Module

Introduction to Natural Language Processing

Ajvad Haneef
Empanelled Trainer (AIML/DS) – [Parttime]
Senior Research Fellow, NIT Calicut

Experience:

- **Assistant Professor (Adhoc), CSE, Govt. Engg. College, Wayanad**
- **Assistant Professor, CSE, Vimal Jyothi Engg. College, Chemperi**
- **Software Engineer, Trogon Media Pvt. Ltd.**

What is Natural Language Processing?

Natural Language Processing (NLP) enables computers to understand and interpret human language

- Text analysis and entity recognition
- Sentiment analysis
- Speech recognition and synthesis
- Machine translation
- Semantic language modeling

Natural Language Processing

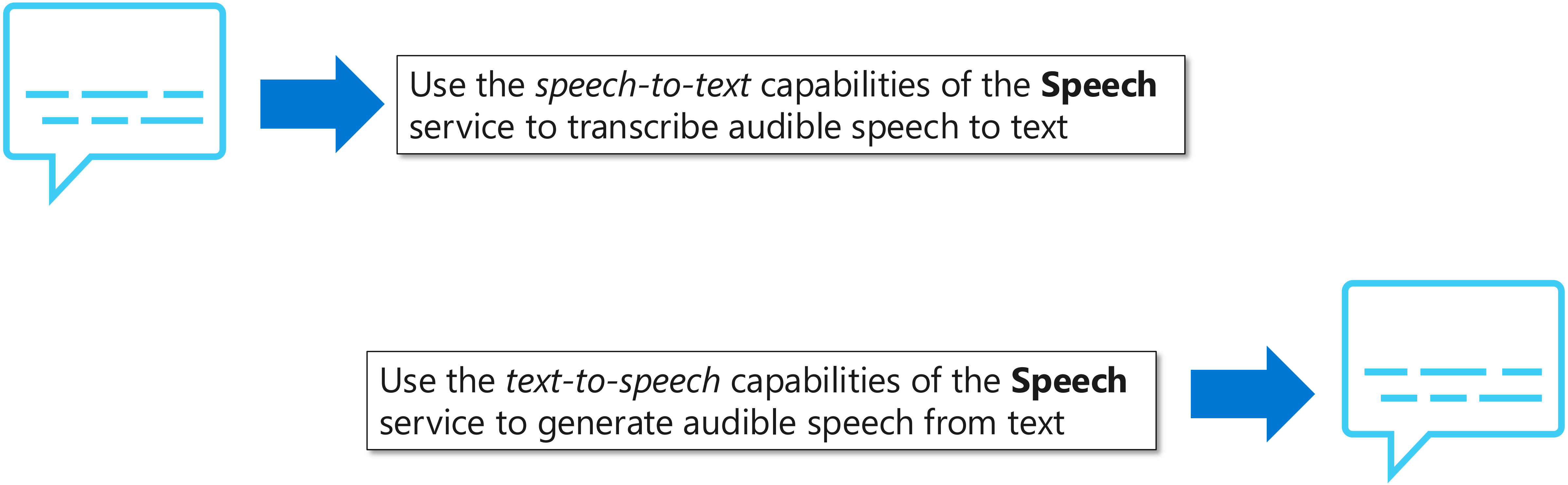
Text Analytics	• Language detection • Key phrase extraction • Entity detection • Sentiment analysis
Speech	• Text to speech • Speech to text • Speech translation
Translator Text	• Text translation
Language Understanding	• Custom language modeling

Text Analytics

I had a wonderful vacation in France.

- Predominant Language: English
- Sentiment: 88% (positive)
- Key Phrases: "wonderful vacation"
- Entities: France

