Introduction to Text Analysis

Jisu Kim, Ph.D

June 2024

Materials

Slides + Codes are available here:

`https://github.com/jisukimmmm/NCCR_MWQTA_2024`

What datasets are you interested in applying text analysis to?

What do you want to perform on those data?

Areas of application in social science

Examples:

- Topic Modelling in **Political Science**: Identifying and analysing themes and topics in political speeches, news articles, or social media discussions to understand public opinion and political discourse.
- Text Classification in **Sociology**: Categorising social media posts, forum discussions, or survey responses to identify patterns or trends in attitudes, behaviours, or demographics.
- Textual Analysis in **Anthropology**: Analysing ethnographic texts, interviews, or historical documents to uncover cultural norms, beliefs, and practices within a society or community.
- Text Mining in **Psychology**: Analysing text data from therapy sessions, social media posts, or online forums to study language patterns and psychological states such as stress, depression, or personality traits.

Areas of application in private companies

- Customer Feedback Analysis: Analysing customer reviews and feedback to understand customer sentiments, preferences, and areas for improvement.
- Market Research: Analysing social media discussions, customer surveys, and online forums to identify market trends, consumer preferences, and competitor analysis.
- Brand Monitoring: Tracking mentions of the company brand or products on social media, news websites, and online forums to monitor brand reputation and sentiment.
- Customer Support: Automating responses to customer inquiries and complaints using chatbots and natural language processing to provide timely and personalised support.

Useful materials

- Sarkar, D. (2016). Text analytics with python (Vol. 2). New York, NY, USA:: Apress.
- Bengfort, B., Bilbro, R., Ojeda, T. (2018). Applied text analysis with Python: Enabling language-aware data products with machine learning. " O'Reilly Media, Inc.".
- Bird, S., Klein, E., Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc.".

Challenges in Text Analysis

- Unstructured data
- Noise in text data
- Ambiguity and context understanding
- Scalability issues due to large volumes of text data

Useful platforms

For data + codes=

- Kaggle (`https://www.kaggle.com/datasets`)
- Hugging Face (`https://huggingface.co/`)
- Google colab (`https://colab.google/`

For data=

- openICPSR (`https://www.openicpsr.org/openicpsr/`)
- Harvard Dataverse (`https://dataverse.harvard.edu/`)

Relevant tools and libraries on Python

- NLTK (Natural Language Toolkit)[1]
- spaCy[2]
- Gensim[3]
- Sentence Transformers[4]
- ...

[1] https://www.nltk.org/
[2] https://spacy.io/
[3] https://pypi.org/project/gensim/
[4] https://sbert.net/

NLTK (Natural Language Toolkit)

- **Purpose**: NLTK is a comprehensive library for NLP tasks, ranging from tokenization and stemming to parsing and semantic reasoning. It is widely used for education, research, and prototyping.
- **Features**: NLTK provides a wide range of tools and algorithms for various NLP tasks, including tokenization, stemming, lemmatization, part-of-speech tagging, named entity recognition, parsing, sentiment analysis, and more.
- **Flexibility**: NLTK offers a lot of flexibility and customisation options, allowing users to build and experiment with their own NLP pipelines.

spaCy

- **Purpose**: spaCy is designed for industrial-strength NLP tasks and focuses on performance and ease of use. It is optimized for speed and efficiency, making it suitable for processing large volumes of text.
- **Features**: spaCy provides fast and efficient implementations of common NLP tasks, including tokenization, part-of-speech tagging, named entity recognition, dependency parsing, and sentence segmentation. It also includes pre-trained models for many languages.
- **Performance**: spaCy is known for its speed and efficiency, making it well-suited for real-world applications where performance is crucial.

