK YOU ARE \1/
- s/. * all . */ IN WHAT WAY?/
- s/. * always . */ CAN YOU THINK OF A SPECIFIC EXAMPLE?/

21

21

SIT330-770: Natural Language Processing

Week 3.3 - Words and Corpora

Dr. Mohamed Reda Bouadjene

School of Information Technology,
 Faculty of Sci Eng & Built Env

22

22

How many words in a sentence?

- "I do uh main- mainly business data processing"
- Fragments, filled pauses
- "Seuss's **cat** in the hat is different from other **cats**!"
- **Lemma**: same stem, part of speech, rough word sense
 - **cat** and **cats** = same lemma
- **Wordform**: the full inflected surface form
 - **cat** and **cats** = different wordforms

23

23

How many words in a sentence?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- How many?
 - 15 tokens (or 14)
 - 13 types (or 12) (or 11?)

24

24

How many words in a corpus?

N = number of tokens
V = vocabulary = set of types, **|V|** is size of vocabulary

Heaps Law = Herdan's Law = $|V| = kN^\beta$, where often $0.67 < \beta < 0.75$
 i.e., vocabulary size grows with > square root of the number of word tokens

	Tokens = N	Types = V
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13+ million

25

Corpora

Words don't appear out of nowhere!
 A text is produced by

- a specific writer(s),
- at a specific time,
- in a specific variety,
- of a specific language,
- for a specific function.

26

Corpora vary along dimension like

- **Language:** 7097 languages in the world
- **Variety**, like African American Language varieties.
 - AAE Twitter posts might include forms like "iont" (*I don't*)
- **Code switching**, e.g., Spanish/English, Hindi/English:

S/E: Por primera vez veo a @username actually being hateful! It was beautiful:
 [For the first time I get to see @username actually being hateful! It was beautiful.]

H/E: dost tha or ra- hega ... don't worry ... but dherya rakhe
 ["he was and will remain a friend ... don't worry ... but have faith"]
- **Genre:** newswire, fiction, scientific articles, Wikipedia
- **Author Demographics:** writer's age, gender, ethnicity, SES

27

Corpus datasheets

Gebru et al (2020), Bender and Friedman (2018)

Motivation:

- Why was the corpus collected?
- By whom?
- Who funded it?

Situation: In what situation was the text written?

Collection process: If it is a subsample how was it sampled? Was there consent? Pre-processing?

- +Annotation process, language variety, demographics, etc.

28

SIT330-770: Natural Language Processing

Week 3.4 - Word tokenization

Dr. Mohamed Reda Bouadjeneq

School of Information Technology,
 Faculty of Sci Eng & Built Env



29

Text Normalization

- Every NLP task requires text normalization:
 1. Tokenizing (segmenting) words
 2. Normalizing word formats
 3. Segmenting sentences

30

Space-based tokenization

- A very simple way to tokenize
 - For languages that use space characters between words
 - Arabic, Cyrillic, Greek, Latin, etc., based writing systems
 - Segment off a token between instances of spaces
- Unix tools for space-based tokenization
 - The "tr" command
 - Inspired by Ken Church's UNIX for Poets
 - Given a text file, output the word tokens and their frequencies

31

Simple Tokenization in UNIX

- Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | uniq -c
```

Change all non-alpha to newlines
Sort in alphabetical order
Merge and count each type

```
1945 A
 72 AARON
 19 ABESS
  5 ABBOT
... ..
```

32

The first step: tokenizing

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairrest
creatures
We
...
```

33

The second step: sorting

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | head
```

```
A
A
A
A
A
A
A
A
...
```

34

More counting

- Merging upper and lower case


```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c
```
- Sorting the counts


```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c | sort -n -r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
 8954 d
```

What happened here?

35

Issues in Tokenization

- Can't just blindly remove punctuation:
 - m.p.h., Ph.D., AT&T, cap'n
 - prices (\$45.55)
 - dates (01/02/06)
 - URLs (<http://www.stanford.edu>)
 - hashtags (#nlproc)
 - email addresses (someone@cs.colorado.edu)
- Clitic: a word that doesn't stand on its own
 - "are" in *we're*, French "je" in *j'ai*, "le" in *l'honneur*
- When should multiword expressions (MWE) be words?
 - New York, rock 'n' roll

36

Tokenization in NLTK

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'((?x) # set flag to allow verbose regexps
... ([A-Z]\.)* # abbreviations, e.g. U.S.A.
... | \w+(-\w+)* # words with optional internal hyphens
... | \$?(d+\.d+)?%? # currency and percentages, e.g. $12.40, 82%
... | \.\.\. # ellipsis
... | [\'\".,:;'"'()?@!&*_`~\|-_]) # these are separate tokens; includes ], [
... )'
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

37

Tokenization in NLTK

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'((?x) # set flag to allow verbose regexps
... ([A-Z]\.)* # abbreviations, e.g. U.S.A.
```

38

Tokenization in languages without spaces

- Many languages (like Chinese, Japanese, Thai) don't use spaces to separate words!
- How do we decide where the token boundaries should be?