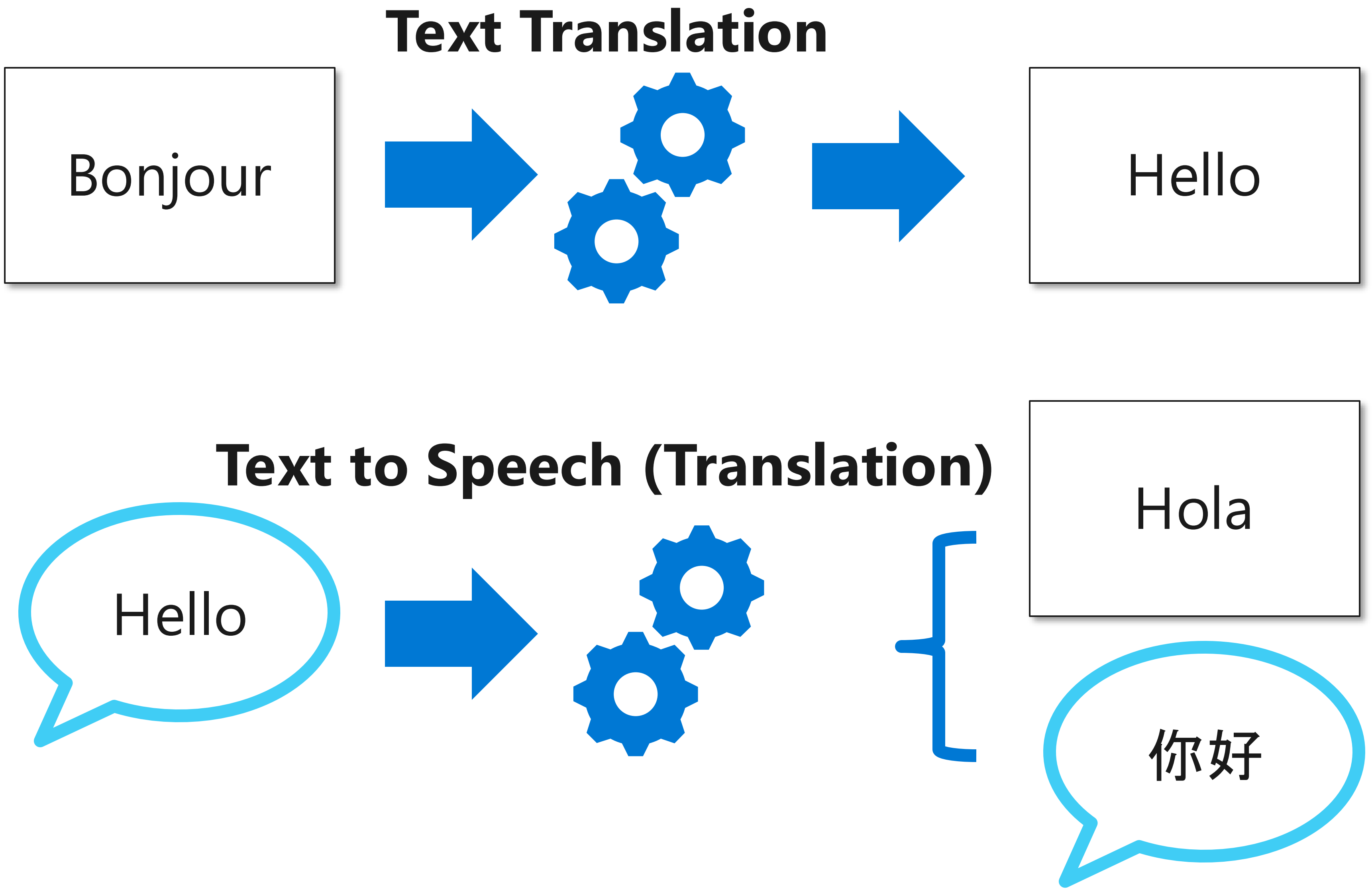
Speech Recognition and Synthesis



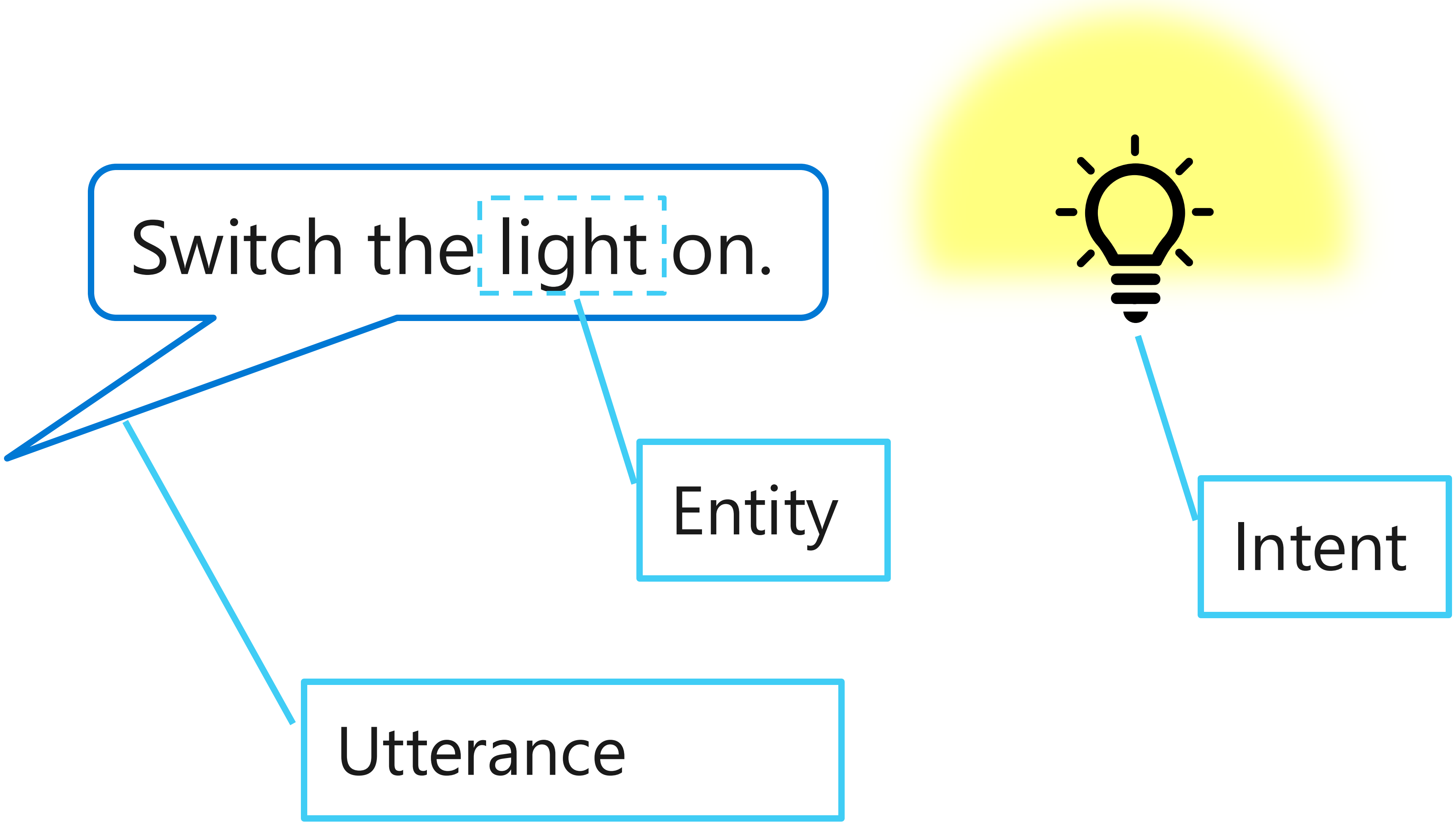
Use the *speech-to-text* capabilities of the **Speech** service to transcribe audible speech to text

Use the *text-to-speech* capabilities of the **Speech** service to generate audible speech from text