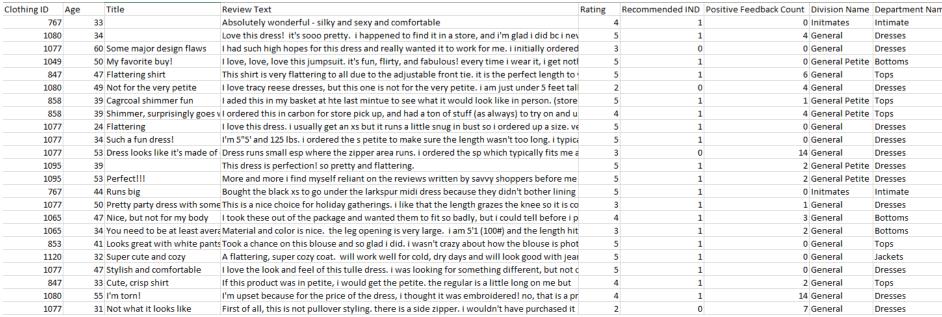
Gensim

- **Purpose**: Gensim is a library for topic modeling and document similarity analysis. It is particularly well-suited for unsupervised learning tasks such as latent semantic analysis (LSA), latent Dirichlet allocation (LDA), and word embeddings.
- **Features**: Gensim provides efficient implementations of algorithms for topic modeling, document similarity analysis, word embeddings (e.g., Word2Vec, Doc2Vec), and other unsupervised learning tasks.
- **Scalability**: Gensim is designed for scalability and can handle large text corpora efficiently. It also provides distributed computing support for training models on multiple cores or machines.
- **Focus**: While NLTK and spaCy cover a wide range of NLP tasks, Gensim is specialized for specific tasks related to topic modeling and unsupervised learning.

Useful keywords

- **Natural Language Processing (NLP)**: The branch of artificial intelligence concerned with the interaction between computers and humans through natural language. NLP enables computers to understand, interpret, and generate human language in a manner that is both meaningful and useful.
- **Corpus**: A collection of texts or documents, typically used for linguistic analysis, language modelling, or training natural language processing algorithms. Corpora can range from small, specialised collections to large, comprehensive datasets containing a wide variety of texts.
- **Unstructured Data**: Data that does not have a predefined data model or organisation, making it more difficult to analyse using traditional methods. Unstructured data includes text documents, audio recordings, images, videos, and other forms of data that lack a structured format.
- **Information Extraction**: The process of automatically extracting structured information from unstructured text data. Information extraction techniques identify and extract specific types of information, such as entities, relationships, and events, from text documents to enable further analysis or processing.

Example of text data



Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Nam
767	33		Absolutely wonderful - silky and sexy and comfortable	4	1	0	Initmates	Intimate
1080	34		Love this dress! it's sooo pretty. i happened to find it in a store, and i'm glad i did bc i nev	5	1	4	General	Dresses
1077	60	Some major design flaws	I had such high hopes for this dress and really wanted it to work for me. i initially ordered	3	0	0	General	Dresses
1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, flirty, and fabulous! every time i wear it, i get noth	5	1	0	General Petite	Bottoms
847	47	Flattering shirt	This shirt is very flattering to all due to the adjustable front tie. it is the perfect length to	5	1	6	General	Tops
1080	49	Not for the very petite	I love tracy reese dresses, but this one is not for the very petite. i am just under 5 feet tall	2	0	4	General	Dresses
858	39	Cagrcoal shimmer fun	I aded this in my basket at hte last mintue to see what it would look like in person. (store	5	1	1	General Petite	Tops
858	39	Shimmer, surprisingly goes v	I ordered this in carbon for store pick up, and had a ton of stuff (as always) to try on and u	4	1	4	General Petite	Tops
1077	24	Flattering	I love this dress. i usually get an xs but it runs a little snug in bust so i ordered up a size. ve	5	1	0	General	Dresses
1077	34	Such a fun dress!	I'm 5"5' and 125 lbs. i ordered the s petite to make sure the length wasn't too long. i typica	5	1	0	General	Dresses
1077	53	Dress looks like it's made of	Dress runs small esp where the zipper area runs. i ordered the sp which typically fits me a	3	0	14	General	Dresses
1095	39		This dress is perfection! so pretty and flattering.	5	1	2	General Petite	Dresses
1095	53	Perfect!!!	More and more i find myself reliant on the reviews written by savvy shoppers before me	5	1	2	General Petite	Dresses
767	44	Runs big	Bought the black xs to go under the larkspur midi dress because they didn't bother lining	5	1	0	Initmates	Intimate
1077	50	Pretty party dress with some	This is a nice choice for holiday gatherings. i like that the length grazes the knee so it is co	3	1	1	General	Dresses
1065	47	Nice, but not for my body	I took these out of the package and wanted them to fit so badly, but i could tell before i p	4	1	3	General	Bottoms
1065	34	You need to be at least avera	Material and color is nice. the leg opening is very large. i am 5'1 (100#) and the length hit	3	1	2	General	Bottoms
853	41	Looks great with white pants	Took a chance on this blouse and so glad i did. i wasn't crazy about how the blouse is phot	5	1	0	General	Tops
1120	32	Super cute and cozy	A flattering, super cozy coat. will work well for cold, dry days and will look good with jear	5	1	0	General	Jackets
1077	47	Stylish and comfortable	I love the look and feel of this tulle dress. i was looking for something different, but not c	5	1	0	General	Dresses
847	33	Cute, crisp shirt	If this product was in petite, i would get the petite. the regular is a little long on me but	4	1	2	General	Tops
1080	55	I'm torn!	I'm upset because for the price of the dress, i thought it was embroidered! no, that is a pr	4	1	14	General	Dresses
1077	31	Not what it looks like	First of all, this is not pullover styling. there is a side zipper. i wouldn't have purchased it	2	0	7	General	Dresses

