Text is everywhere and grows

By humans:

- Literature: books, poems
- Social media and news
- Administrative records: meteorology, finance
- Papers
- ...

Text analysis and visualization:

- Data mining
- Corpus linguist
- Language learning

Text analysis challenges

- **High-dimensional** aspects of language; many entities, relations
- **Variation** in spelling and style across different epochs and places
- **Diversity** of languages and alphabets across cultures
- **Uncertainty**: Linguistic methods may introduce errors
- **Ambiguity** (synonyms)

Key concepts

- Tokenization
- Stop words
- Part-of-Speech (POS) tagging
- Stemming and lemmatization

Libraries

```
import pandas as pd
import requests

# Natural Language Toolkit
import nltk
# downloading some additional packages and corpora
nltk.download('punkt_tab') # necessary for tokenization
nltk.download('wordnet') # necessary for lemmatization
nltk.download('stopwords') # necessary for removal of stop words
nltk.download('averaged_perceptron_tagger_eng') # necessary for POS tagging
nltk.download('maxent_ne_chunker' ) # necessary for entity extraction
nltk.download('omw-1.4') # necessary for lemmatization
nltk.download('words')
```

Data

https://raw.githubusercontent.com/erickedu85/dataset/master/story.txt

```
r = requests.get("https://raw.githubusercontent.com/erickedu85/dataset/master/story.txt")
r.encoding="utf-8"

story = r.text
story
```

```
'The seventh Sally or how Trurl\'s own perfection led to no good\nBy StanisÅ‚aw Lem, 1965.\nTranslated by
Michael Kandel, 1974.\n\nThe Universe is infinite but bounded, and therefore a beam of light, in whatever
direction it may travel, will after billions of centuries return –  if powerful enough – to the point of i
ts departure; and it is no different with rumor, that flies about from star to star and makes the rounds o
f every planet. One day Trurl heard distant reports of two mighty constructor-benefactors, so wise and so
accomplished that they had no equal; with this news he ran to Klapaucius, who explained to him that these
were not mysterious rivals, but only themselves, for their fame had circumnavigated space. Fame, however,
has this fault, that it says nothing of one\'s failures, even when those very failures are the product of
a great perfection. And he who would doubt this, let him recall the last of the seven sallies of Trurl, wh
ich was undertaken without Klapaucius, whom cer…'
```

Tokenization

It is the process of breaking down a text into smaller units called **tokens.** Tokens can be words, sentences or phrases. It is often the first step in NLP task

- **Work tokenization:** splitting text into individual words
  - E.g.: the sentence "*NLP is fun*" is tokenized into `["NLP", "is", "fun"]`
- **Sentence tokenization:** splitting text into sentences
  - E.g.: the text "*I love NLP. It’s fascinating*" is split into `[”I love NLP", ”It’s fascinating"]`

Tokenization

```
from nltk import word_tokenize, pos_tag

words = word_tokenize(story)
words[:20]
```

```
['The',
 'seventh',
 'Sally',
 'or',
 'how',
 'Trurl',
 "'s",
 'own',
 'perfection',
 'led',
 'to',
 'no',
 'good',
 'By',
 'StanisÅ,aw',
 'Lem',
 ',',
 '1965',
 '.',
 'Translated']
```

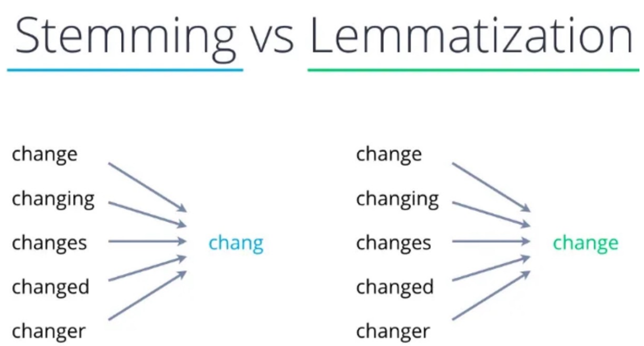
Stemming and Lemmatization

Stemming

- It is a rule-based approach to generating variants of root/base words.
- It reduces words to base words