Translation

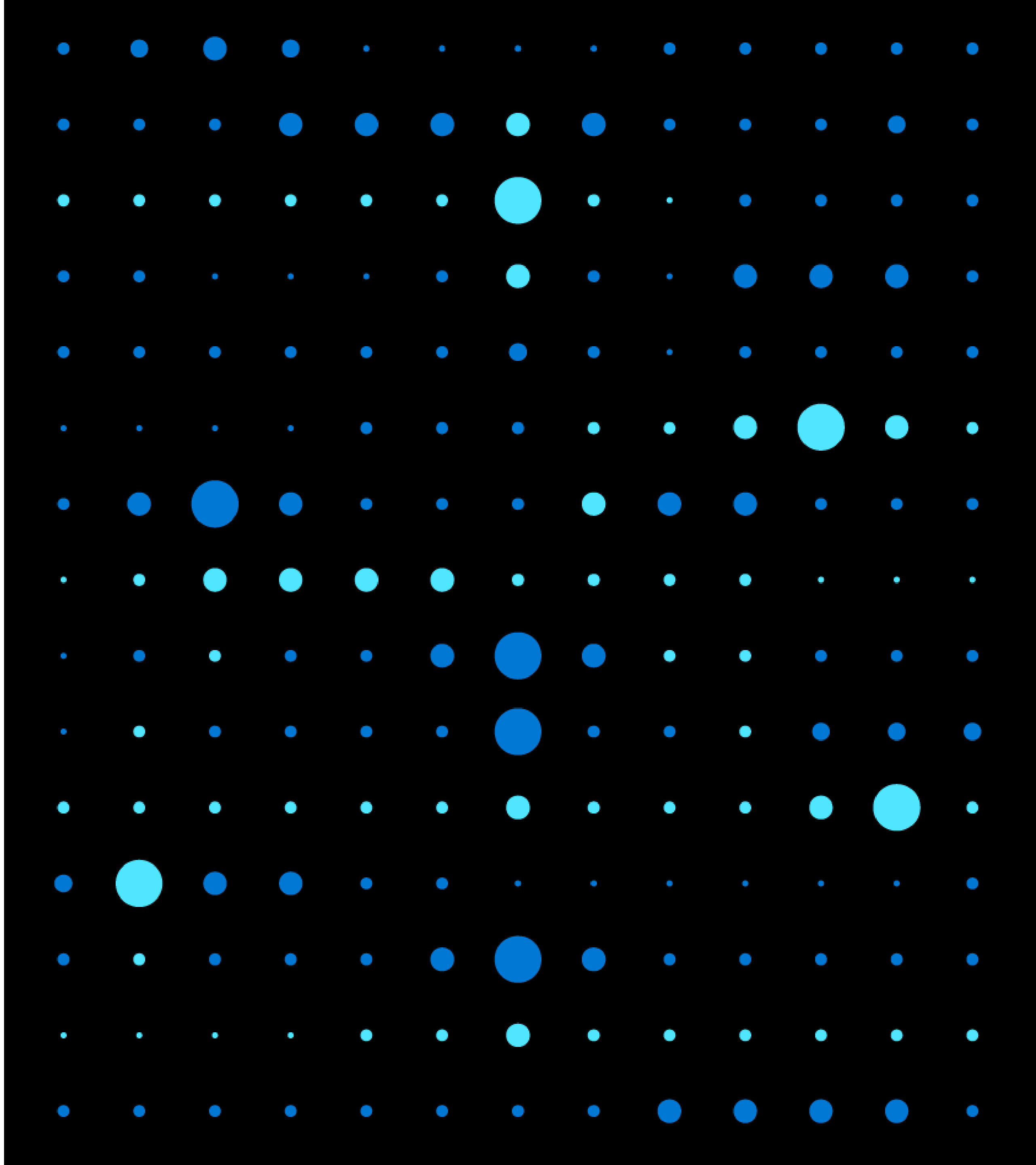


Language Understanding

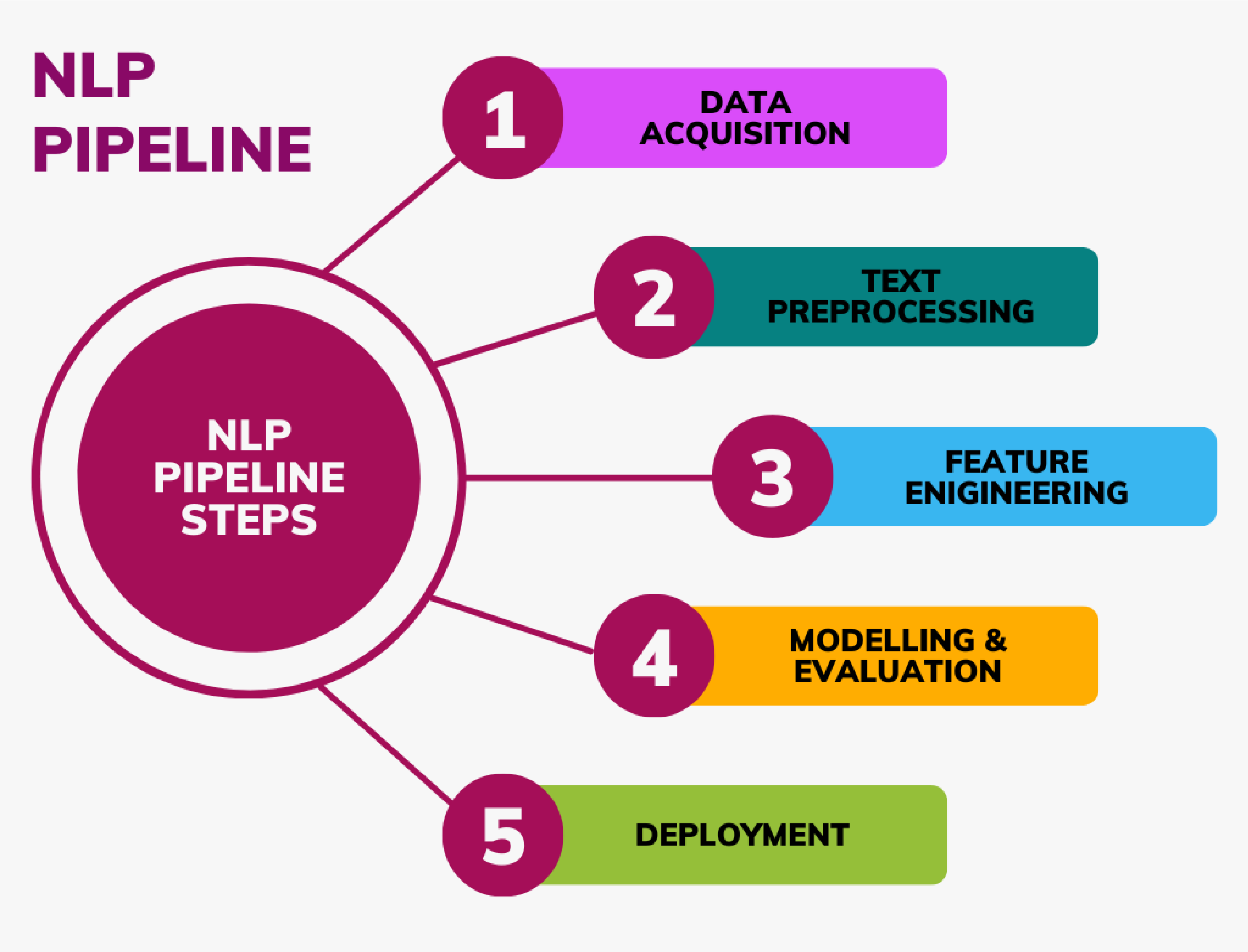


Demo

Natural Language Processing



NLP Pipeline



NLP PIPELINE

NLP PIPELINE STEPS