Figure: Women's clothing E-commerce reviews[5]

[5] https://www.kaggle.com/datasets/

How computer sees the data

Texts are stored as strings, i.e., a sequence of characters:

```
In : text = ‘My favourite buy!’
In : text
Out: ‘My favourite buy!”
In : text[0]
Out: ‘M’
In : len(text)
Out: 17
```

Encodings

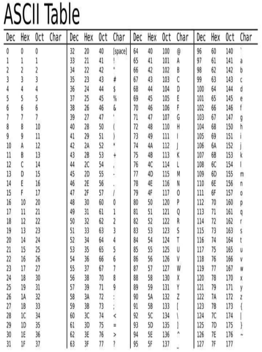
An encoding is a method or scheme used to represent characters, symbols, or text data in a form that can be stored, transmitted, or processed by a computer system. It defines a mapping between the characters of a language and their binary representations, allowing computers to understand and manipulate textual information.

In simpler terms, encoding translates human-readable text into a format that computers can understand and work with. It involves converting characters, such as letters, numbers, and symbols, into binary code (sequences of 0s and 1s), which computers can process and store.

Encodings

A **character encoding** scheme maps characters to their binary representations.

ASCII (American Standard Code for Information Interchange) is a widely-used character encoding standard that represents characters using 7 bits, providing mappings for 128 characters, including letters, digits, punctuation marks, and control characters. Extended ASCII extends the ASCII encoding to use 8 bits, allowing for an additional 128 characters, including accented letters and symbols.



ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

https://dev.to/

Encodings

Unicode is a standard for encoding characters from all writing systems in the world, encompassing a vast range of characters including letters, digits, symbols, and emojis.

UTF-8 (Unicode Transformation Format 8-bit) is a variable-width character encoding that uses one to four bytes to represent each character. It's the most commonly used Unicode encoding on the web because it's backward compatible[6] with ASCII and more efficient for English text.

[6] compatible with earlier versions or standards.

Encodings

UTF-16 and UTF-32 are fixed-width encodings that use two and four bytes per character, respectively. They're less commonly used than UTF-8 but offer advantages for languages with large character sets.

Byte Order Mark (BOM): BOM is a special marker added to the beginning of a text file to indicate the byte order (endianness) of the file's content. It's primarily used with UTF-16 and UTF-32 encodings to distinguish between little-endian and big-endian byte order.
UTF-8 doesn't require a BOM because its byte order is always the same, making it unnecessary and sometimes even discouraged in UTF-8 encoded files.

Encodings

Encoding Conversion: refers to the process of converting text data from one encoding to another. This is often necessary when dealing with text data that uses different encodings, or when working with systems that expect specific encodings. Python provides libraries such as *codec*[7] and *chardet*[8] for encoding detection and conversion.

[7] `https://docs.python.org/3/library/codecs.html`

[8] `https://pypi.org/project/chardet/`

Why Encodings?