39

Word tokenization in Chinese

- Chinese words are composed of characters called "**hanzi**" (or sometimes just "**zi**")
- Each one represents a meaning unit called a morpheme.
- Each word has on average 2.4 of them.
- But deciding what counts as a word is complex and not agreed upon.

40

How to do word tokenization in Chinese?

姚明进入总决赛 "Yao Ming reaches the finals"

- 3 words?

姚明 进入 总决赛

YaoMing reaches finals

- 5 words?

姚 明 进 入 总 决 赛

Yao Ming reaches overall finals

- 7 characters? (don't use words at all):

姚 明 进 入 总 决 赛

Yao Ming enter enter overall decision game

41

Word tokenization / segmentation

- So, in Chinese it's common to just treat each character (zi) as a token.
 - So, the **segmentation** step is very simple
- In other languages (like Thai and Japanese), more complex word segmentation is required.
 - The standard algorithms are neural sequence models trained by supervised machine learning.

42

SIT330-770: Natural Language Processing

Week 3.5 - Byte Pair Encoding

Dr. Mohamed Reda Bouadjene

School of Information Technology,
Faculty of Sci Eng & Built Env



43

Another option for text tokenization

- Instead of
 - white-space segmentation
 - single-character segmentation
- **Use the data** to tell us how to tokenize.
- **Subword tokenization** (because tokens can be parts of words as well as whole words)

44

Subword tokenization

- Three common algorithms:
 - **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
 - **Unigram language modeling tokenization** (Kudo, 2018)
 - **WordPiece** (Schuster and Nakajima, 2012)
- All have 2 parts:
 - A token **learner** that takes a raw training corpus and induces a vocabulary (a set of tokens).
 - A token **segmenter** that takes a raw test sentence and tokenizes it according to that vocabulary

45

Byte Pair Encoding (BPE) token learner

- Let vocabulary be the set of all individual characters
= {A, B, C, D, ..., a, b, c, d, ...}
- Repeat:
 - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
 - Add a new merged symbol 'AB' to the vocabulary
 - Replace every adjacent 'A' 'B' in the corpus with 'AB'.
- Until k merges have been done.

46

BPE token learner algorithm

```

function BYTE-PAIR ENCODING(strings  $C$ , number of merges  $k$ ) returns vocab  $V$ 
   $V \leftarrow$  all unique characters in  $C$       # initial set of tokens is characters
  for  $i = 1$  to  $k$  do                      # merge tokens til  $k$  times
     $t_L, t_R \leftarrow$  Most frequent pair of adjacent tokens in  $C$ 
     $t_{NEW} \leftarrow t_L + t_R$           # make new token by concatenating
     $V \leftarrow V + t_{NEW}$               # update the vocabulary
    Replace each occurrence of  $t_L, t_R$  in  $C$  with  $t_{NEW}$  # and update the corpus
  return  $V$ 
  
```

47

Byte Pair Encoding (BPE) Addendum

- Most subword algorithms are run inside space-separated tokens.
- So we commonly first add a special end-of-word symbol '___' before space in training corpus
- Next, separate into letters.

48

BPE token learner

- Original (very fascinating 😊) corpus:
- low low low low low lowest lowest newer newer newer newer newer
- newer wider wider wider new new
- Add end-of-word tokens, resulting in this vocabulary:

Corpus	Vocabulary
5 low _	_, d, e, i, l, n, o, r, s, t, w
2 low est _	
6 new er _	
3 wid er _	
2 new _	

49

49

BPE

Corpus	Vocabulary
5 low _	_, d, e, i, l, n, o, r, s, t, w
2 low est _	
6 new er _	
3 wid er _	
2 new _	

Merge [e r] to [er]

Corpus	Vocabulary
5 low _	_, d, e, i, l, n, o, r, s, t, w, er
2 low est _	
6 new er _	
3 wid er _	
2 new _	

50

50

BPE

Corpus	Vocabulary
5 low _	_, d, e, i, l, n, o, r, s, t, w, er
2 low est _	
6 new er _	
3 wid er _	
2 new _	

Merge [er _] to [er_]

Corpus	Vocabulary
5 low _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2 low est _	
6 new er_ _	
3 wid er_ _	
2 new _	

51

51

BPE

Corpus	Vocabulary
5 low _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2 low est _	
6 new er _	
3 wid er _	
2 new _	

Merge [n e] to [ne]

Corpus	Vocabulary
5 low _	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne
2 low est _	
6 ne w er _	
3 wid er _	
2 ne w _	

52

52

BPE

- The next merges are:

Merge	Current Vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

53

53

BPE token segmenter algorithm

On the test data, run each merge learned from the training data:

- Greedily
- In the order we learned them
- (test frequencies don't play a role)

So: merge every [e r] to [er], then merge [er _] to [er_], etc.