1 DATA ACQUISITION

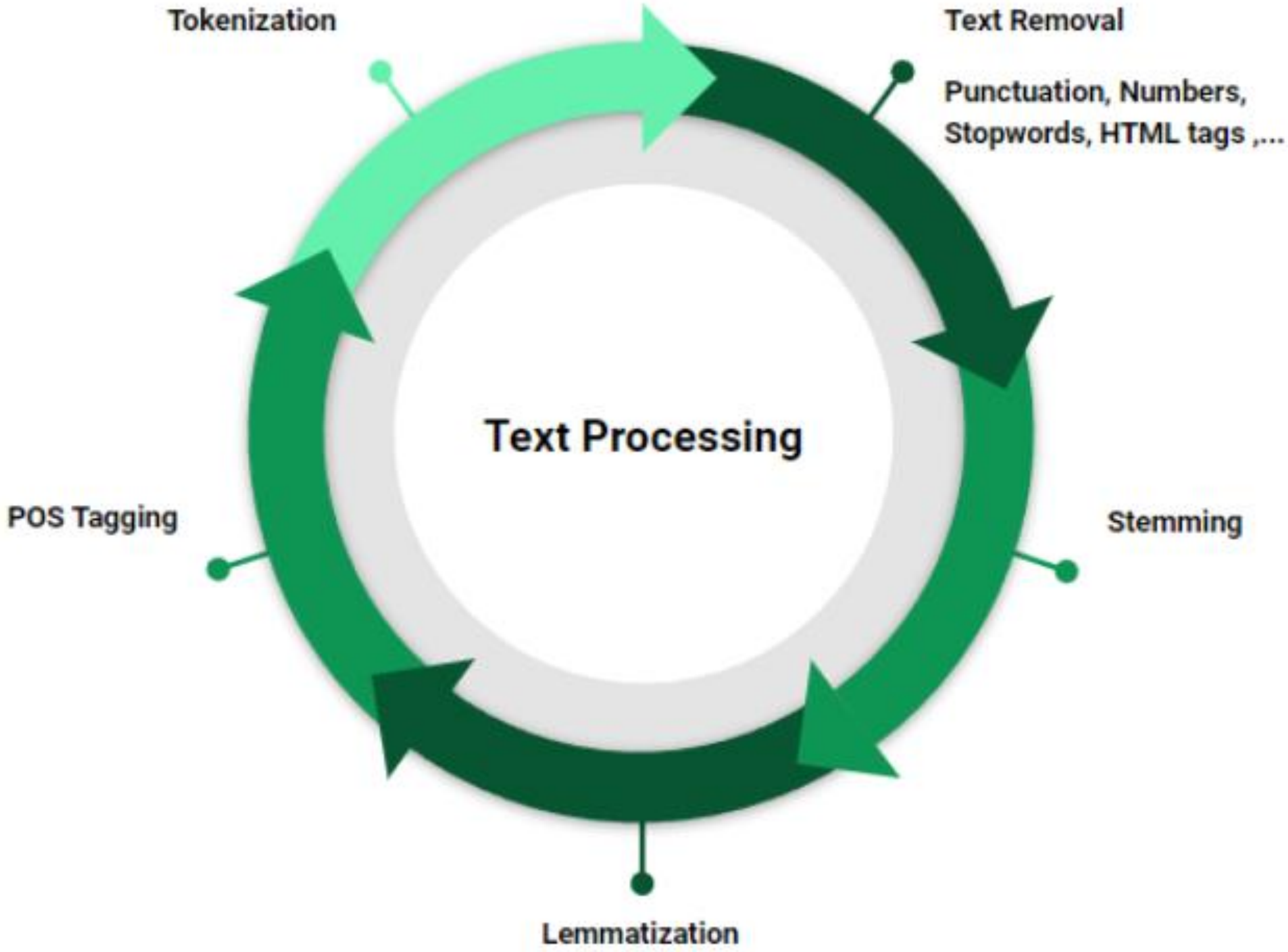
2 TEXT PREPROCESSING

3 FEATURE ENIGINEERING

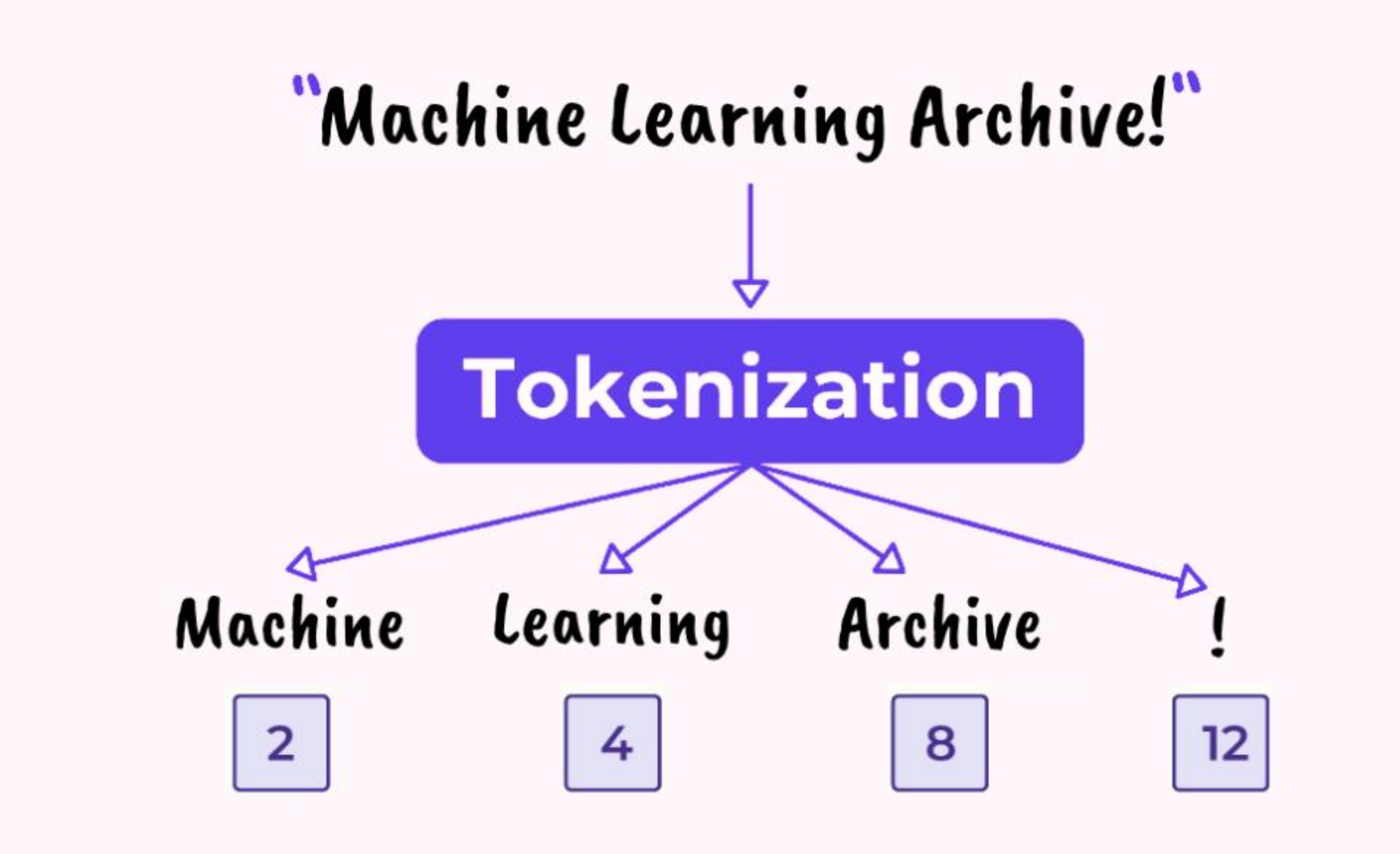
4 MODELLING & EVALUATION

5 DEPLOYMENT

