Tutorial 3
Text processing

Xi Chen
E-mail: xichen7@link.cuhk.edu.cn

Outline

- Regular Expressions
- Word Tokenize
- Edit distance

Regular Expressions

Many languages allow programmers to define regexes and then use them to:

- **Validate** that a piece of text (or a portion of that text) matches some pattern
- **Find** fragments of some text that match some pattern
- **Extract** fragments of some text
- **Replace** fragments of text with other text

Regular Expressions

Example 1 : Find "color" or "colour" in a given string.

```
import re
p = re.compile("colou?r")
m = p.search("The color green")
print(m.start())                    # 4
m = p.search("abc")
print(m)                            # None
```

Regular Expressions

Example 2 : Match any email address from the domains yahoo.com, hotmail.com, and gmail.com.

```
p = re.compile("(\W|^)([\w.\-]{0,25}@(yahoo|hotmail|gmail)\.com)(\W|$)")
m = p.search("My email is cxxx-17@gmail.com.")
print(m.group(2))     # cxxx-17@gmail.com
m = p.search("cxxx-17@hotmail.com is my email.")
print(m.group(2))     # cxxx-17@hotmail.com
m = p.search("cxxx-17@163.com is my email.")
print(m)                    # None
```

Reference: https://docs.python.org/zh-cn/3/library/re.html

Word Tokenize

- Different tools for tokenization
  - White Space Tokenization

```
sentence = "I was born in Tunisia in 1995."
sentence.split()
```

```
['I', 'was', 'born', 'in', 'Tunisia', 'in', '1995.']
```

```
sentence = "I was born in Tunisia in 1995, I am 26 years old"
sentence.split(', ')
```

```
['I was born in Tunisia in 1995', 'I am 26 years old']
```

Word Tokenize

- Different tools for tokenization
  - NLTK Word Tokenize

  NLTK (Natural Language Toolkit) is an open-source Python library for Natural Language Processing. You can easily tokenize the sentences and words of the text with the tokenize module of NLTK.

```
import nltk
from nltk.tokenize import (word_tokenize,
                           sent_tokenize,
                           TreebankWordTokenizer,
                           wordpunct_tokenize,
                           TweetTokenizer,
                           MWETokenizer)
text="Hope, is the only thing stronger than fear! #Hope #Amal.M"
```

Word Tokenize

- Different tools for tokenization
  - NLTK Word Tokenize

  Word and Sentence tokenizer

```
print(word_tokenize(text))
```

```
['Hope', ',', 'is', 'the', 'only', 'thing', 'stronger', 'than', 'fear', '!', '#', 'Hope', '#', 'Amal.M']
```

```
print(sent_tokenize(text))
```

```
['Hope, is the only thing stronger than fear!', '#Hope #Amal.M']
```

  Punctuation-based tokenizer

```
print(wordpunct_tokenize(text))
```

```
['Hope', ',', 'is', 'the', 'only', 'thing', 'stronger', 'than', 'fear', '!', '#', 'Hope', '#', 'Amal', '.', 'M']
```

Word Tokenize

- Different tools for tokenization
  - NLTK Word Tokenize

  Treebank Word tokenizer

```
text="What you don't want to be done to yourself, don't do to others..."
tokenizer= TreebankWordTokenizer()
print(tokenizer.tokenize(text))
```

```
['What', 'you', 'do', "n't", 'want', 'to', 'be', 'done', 'to', 'yourself', ',', 'do', "n't", 'do', 'to', 'others',
'...']
```

  - TextBlob Word Tokenize, spaCy Tokenizer, Gensim word tokenizer, jieba(for chinese), etc

reference: https://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.treebank

Edit distance

Given two character strings s_1 and s_2, the edit distance between them is the minimum number of edit operations required to transform s_1 into s_2

Example:

Input: str1 = “sunday”, str2 = “saturday”

Output: 3

Explanation: Last three and first characters are same. We basically need to convert “un” to “atur”. This can be done using below three operations. Replace ‘n’ with ‘r’, insert t, insert a

	0	s	a	t	u	r	d	a	y
0	0	1	2	3	4	5	6	7	8
s	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	5	5	5	5	4	3

Edit distance

Applications of Edit distance

- Character Error Rate (CER) is a metric of the performance of an automatic speech recognition (ASR) system.

This value indicates the percentage of characters that were incorrectly predicted. The lower the value, the better the performance of the ASR system with a CharErrorRate of 0 being a perfect score. Character error rate can then be computed as:

$$CharErrorRate = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

S is the number of substitutions,
D is the number of deletions,
I is the number of insertions,
C is the number of correct characters,
N is the number of characters in the reference (N=S+D+C).

Edit distance

Applications of Edit distance

Input: prediction = “sunday”, reference = “saturday”

$$CharErrorRate = \frac{S + D + I}{N} = \frac{3}{8} = 0.375$$

- the correction of spelling mistakes or OCR errors
- approximate string matching
- check for plagiarism
- etc.

Edit distance

Exercise:

Implement the edit distance algorithm

Thanks