Text normalisation

1. **Process of transforming text into a single canonical form**
2. **Before text normalization we should be aware of**

   i) what type of text is to be normalized and,

   ii) how it is to be processed afterwards

Why text normalisation ?

> 1) **In string searching** *e.g. 'john' and 'John' (Case based matching)*
>
> 2) **American or British English spelling**
>
> 3) **Multiple form of single word** e.g USA or US
>
> 4) **frequently used in converting text to speech**
>
> > Numbers, dates, acronyms, and abbreviations are non-standard "words" that need to be pronounced differently depending on context

Sample example of text normalisation

```
1 import numpy as np
2 import string
3 import re
```

```
1 data_list = ['The Patient! is s wai@ting, for5 you in room Number','johN', 'J
2 print(data_list)
```

```
1 preprocessed_text = []
```

```
1
2 for i in range(len(data_list)):
3         data = data_list[i]
4         # Tokenize i.e. split on white spaces
5         data = data.split()
6         # Convert to lowercase
7         data = [word.lower() for word in data]
```

✓ 0s completed at 10:36 AM

```
        # Remove tokens with special character
        data = [re.sub(r"[@?\(^)+\) 0-9]", "", word) for word in data ]  # No
        # Store as string
        data  =  ' '.join(data)
        preprocessed_text.append(data)
```

```
print("******Text before preprocessing******")
for text in data_list:
  print(text)

print("******Text after preprocessing******")
for text in preprocessed_text:
  print(text)
```

1. Parsing

---

2. Morpheme

---

3. Stemming

- chopping affies from a word
- may or may not has dictionary meaning
- Used in **information retrieval** for searching, Sentiment Analysis, document clustring. e.g search for party (search engine will show parties)
- Mostly used stemmer **Porter stemmer**
- other stemmers:

    1. Lovins Stemmer
    2. Dawson Stemmer
    3. Krovetz Stemmer
    4. Xerox Stemmer
    5. N-Gram Stemmer

Steps to use Porter stemmer

Step 1 - Import the NLTK library and from NLTK import PorterStemmer

Step 2 - Creat a variable and store PorterStemmer into it

Step 3 - use PorterStemmer

Stemming of words

```
1 import nltk
2 from nltk.stem import PorterStemmer
```

```
1 ps = PorterStemmer()
```

```
1 print(ps.stem('bat'))
2 print(ps.stem('batting'))
```

Stemming of a sentence

```
1 from nltk.tokenize import word_tokenize
2 nltk.download('punkt')
```

```
1 text = "This was not the map we found in Billy Bones’s chest, but an accurate
```

```
1 words = word_tokenize(text) # or you can use above approach of spliting based
2 print(words)
```

```
1 stemed_words = []
2 for w in words:
3   stemed_words.append(ps.stem(w))
```

```
1 print(stemed_words)
```

```
He is reading detective stories
⬇️ ⬇️ ⬇️        ⬇️       ⬇️
He be read   detective   story     NOTE:  am, are, and is have the shared lemma be
```

5. Different lemmatizers:

    1. WordnetLemmatizer
    2. spaCy
    3. TexxtBlob Lemmatizer
    4. Pattern Lemmatizer
    5. Standford CoreNLP Lemmatizer
    6. Gensim lemmatizer

Lemmatization of words

```
1 from nltk.stem import WordNetLemmatizer
2 nltk.download('wordnet') #wordnet last tutorial
3 nltk.download('averaged_perceptron_tagger') # will download pos tag
```

```
1 lemmatizer = WordNetLemmatizer()
```

```
1 print(lemmatizer.lemmatize('sang'))
```

Lemmatization of text

```
text = "This was not the map we found in Billy Bones’s chest, but an accurate
```

```
words = word_tokenize(text) # or you can use above approach of spliting based
print(words)
```

```
lemma_words = []
for w in words:
  lemma_words.append(lemmatizer.lemmatize(w, pos='v'))
```

```
print(lemma_words)
```

```
# converting list into text stream
print("***** Text before lemmatization*****\n")
print(text)
lemmatized_text = ' '.join(lemma_words)
print("\n***** Text after lemmatization*****\n")
print(lemmatized_text)
```