- Result:
 - Test set "n e w e r _" would be tokenized as a full word
 - Test set "l o w e r _" would be two tokens: "low er_"

54

54

Properties of BPE tokens

- Usually include frequent words and frequent subwords
 - Which are often morphemes like *-est* or *-er*
- A **morpheme** is the smallest meaning-bearing unit of a language
 - unlikeliest* has 3 morphemes *un-*, *likely*, and *-est*

55

55

SIT330-770: Natural Language Processing

Week 3.6 - Word Normalization and other issues

Dr. Mohamed Reda Bouadjeneq

School of Information Technology,
Faculty of Sci Eng & Built Env

56

56

Word Normalization

- Putting words/tokens in a standard format
 - U.S.A. or USA
 - uhhuh or uh-huh
 - Fed or fed
 - am, is, be, are

57

57

Case folding

- Applications like IR: reduce all letters to lower case
 - Since users tend to use lower case
 - Possible exception: upper case in mid-sentence?
 - e.g., *General Motors*
 - Fed* vs. *fed*
 - SAIL* vs. *sail*
- For sentiment analysis, MT, Information extraction
 - Case is helpful (*US* versus *us* is important)

58

58

Lemmatization

- Represent all words as their lemma, their shared root = dictionary headword form:
 - am, are, is* → *be*
 - car, cars, car's, cars'* → *car*
 - Spanish *quiero* ('I want'), *quieres* ('you want')
→ *querer* 'want'
 - He is reading detective stories*
→ *He be read detective story*

59

59

Lemmatization is done by Morphological Parsing

- Morphemes:
 - The small meaningful units that make up words
 - Stems**: The core meaning-bearing units
 - Affixes**: Parts that adhere to stems, often with grammatical functions
- Morphological Parsers:
 - Parse *cats* into two morphemes *cat* and *s*
 - Parse Spanish *amaren* ('if in the future they would love') into morpheme *amar* 'to love', and the morphological features *3PL* and *future subjunctive*.

60

60

Stemming

- Reduce terms to stems, chopping off affixes crudely

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things—names and heights and soundings—with the single exception of the red crosses and the written notes.

→

Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note.

61

Porter Stemmer

- Based on a series of rewrite rules run in series
 - A cascade, in which output of each pass fed to next pass
- Some sample rules:

ATIONAL	→	ATE (e.g., ATIONAL→ATE)
ING	→	ε if stem contains vowel (e.g., motoring → motor)
SSSES	→	SS (e.g., grasses → grass)

62

Dealing with complex morphology is necessary for many languages

- e.g., the Turkish word:
- Uygarlastiramadiklarimizdanmissinizcasina
- '(behaving) as if you are among those whom we could not civilize'
- Uygar 'civilized' + las 'become'
 - + tir 'cause' + ama 'not able'
 - + dik 'past' + lar 'plural'
 - + imiz 'p1pl' + dan 'abl'
 - + mis 'past' + siniz '2pl' + casina 'as if'

63

Sentence Segmentation

- ! , ? mostly unambiguous but period "." is very ambiguous
 - Sentence boundary
 - Abbreviations like Inc. or Dr.
 - Numbers like .02% or 4-3
- Common algorithm: Tokenize first: use rules or ML to classify a period as either (a) part of the word or (b) a sentence-boundary.
 - An abbreviation dictionary can help
- Sentence segmentation can then often be done by rules based on this tokenization.