Improper encoding can cause I/O errors and data loss:

```
Traceback (most recent call last):
File "text_indexing.py", line 34, inmodulemain()
File "text_indexing.py", line 54, inmainforrowinreader
File "codecs.py", line 321, in decode
(result, consumed) = self.buffer_decode(data, self.errors, final)
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xf3 in position 234:
invalid continuation byte
```

$\rightarrow$ *L'Universit?deNeuch?tel*

Strings in Python

In Python 2, strings (str) are treated as byte sequences, while Unicode is considered a separate data type.

In contrast, Python 3 treats all strings as Unicode, offering encoding/decoding methods to obtain byte representations when needed, such as for serialisation purposes[9].

This is one of the key advantages of 3 over 2.

[9] Serialisation refers to the process of converting data structures or objects into a format that can be easily stored, transmitted, or reconstructed later.

Text Preprocessing-First steps

- Lowercasing
- Removing Punctuation
- Removing Stopwords
- Removing Numbers and Special Characters
- Handling Contractions
- Spell Checking and Correction

Lowercasing

Definition: Converting all text to lowercase.
Purpose: Ensure consistency in word representation.
Example: "Hello World" $>$ "hello world".

Removing Punctuation

Definition: Eliminating punctuation marks from text.
Purpose: Reduce noise and simplify text.
Example: "Hello, world!" $>$ "Hello world".

Removing Stopwords

Definition: Filtering out common words that do not contribute much to the meaning of the text.

Examples: "the", "is", "and", "of". Purpose: Improve computational efficiency and focus on meaningful content.

Removing Numbers and Special Characters

Deleting numerical digits and non-alphanumeric characters.
Purpose: Remove noise and irrelevant information.
Example: "@World123" $>$ "World"

Tweets and Texts: Replace emojis and emoticons with relevant words.
Example: :-) $>$ Smiley face

Handling Contractions

Definition: Expanding contracted forms of words.
Example: "can't" $>$ "cannot", "I've" $>$ "I have".

Spell Checking and Correction

Identifying and rectifying misspelled words.
Example: **Rule-based Correction**, Implementing rules to correct misspellings based on common patterns of errors. For example, replacing "teh" with "the" or "mispeling" with "misspelling".

Further Text Preprocessing Steps

- Tokenization
- Regular expressions
- Normalisation
- Stemming
- Lemmatization
- Named Entity Recognition (NER)

Tokenization

- Definition: Breaking text into words, phrases, symbols, or other meaningful elements.
- Purpose: It helps in organising the text into a structured format that can be easily manipulated and analysed by algorithms and models in natural language processing tasks.
- Techniques:
    - Word Tokenization: Splitting text into individual words, suitable for word-level analysis.
    - Sentence Tokenization: Dividing text into sentences, often using punctuation cues or specialised models.
    - ...

Regular expressions

Regular expressions are a powerful tools to tokenize text. Python's regular expressions are provided by the 're' package.
Match a specific string : 'word'
Match any four characters word: `[a-z]{4}`
Match any sequence of digits : `[09]+`
Match any email address: `([^@]+@[^@]+[^@]+)`

Normalization

- Definition: The process of converting text into a standard or normalized form. This typically involves converting text to lowercase, removing accents or diacritics, and handling special characters or symbols in a consistent manner.
- Purpose: To ensure that different variations of words or phrases are treated the same way, making it easier for downstream text analysis tasks like tokenization and feature extraction. Normalization helps improve the accuracy and effectiveness of NLP algorithms by reducing the complexity of the text data and ensuring uniformity in representation.
- Example: "I'm going to the café to meet my friend José." $>$ "im going to the cafe to meet my friend jose."

