# Feature Extraction

Type of Tokens

1. Word tokens
2. Character tokens
3. Sentence tokens
4. Named entity tokens
5. Part-of-speech (POS) tags
6. Sub-word tokens

Text Preprocessing

Stemming and Lemmatization are Text Normalization (or sometimes called Word Normalization) techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing. Stemming and Lemmatization have been studied, and algorithms have been developed in Computer Science since the 1960s. We will learn about Stemming and Lemmatization in a practical approach covering the background, some famous algorithms, applications of Stemming and Lemmatization, and how to stem and lemmatize words, sentences, and documents using the Python **nltk package** which is the Natural Language Tool Kit package provided by Python for Natural Language Processing tasks.

Text Preprocessing
Stemming

Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and natural language processing (NLP). When a new word is found, it can present new research opportunities. For example -
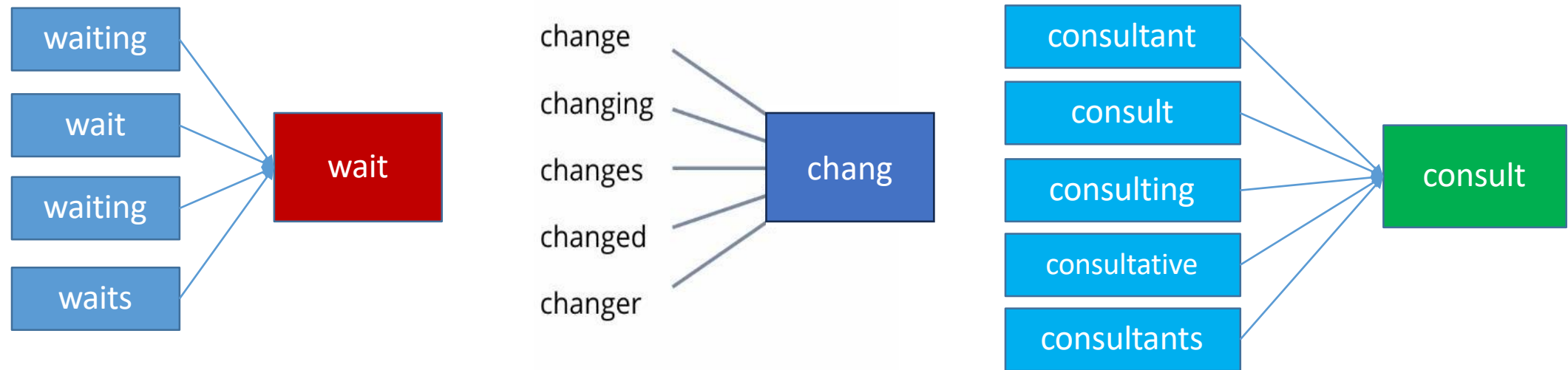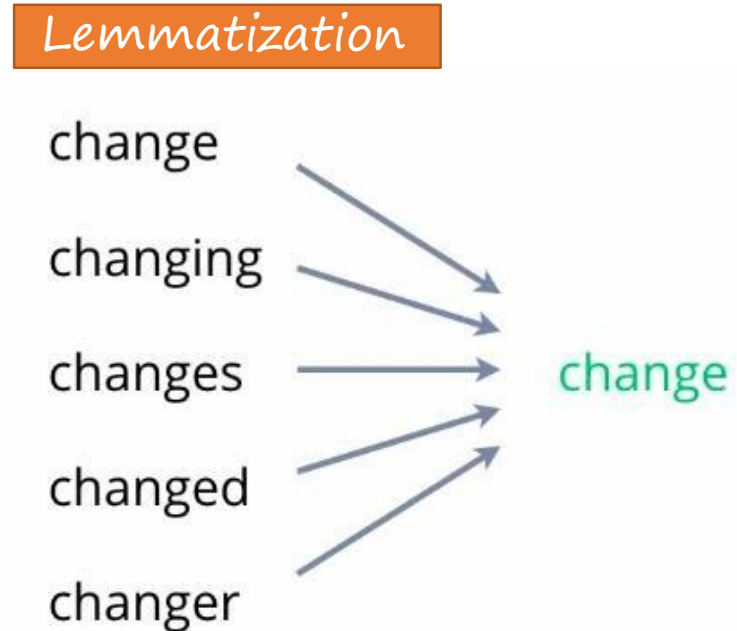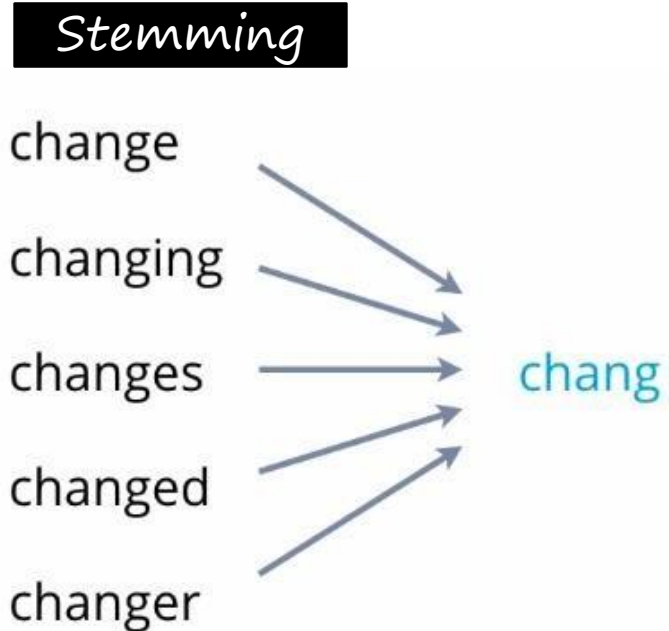
- **Porter Stemmer():** The Porter stemming algorithm (or 'Porter stemmer') removes the commoner morphological endings from words in English.
- **Lovins Stemmer**
- **Dawson Stemmer**
- **Krovetz Stemmer**
- **Xerox Stemmer**
- **N-Gram Stemmer**
- **Snowball Stemmer**
- **Lancaster Stemmer**