64

SIT770: Natural Language Processing

Week 3.7 - Definition of Minimum Edit Distance

Dr. Mohamed Reda Bouadjene

School of Information Technology,
Faculty of Sci Eng & Built Env



65

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


```
AGGCTATCACTGACCTGACGCGGATGCCC
TAGCTATCAACGACGCGGATGATTTGCCCGAC
```
 - Resulting alignment:


```
--AGGCTATCACTGACCTGACGCGGATGCCC--
TAGCTATCACTGACCTGACGCGGATGATTTGCCCGAC
```

Also for Machine Translation, Information Extraction, Speech Recognition

66

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

67

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
  
```

68

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

69

Alignment in Computational Biology

- Given a sequence of bases


```

AGGCTATCACCTGACCTCCAGGCCGATGCC
TAGCTATCACGACCGCGGTCGATTGCCCGAC
      
```
- An alignment:


```

-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC--
TAG-CTATCAC--GACCGC--GGTCGATTGCCCGAC
      
```
- Given two sequences, align each letter to a letter or gap

70

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 I
      
```
- Named Entity Extraction and Entity Coreference
 - IBM Inc. announced today
 - IBM profits
 - Stanford Professor Jennifer Eberhardt announced yesterday
 - for Professor Eberhardt...

71

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

```

      intention
     /  |  \
  Del Ins Sub
 /   |   \
intention intention intention
  
```

72

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.

73

73

SIT770: Natural Language Processing

Week 3.8 - Computing Minimum Edit Distance

Dr. Mohamed Reda Bouadjene

School of Information Technology,
Faculty of Sci Eng & Built Env

74

74

Defining Min Edit Distance

- 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
 - The edit distance between X and Y is thus $D(n,m)$

75

75

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 ($0 < j < m$)

76

76

Defining Min Edit Distance (Levenshtein)

Initialization

$$D(1,0) = 1$$

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

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

77

77

The Edit Distance Table

N	5								
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
#	E	X	E	C	U	T	I	O	N

78

78

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	

79

79

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	

80

80

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

81

81

SIT770: Natural Language Processing

Week 3.9 - Backtrace for Computing Alignments

Dr. Mohamed Reda Bouadjene

School of Information Technology,
Faculty of Sci Eng & Built Env

82

82

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

83

83

Edit Distance

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

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	

84

84

MinEdit with Backtrace

n	9	8	7	6	5	4	3	2	1	0	9	8
o	8	7	6	5	4	3	2	1	0	9	8	7
i	7	6	5	4	3	2	1	0	9	8	7	6
t	6	5	4	3	2	1	0	9	8	7	6	5
n	5	4	3	2	1	0	9	8	7	6	5	4
e	4	3	2	1	0	9	8	7	6	5	4	3
t	3	2	1	0	9	8	7	6	5	4	3	2
n	2	1	0	9	8	7	6	5	4	3	2	1
i	1	0	9	8	7	6	5	4	3	2	1	0
#	0	1	2	3	4	5	6	7	8	9	8	7
#	e	x	e	c	u	t	i	o	n			

85

Adding Backtrace to Minimum Edit Distance

Base conditions: $D(i, 0) = i$ Termination: $D(0, j) = j$ $D(N, M)$ is distance

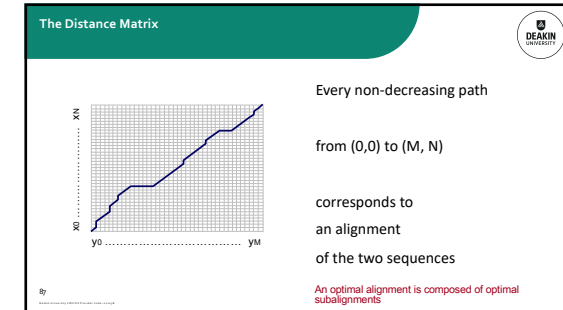
Recurrence Relation:

For each $i = 1..M$
For each $j = 1..N$

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

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

86



87

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

88

Performance

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

89

SIT770: Natural Language Processing

Week 3.10 - Weighted Minimum Edit Distance

Dr. Mohamed Reda Bouadjene

School of Information Technology,
Faculty of Sci Eng & Built Env

90

Weighted Edit Distance

- Why would we add weights to the computation?

- Spell Correction: some letters are more likely to be mistyped than others
- Biology: certain kinds of deletions or insertions are more likely than others

91

Confusion matrix for spelling errors

		sub(X, Y) = Substitution of X (incorrect) for Y (correct)																									
		X \ Y																									
X \ Y	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
b	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
d	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
e	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
f	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
g	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
k	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
l	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
n	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
p	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
s	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
v	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

92



93

Weighted Min Edit Distance

- Initialization:

$$\begin{aligned}
 D(0,0) &= 0 \\
 D(i,0) &= D(i-1,0) + \text{del}[x(i)]; \quad 1 \leq i \leq N \\
 D(0,j) &= D(0,j-1) + \text{ins}[y(j)]; \quad 1 \leq j \leq M
 \end{aligned}$$

- 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

94

Where did the name, dynamic programming, come from?