Text Preprocessing

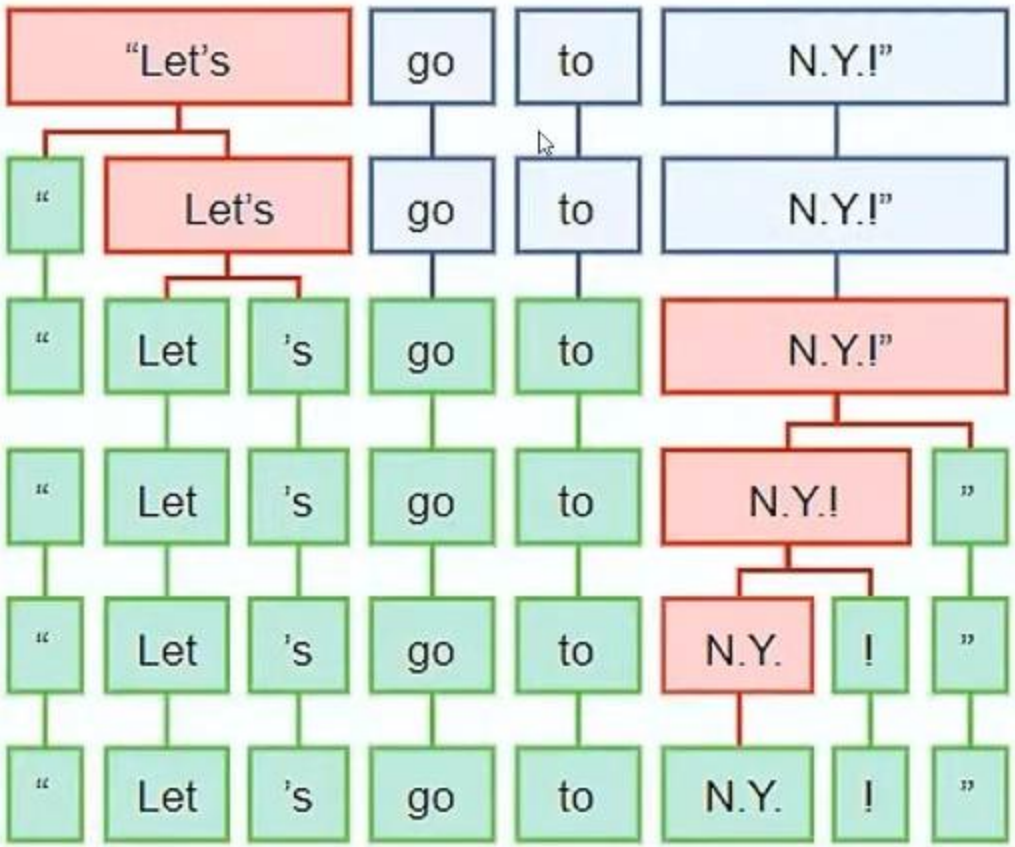


Tokenization



Tokenization

"A process of breaking up text into many small pieces. Works by separating words using spaces & punctuation"