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

- Word Net Lemmatizer
- Spacy Lemmatizer
- TextBlob
- Gensim Lemmatizer
- TreeTagger

studying
studies
study

Lemmatization →

study
study
study

	original_word	lemmatized_word
0	trouble	trouble
1	troubling	trouble
2	troubled	trouble
3	troubles	trouble

Stemming

change
changing
changes
changed
changer
→ chang

Lemmatization

change
changing
changes
changed
changer
→ change
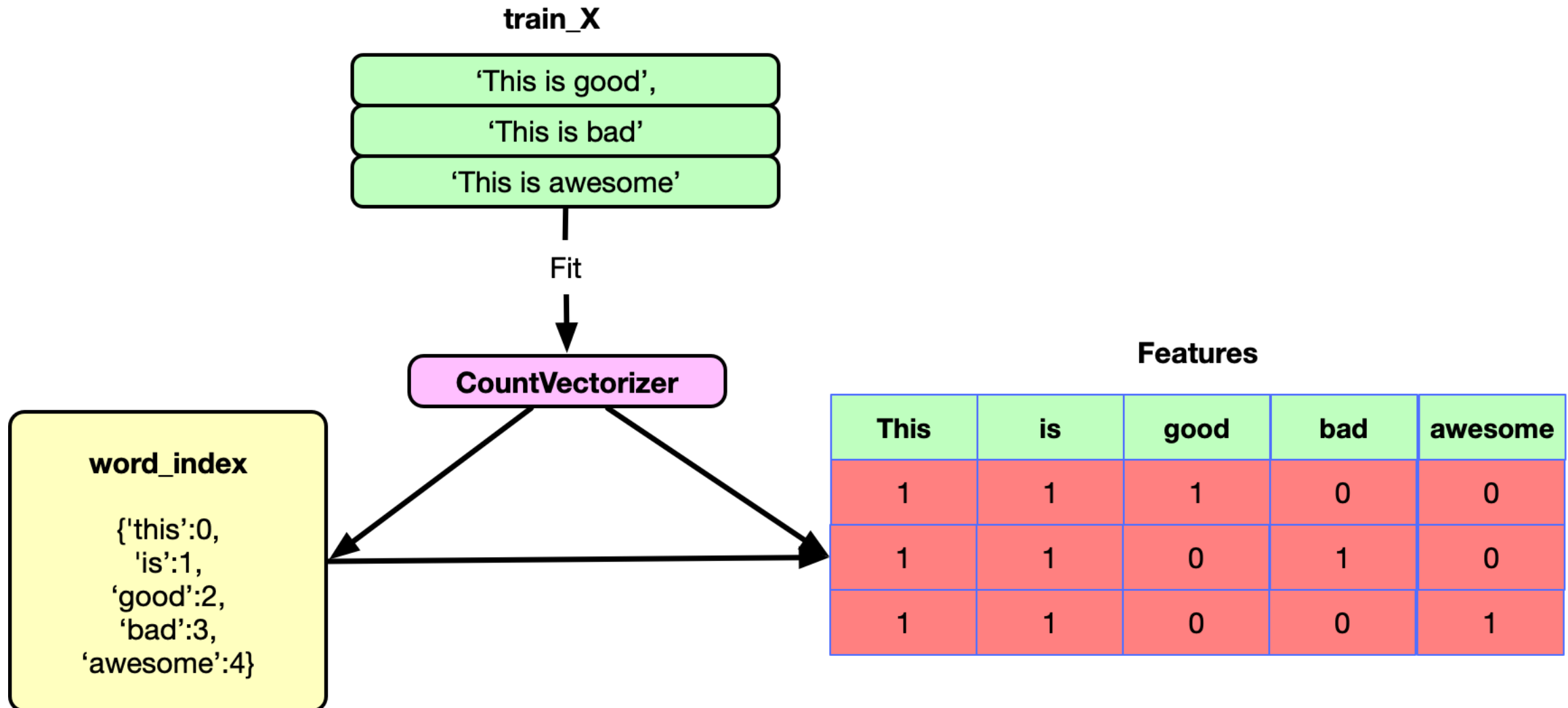
Vectorizer!

www.aiquest.org

Feature transformation: Transformation of data to improve the accuracy of the algorithm.