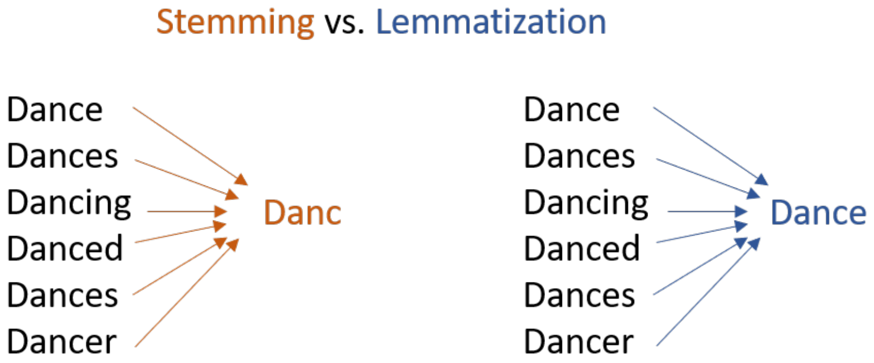
Stemming

- Definition: a rule-based technique that removes suffixes from words to extract their root form, known as the stem.
- Purpose: The goal of stemming is to reduce words to a common base form, even if the resulting stem may not be a valid word itself. Stemming algorithms apply heuristic rules to chop off common suffixes, such as "-ing," "-ly," "-es," etc., to obtain the stem. Stemming is typically faster and less computationally intensive than lemmatization, but it may produce stems that are not actual words, leading to potential loss of meaning.
- E.g., Changing $>$ chang

Lemmatization

- Definition: It aims to reduce words to their base or dictionary form, known as the lemma. Unlike stemming, lemmatization considers the context of the word and its part of speech (POS) to determine the lemma, resulting in valid dictionary words. Lemmatization algorithms use dictionary lookup to find the lemma of a word, considering factors such as inflection, tense, case, and POS.
- Purpose: It produces more accurate results compared to stemming because it considers the linguistic meaning of words, but it is usually slower and requires access to a comprehensive dictionary or lexicon.
- E.g., Eats $>$ eat

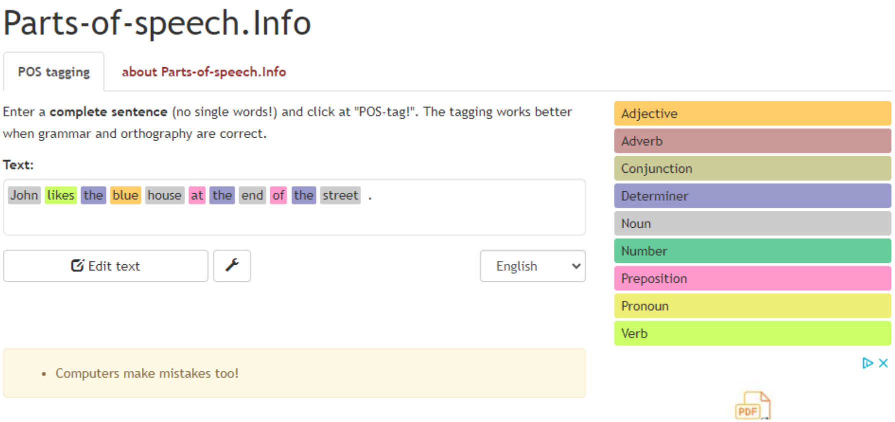
Stemming and Lemmatization



Part-of-Speech (POS) Tagging

- Definition: It involves assigning a specific part-of-speech tag to each word in a given text corpus, based on its syntactic role and context within the sentence. Parts of speech include nouns, verbs, adjectives, adverbs, pronouns, conjunctions, prepositions, and interjections.
- Purpose: To analyse the structure of sentences and understand how words function within them. By identifying the part of speech of each word, NLP systems can perform various tasks more accurately, such as syntactic parsing, sentiment analysis, named entity recognition, and machine translation.

POS[10]



Parts-of-speech.Info

POS tagging | about Parts-of-speech.Info

Enter a **complete sentence** (no single words!) and click at "POS-tag!". The tagging works better when grammar and orthography are correct.

Text:

John likes the blue house at the end of the street .

Edit text | English

- Computers make mistakes too!

Adjective
Adverb
Conjunction
Determiner
Noun
Number
Preposition
Pronoun
Verb

[10]https://parts-of-speech.info/

Named Entity Recognition (NER)

Named Entity Recognition is a technique used to identify and classify named entities within unstructured text into predefined categories such as names of persons, organizations, locations, dates, numerical expressions, and more.