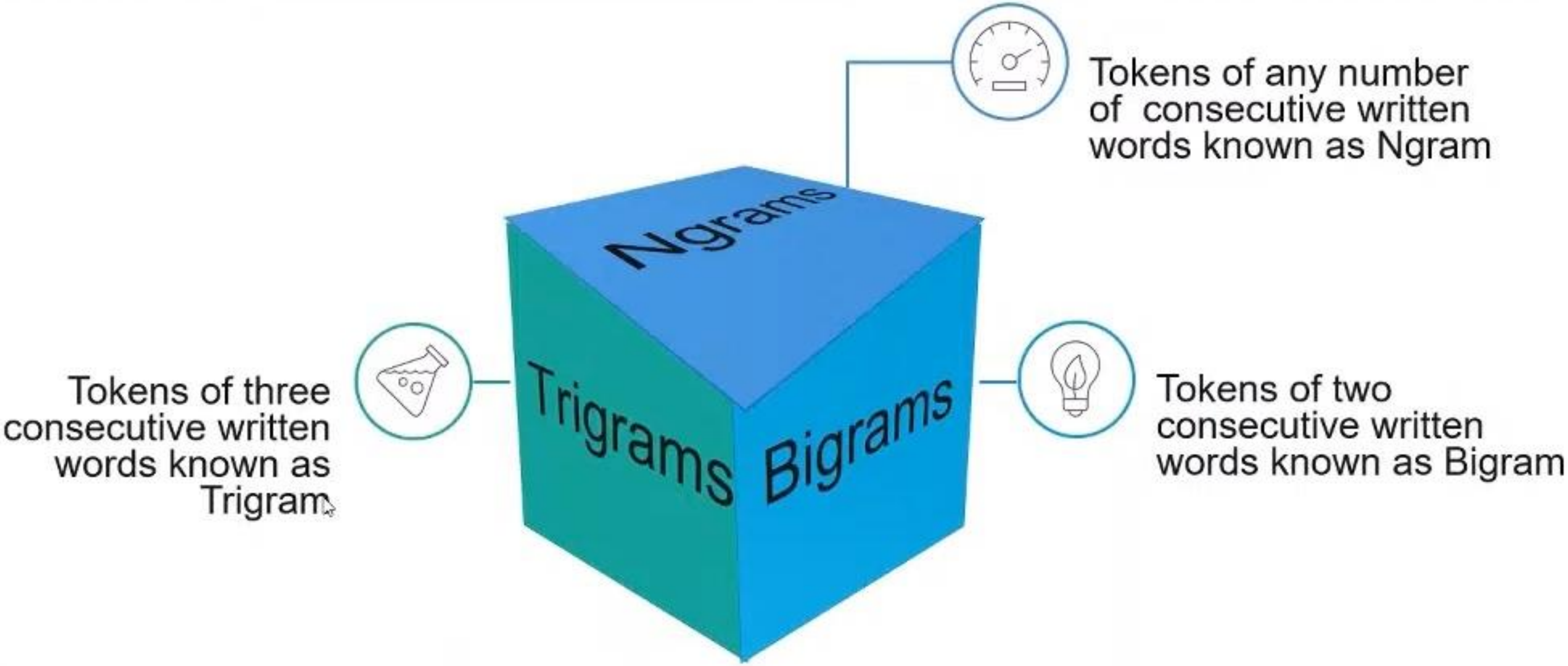
Tokenization - Use

01 Break a complex sentence into words

02 Understand the importance of each of the words with respect to the sentence

03 Produces a structural description on an input sentence

Tokenization



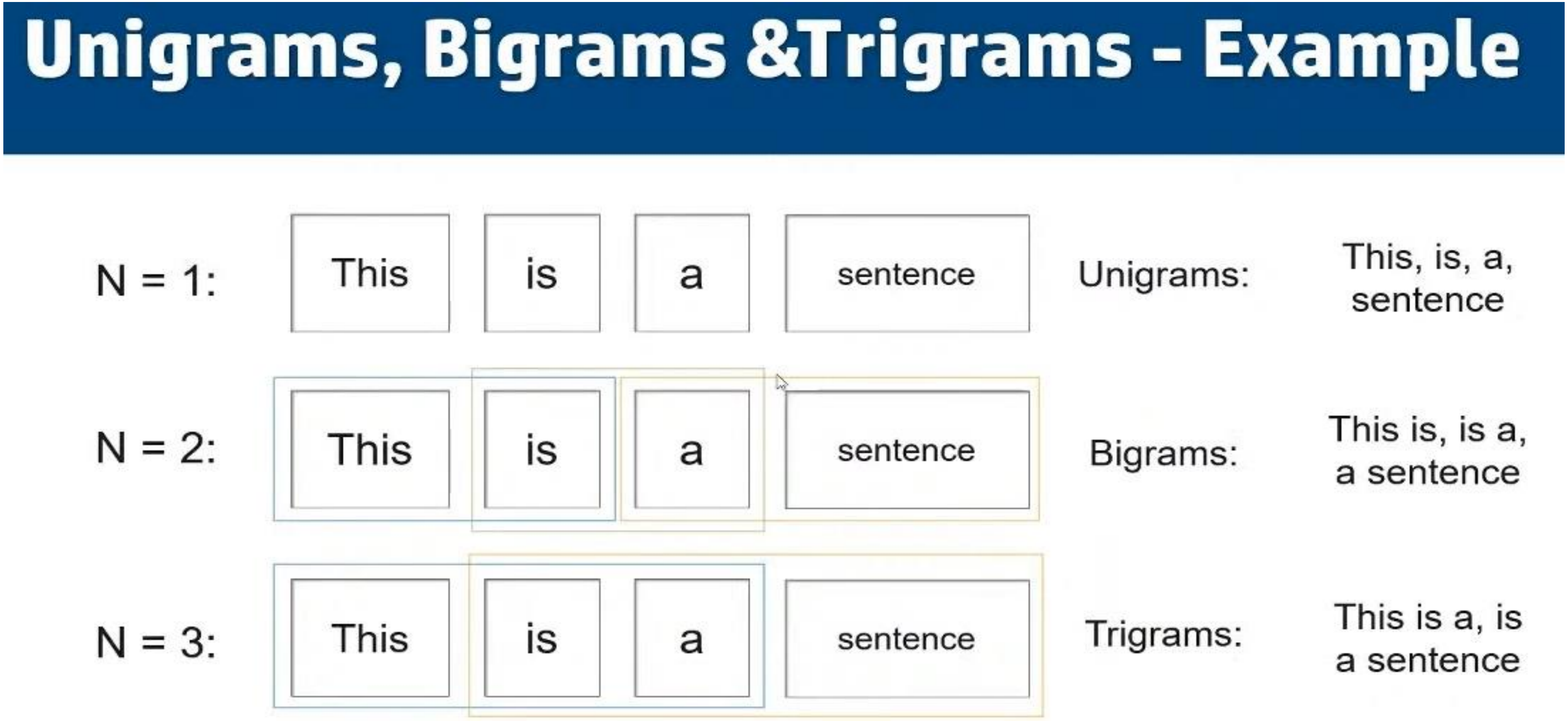
Tokens of any number of consecutive written words known as Ngram

Tokens of three consecutive written words known as Trigram

Tokens of two consecutive written words known as Bigram

NLTK also allows you to tokenize phrases, containing more than one word

Tokenization



Unigrams, Bigrams &Trigrams - Example

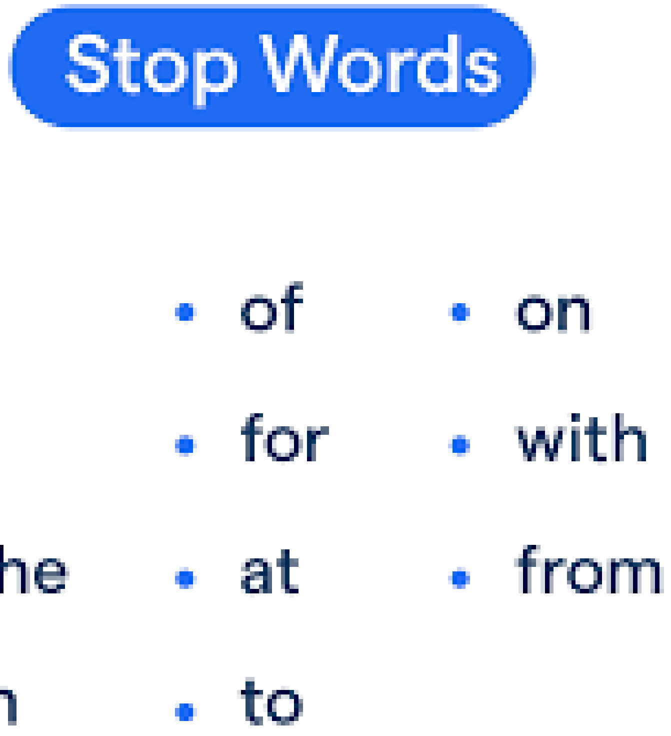
N = 1: This is a sentence Unigrams: This, is, a, sentence

N = 2: This is a sentence Bigrams: This is, is a, a sentence

N = 3: This is a sentence Trigrams: This is a, is a sentence

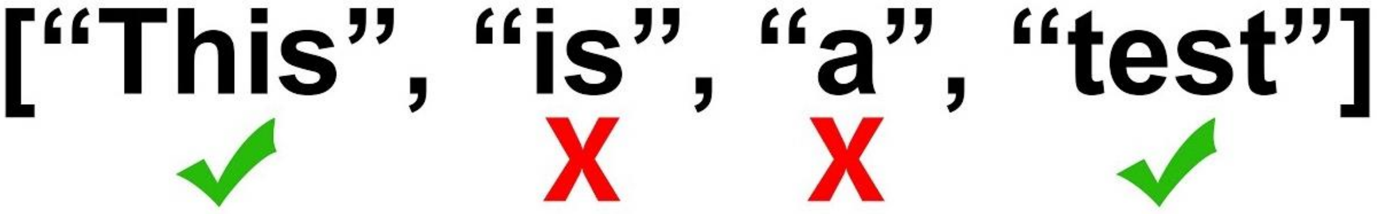
edureka!