Feature selection: Removing unnecessary features.

Feature extraction: The transformation of raw data into features suitable for modeling.

Types of Vectorizer

- Bag of Words: Count Vectorizer

- TF-IDF Vectorizer

- Word2Vec

- Global Vectors for Word Representation

Count Vectorizer
Mathematical architecture

train_X

'This is good',
'This is bad'
'This is awesome'

Fit

CountVectorizer

word_index

{'this':0,
'is':1,
'good':2,
'bad':3,
'awesome':4}

Features

This	is	good	bad	awesome
1	1	1	0	0
1	1	0	1	0
1	1	0	0	1

TF-IDF Vectorizer
Mathematical architecture

train_X

'This is good and awesome',

'This is bad'

Fit

TFIDFVectorizer

Term Frequency
No of time word appears/No of total terms in Document

This	is	good	bad	awesome	and
1/5	1/5	1/5	0	1/5	1/5
1/3	1/3	0	1/3	0	0

Inverse Document Frequency
-log(ratio of documents that include the word)

This	is	good	bad	awesome	and
log(2/2)	log(2/2)	log(2/1)	log(2/1)	log(2/1)	log(2/1)

X

Features

This	is	good	bad	awesome	and
0	0	1/5*log(2/1)	0	1/5*log(2/1)	1/5*log(2/1)
0	0	0	1/3*log(2/1)	0	0

- TF-IDF is better than Count Vectorizers because it not only focuses on the frequency of words present in the corpus but also provides the importance of the words. The term "df" is called document frequency which means in how many documents the word "subfield" is present within the corpus.

  - ❑ Can TF IDF Be Negative?

  - ❑ No. The lowest value is 0. Both term frequency and inverse document frequency are positive numbers.

- In AI inference and machine learning, sparsity refers to a matrix of numbers that includes many zeros or values that

- will not significantly impact a calculation.

Word2Vec

Word2vec is a Neural Network that processes text by "vectorizing" words. Its input is a text corpus and its output is a set of vectors: feature vectors that represent words in that corpus.
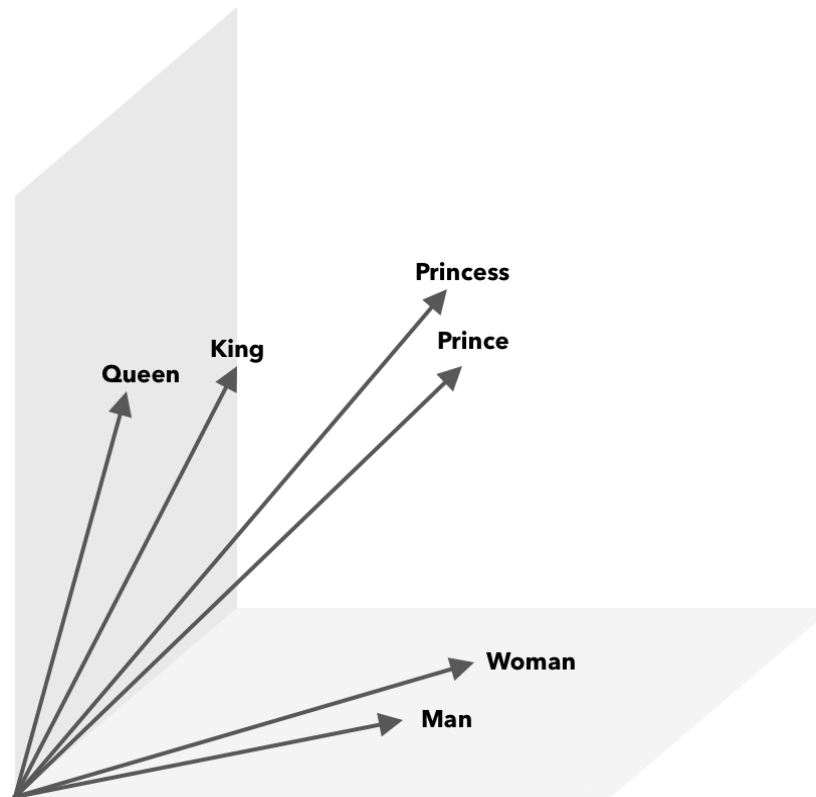
- King – Man + Women = Queen

- Prince + mom = Queen

- vec("king") - vec("man") + vec("woman") =~ vec("queen")