Lemmatization

It is an evolution of *stemming* and describes the process of grouping the various inflectional forms of a word so that they can be analyzed as a single element



https://nirajbhoi.medium.com/stemming-vs-lemmatization-in-nlp-efc280d4e845

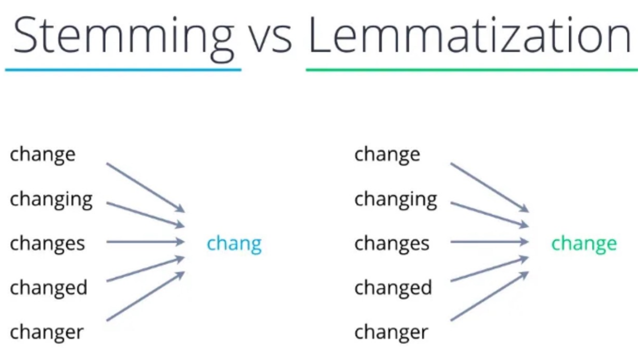
Stemming and Lemmatization

```python
from nltk.stem import PorterStemmer as stemmer
from nltk.stem import WordNetLemmatizer as lemmatizer
from nltk.corpus import wordnet # for robust lemmatization

word = "change"

print(stemmer().stem(word))
print(lemmatizer().lemmatize(word, pos = wordnet.VERB))
```

```
chang
change
```



Part-of-speech tagging (POS)

CC	Coordinating conjunction	
CD	Cardinal number	
DT	Determiner	
EX	Existential there	
FW	Foreign word	
IN	Preposition or subordinating conjunction	
JJ	Adjective	Qualities
JJR	Adjective, comparative	
JJS	Adjective, superlative	
LS	List item marker	
MD	Modal	
NN	Noun, singular or mass	Entities
NNS	Noun, plural	
NP	Proper noun, singular	
NPS	Proper noun, plural	
PDT	Predeterminer	
POS	Possessive ending	
PP	Personal pronoun	

PP$	Possessive pronoun	
RB	Adverb	
RBR	Adverb, comparative	
RBS	Adverb, superlative	
RP	Particle	
SYM	Symbol	
TO	to	
UH	Interjection	
VB	Verb, base form	Actions
VBD	Verb, past tense	
VBG	Verb, gerund or present participle	
VBN	Verb, past participle	
VBP	Verb, non-3rd person singular present	
VBZ	Verb, 3rd person singular present	
WDT	Wh-determiner	
WP	Wh-pronoun	
WP$	Possessive wh-pronoun	
WRB	Wh-adverb	

Syntactic function of a word in a sentence (noun, adjective, verb, adverb, etc.)

POS tags:
https://stackoverflow.com/questions/15388831/what-are-all-possible-pos-tags-of-nltk/38264311#38264311

Part-of-speech tagging (POS)

```
#Part-of-speech tagging

pos = pos_tag(words)
pos[:20]
```

```
[('The', 'DT'),
 ('seventh', 'JJ'),
 ('Sally', 'NNP'),
 ('or', 'CC'),
 ('how', 'WRB'),
 ('Trurl', 'NNP'),
 ("'s", 'POS'),
 ('own', 'JJ'),
 ('perfection', 'NN'),
 ('led', 'VBD'),
 ('to', 'TO'),
 ('no', 'DT'),
 ('good', 'JJ'),
 ('By', 'IN'),
 ('StanisÅ‚aw', 'NNP'),
 ('Lem', 'NNP'),
 (',', ','),
 ('1965', 'CD'),
 ('.', '.'),
 ('Translated', 'VBN')]
```

Stop Words

They are common words (such as “the”, ”is”, “and”) that are often filtered out in NLP tasks because they don’t carry significant meaning

- E.g.: in the sentence “`The dog is running`”, the stop words “`the`” and “`is`” may be removed, leaving just [“`dog`”, “`running`”]

Remove stop words

```
from nltk.corpus import stopwords as stop
stopwords = stop.words("english")
stopwords
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "y
```

```
tokens = nltk.word_tokenize(story.lower())

# tokens that contain only letters
lettertokens = [word for word in tokens if word.isalpha()]

# remove stopwords
without_stopwords = [word for word in lettertokens if word not in stopwords]
without_stopwords[:20]
```

```
['seventh',
 'sally',
 'trurl',
 'perfection',
 'led',
 'good',
 'lem',
 'translated',
 'michael',
 'kandel',
 'universe',
 'infinite',
 'bounded',
 'therefore',
 'beam',
 'light',
 'whatever',
 'direction',
 'may',
 'travel']
```

Common NLP techniques

- Bag of Words (BoW) `CountVectorizer`
- Term Frequency – Inverse Document Frequency (TF-IDF) `TfidfVectorizer`
- Word embeddings

Bag of Words (BoW)

Bag of Words (BoW) model represents text as a collection of words, ignoring grammar and word order.