...The 1950s were not good years for mathematical research. [the] Secretary of Defense ...had a pathological fear and hatred of the word, research... I decided therefore to use the word, "programming". I wanted to get across the idea that this was dynamic, this was multistage... I thought, let's ... take a word that has an absolutely precise meaning, namely **dynamic**... it's impossible to use the word, **dynamic**, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to."

Richard Bellman, "Eye of the Hurricane: an autobiography" 1984.

95

SIT770: Natural Language Processing

Week 3.11 - Minimum Edit Distance in Computational Biology

Dr. Mohamed Reda Bouadjene

School of Information Technology,
Faculty of Sci Eng & Built Env

96

Sequence Alignment

AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTGCCCCGAC

-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC--
TAG-CTATCAC--GACCGC--GGTCGATTGCCCCGAC

97

97

Why sequence alignment?

- Comparing genes or regions from different species
 - to find important regions
 - determine function
 - uncover evolutionary forces
- Assembling fragments to sequence DNA
- Compare individuals to looking for mutations

98

98

Alignments in two fields

- In Natural Language Processing
 - We generally talk about **distance** (minimized)
 - And **weights**
- In Computational Biology
 - We generally talk about **similarity** (maximized)
 - And **scores**

99

99

The Needleman-Wunsch Algorithm

- Initialization:
 - $D(i, 0) = -i * d$
 - $D(0, j) = -j * d$
- Recurrence Relation:

$$D(i, j) = \min \begin{cases} D(i-1, j) & -d \\ D(i, j-1) & -d \\ D(i-1, j-1) + s[x(i), y(j)] \end{cases}$$
- Termination:
 - $D(N, M)$ is distance

100

100

The Needleman-Wunsch Matrix

(Note that the origin is at the upper left.)

101

101

A variant of the basic algorithm:

- Maybe it is OK to have an unlimited # of gaps in the beginning and end:

-----CTATCACCTGACCTCCAGGCCGATGCCCTTCCGGC
GCGAGTTCATCTATCAC--GACCGC--GGTCG-----

- If so, we don't want to penalize gaps at the ends

102

102

Different types of overlaps

Example:
2 overlapping "reads" from a sequencing project

Example:
Search for a mouse gene within a human chromosome

103

The Overlap Detection variant

Changes:

1. Initialization
For all i, j ,
 $F(i, 0) = 0$
 $F(0, j) = 0$
2. Termination
 $F_{opt} = \max \begin{cases} \max_i F(i, N) \\ \max_j F(M, j) \end{cases}$

104

The Local Alignment Problem

Given two strings $X = X_1 \dots X_{M_t}$
 $Y = Y_1 \dots Y_N$

Find substrings x', y' whose similarity (optimal global alignment value) is maximum

$x = \text{aaaaccccggggtta}$
 $y = \text{ttccggggaaccaacc}$

105

The Smith-Waterman algorithm

Idea: Ignore badly aligning regions

Modifications to Needleman-Wunsch:

Initialization: $F(0, j) = 0$
 $F(i, 0) = 0$

Iteration: $F(i, j) = \max \begin{cases} 0 \\ F(i-1, j) - d \\ F(i, j-1) - d \\ F(i-1, j-1) + s(x_i, y_j) \end{cases}$

106

The Smith-Waterman algorithm

Termination:

1. If we want the **best** local alignment...
 $F_{opt} = \max_{i,j} F(i, j)$
Find F_{opt} and trace back
2. If we want **all** local alignments **scoring > t**
?? For all i, j find $F(i, j) > t$, and trace back?
Complicated by overlapping local alignments

107

Local alignment example

$X = \text{ATCAT}$
 $Y = \text{ATTATC}$

Let:
 $m = 1$ (1 point for match)
 $d = 1$ (-1 point for del/ins/sub)

		A	T	T	A	T	C
	0	0	0	0	0	0	0
A	0						
T	0						
C	0						
A	0						
T	0						

108

Local alignment example

X = ATCAT
Y = ATTATC

	A	T	A	T	C
	0	0	0	0	0
A	0	1	0	0	1
T	0	0	2	1	0
C	0	0	1	1	1
A	0	1	0	0	2
T	0	0	2	0	1

109

109

Local alignment example

X = **ATCAT**
Y = **ATTATC**

	A	T	A	T	C
	0	0	0	0	0
A	0	1	0	0	1
T	0	0	2	1	0
C	0	0	1	1	1
A	0	1	0	0	2
T	0	0	2	0	1

110

110

Local alignment example

X = **ATCAT**
Y = ATT**ATC**

	A	T	A	T	C
	0	0	0	0	0
A	0	1	0	0	1
T	0	0	2	1	0
C	0	0	1	1	1
A	0	1	0	0	2
T	0	0	2	0	1

111

111