Stop Words Removal



Stop Words

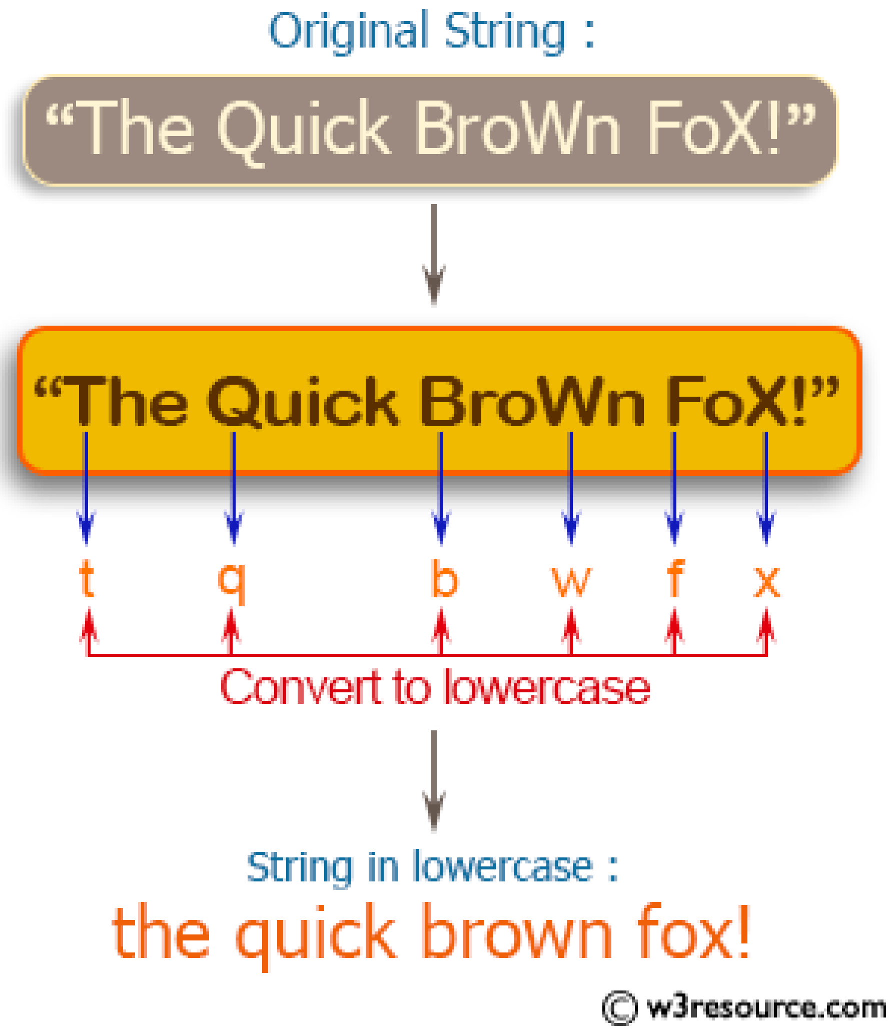
- a
- I
- the
- in
- of
- for
- at
- to
- on
- with
- from



["This", "is", "a", "test"]

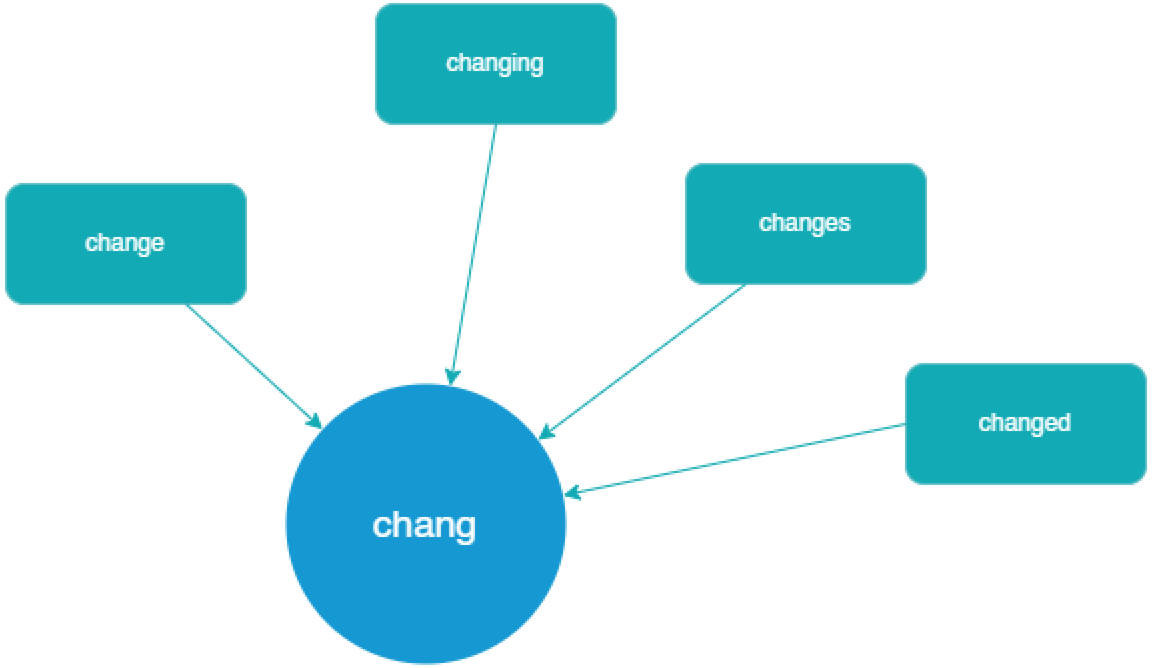
Remove Digits & Punctuations

Convert to Lowercase

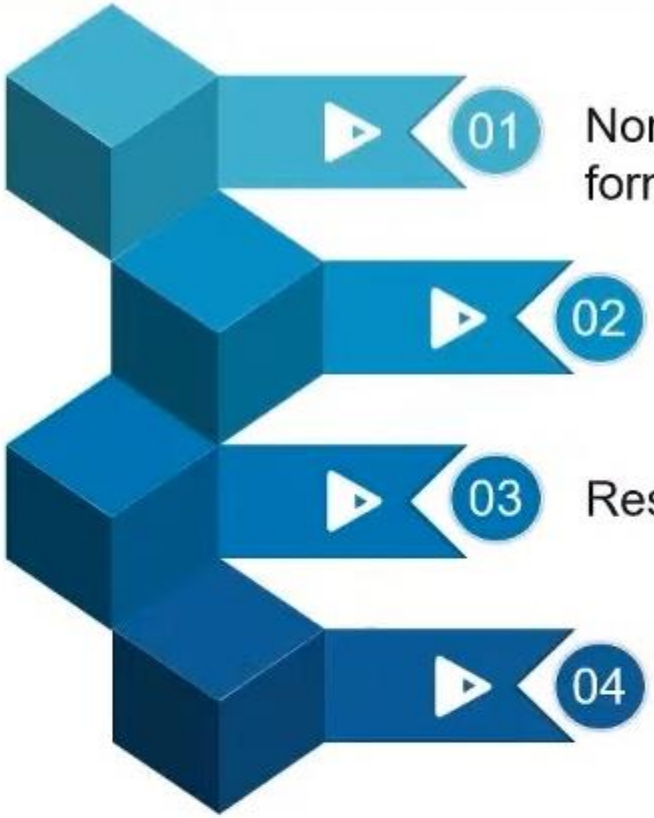


Stemming

Stemming is the process of reducing a word to its word stem, which affixes to suffixes and prefixes or to the roots of words known as the lemma.