For example, given the sentence "Apple is headquartered in Cupertino, California, and was founded by Steve Jobs, Steve Wozniak, and Ronald Wayne on April 1, 1976," NER would identify and classify the following named entities:

- Organization: "Apple"
- Location: "Cupertino, California"
- Persons: "Steve Jobs," "Steve Wozniak," "Ronald Wayne"
- Date: "April 1, 1976"

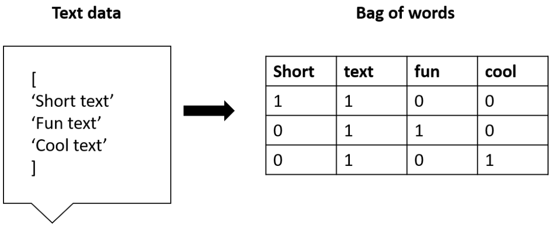
Feature Engineering

It involves transforming raw text data into numerical or categorical features that can be used as input for machine learning algorithms.

- Bag of Words (BoW) representation
- TF-IDF (Term Frequency-Inverse Document Frequency)
- Word Embeddings
- N-grams
- Text Length and Complexity
- Collocation and Terminology extraction

Bag of Words (BoW)

BoW represents text data as a matrix where each row corresponds to a document and each column corresponds to a unique word in the corpus. The values in the matrix indicate the frequency of each word in the document. This approach disregards word order and grammar but captures the occurrence of words in documents.



Short	text	fun	cool
1	1	0	0
0	1	1	0
0	1	0	1

TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF calculates the importance of a word in a document relative to a corpus of documents. It assigns higher weights to words that are frequent in a document but rare in other documents. TF-IDF is useful for identifying keywords and reducing the importance of common words like "the" and "is".

$$TF_{(t,d)} = \frac{\#\text{ of occurrences of term, } t \text{ in document, } d}{\text{Total } \#\text{ of terms in the document, } d} \tag{1}$$

$$IDF_{(t,D)} = log_e \frac{\text{Total } \#\text{ of documents in the corpus}}{\#\text{ of documents with term, } t \text{ in them}} \tag{2}$$

$$TFIDF_{(t,d,D)} = TF_{(t,d)} \times IDF_{(t,D)} \tag{3}$$

Term Frequency (TF)

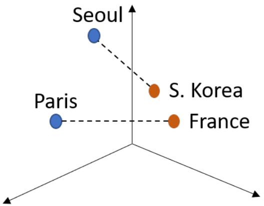
TF measures how often a particular word appears in a document relative to the total number of words in that document. In other words, It calculates the frequency of occurrence of each word in a document. Words that occur more frequently are typically considered more important in representing the content or theme of the document.

Inverse Document Frequency (IDF)

IDF measures the rarity of a word across all documents in the corpus. Words that occur in fewer documents are considered more important or informative because they help distinguish one document from another.

Word Embeddings

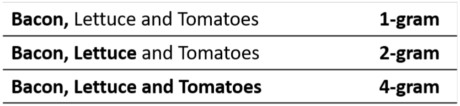
Word embeddings represent words as dense vector representations in a continuous vector space. It learns distributed representations of words based on their context in a large corpus of text. Word embeddings capture semantic relationships between words and are useful for tasks like semantic similarity and text classification.



More to be learned tomorrow...

N-grams

N-grams are sequences of N consecutive words in a document. Unigrams (N=1) represent single words, bigrams (N=2) represent pairs of adjacent words, and trigrams (N=3) represent triplets of adjacent words. N-grams capture local word patterns and can provide additional context for text analysis tasks.



Bacon, Lettuce and Tomatoes	**1-gram**
Bacon, Lettuce and Tomatoes	**2-gram**
Bacon, Lettuce and Tomatoes	**4-gram**

Text Length and Complexity

Features such as the length of the text, average word length, and vocabulary richness (number of unique words) can provide insights into the complexity and style of the text. These features can be useful for tasks like readability assessment and authorship attribution.

Collocation and Terminology Extraction

Collocations are pairs or groups of words that often co-occur together in a text. They carry a specific meaning and are commonly used in language. Example: "strong coffee," "heavy rain," "fast food."