- It creates a matrix where rows represent documents, and columns represent words, with each cell containing the frequency of the word in the document

BoW model

Text Corpus D

It consists of a large and structured set of texts

Corpus D

d_1	John likes to watch movies. Mary likes movies too.
d_2	Mary also likes to watch football games.
..	..

BoW model

Stop words: Common words (e.g. and, the, are, of) are removed for analysis

	John	Likes	~~To~~	Watch	Movies	Mary	~~Too~~	~~Also~~	Football	Games
$\mathbf{BoW_1}$	1	2	~~1~~	1	2	1	~~1~~	~~0~~	0	0
$\mathbf{BoW_2}$	0	1	~~1~~	1	0	1	~~0~~	~~1~~	1	1

Histogram representation (It does not preserve the order of the words in the original sentences)

	John	Likes	Watch	Movies	Mary	Football	Games
$\mathbf{BoW_1}$	1	2	1	2	1	0	0
$\mathbf{BoW_2}$	0	1	1	0	1	1	1

Term Frequency – Inverse Document Frequency (TF-IDF)

TF-IDF is a numerical statistic that is intended to reflect how important a word is to a **document** in a **corpus** D

- It increases **proportionally** to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word
- It reduces the weightage of more common words like (the, is, an etc.) which occurs in all document

Term Frequency (TF)

It is the frequency of 'any' term in a given 'document'

$$TF(t,d) = \frac{\begin{array}{c} Number\ of\ times\ a\ term\ 't' \\ is\ found\ in\ document\ 'd' \end{array}}{\begin{array}{c} Number\ of\ words \\ in\ document\ 'd' \end{array}}$$

	John	Likes	Watch	Movies	Mary	Football	Games	***Total words***
BoW_1	1	2	1	2	1	0	0	**7**
BoW_2	0	1	1	0	1	1	1	**5**

$$TF(movies, d_1) = \frac{2}{7} = 0.285$$

$$TF(movies, d_2) = \frac{0}{5} = 0$$

Inverse Document Frequency (IDF)

It measures of how much information the word provides, i.e., if it is common or rare across all documents

$$IDF(t, D) = log\left(\frac{\begin{array}{c}total\ number\ of\ documents\\ in\ the\ corpus\ 'D'\end{array}}{\begin{array}{c}number\ of\ documents\ where\\ the\ term\ 't'\ appears\end{array}}\right)$$

	John	Likes	Watch	Movies	Mary	Football	Games	*Total words*
$\mathbf{BoW_1}$	1	2	1	2	1	0	0	**7**
$\mathbf{BoW_2}$	0	1	1	0	1	1	1	**5**

$$IDF(movies, D) = log\left(\frac{2}{1}\right) = 0.301$$

TF-IDF

$$\boldsymbol{TFIDF\ (t,d,D) = TF(t,d) * IDF(t,D)}$$

	John	Likes	Watch	Movies	Mary	Football	Games	*Total words*
$\mathbf{BoW_1}$	1	2	1	2	1	0	0	**7**
$\mathbf{BoW_2}$	0	1	1	0	1	1	1	**5**

$$TFIDF\ (movies, d_1, D) = TF(movies, d_1) * IDF(movies, D)$$

$$TFIDF\ (movies, d_1, D) = 0.285 * 0.301 = 0.085785$$

$$TFIDF\ (movies, d_2, D) = TF(movies, d_2) * IDF(movies, D)$$

$$TFIDF\ (movies, d_2, D) = 0 * 0.301 = 0$$

TF-IDF

$$\boldsymbol{TFIDF\ (t, d, D) = TF(t, d) * IDF(t, D)}$$

	John	Likes	Watch	Movies	Mary	Football	Games	*Total words*
$\mathbf{BoW_1}$	1	2	1	2	1	0	0	**7**
$\mathbf{BoW_2}$	0	1	1	1	1	1	1	**6**

$$IDF(movies, D) = log\left(\frac{2}{2}\right) = 0$$

$TFIDF\ (movies, d_1, D) = TF(movies, d_1) * IDF(movies, D)$

$TFIDF\ (movies, d_1, D)$=*0.285*0=0*

$TFIDF\ (movies, d_2, D) = TF(movies, d_2) * IDF(movies, D)$

$TFIDF\ (movies, d_2, D)$=*0.166*0=0*