Stemming



01 Normalize words into its base form or root form

02 For example, the words fish, fishes and fishing all stem into **fish**

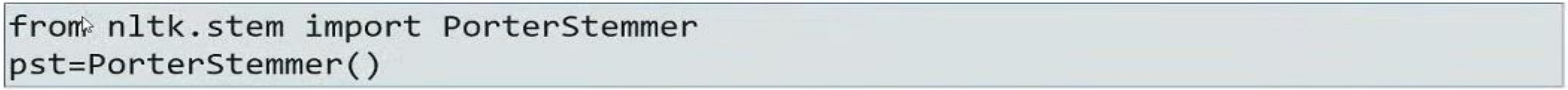
03 Result may not be the ‘root’ word

04 Common types of Stemming algorithms: Porter, Lancaster and Snowball

Few words like study, studies and studying stems into ‘studi’, which is not an English word

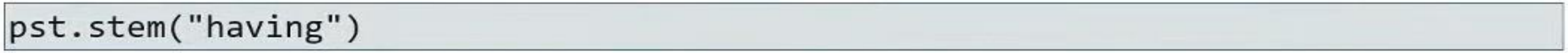
Stemming

Porter Stemmer with NLTK

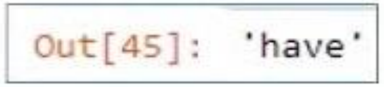


```
from nltk.stem import PorterStemmer
pst=PorterStemmer()
```

Use the stemmer for the word ‘having’:

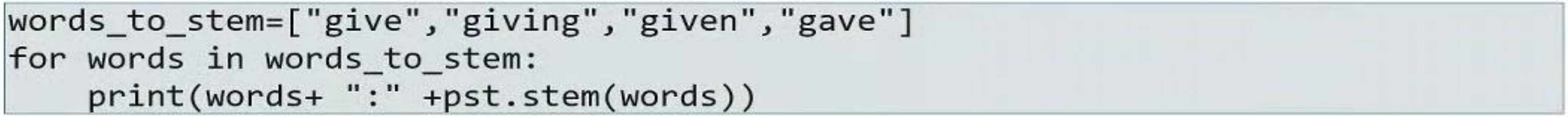


```
pst.stem("having")
```

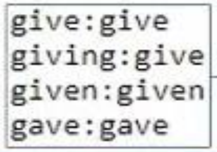


```
Out[45]: 'have'
```

Using Porter Stemmer to stem a list of words:



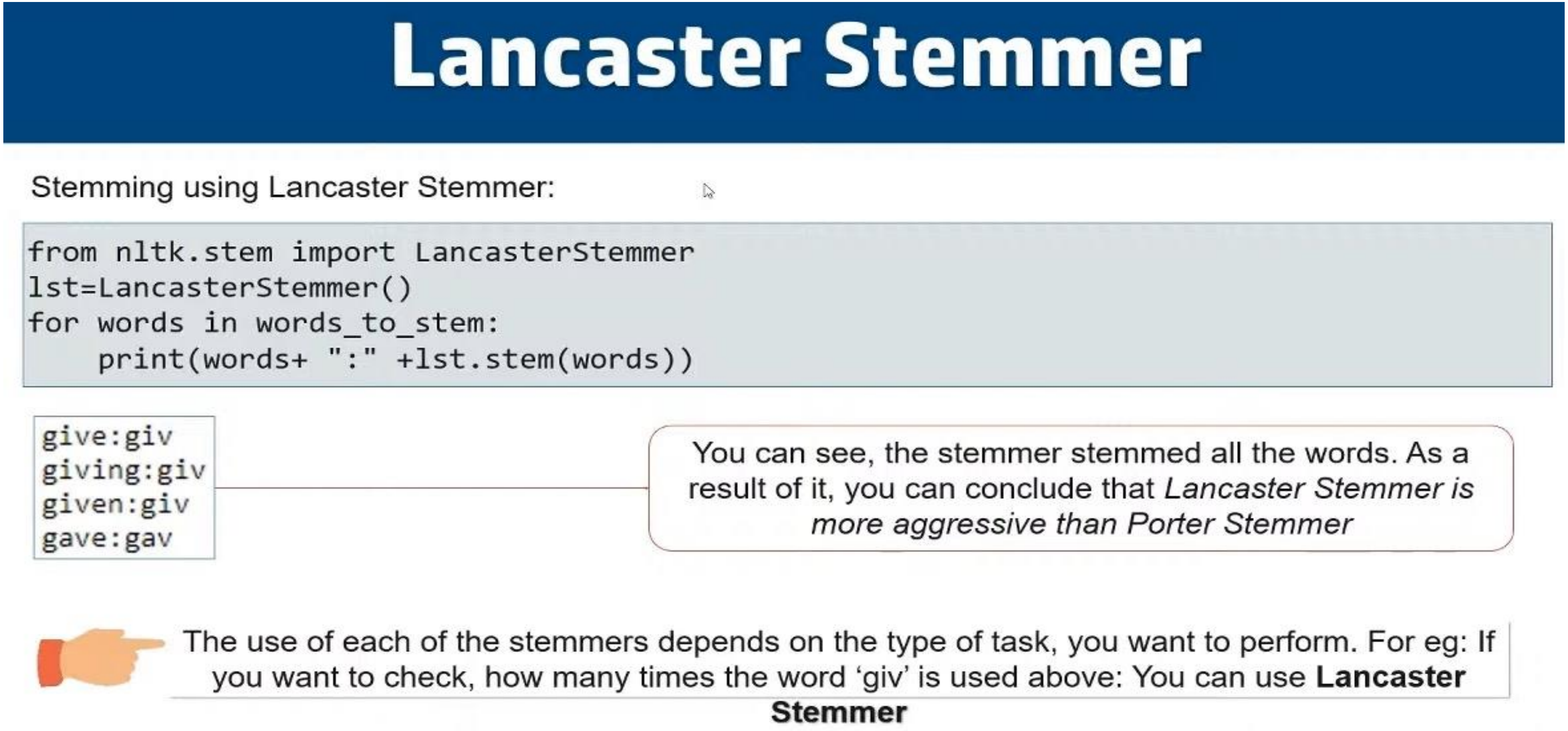
```
words_to_stem=["give","giving","given","gave"]
for words in words_to_stem:
    print(words+ ":" +pst.stem(words))
```



```
give:give
giving:give
given:given
gave:gave
```

You can see, the stemmer removed only ‘ing’ and replaced it with ‘e’

Stemming



Lancaster Stemmer

Stemming using Lancaster Stemmer:

```
from nltk.stem import LancasterStemmer
lst=LancasterStemmer()
for words in words_to_stem:
    print(words+ ":" +lst.stem(words))
```

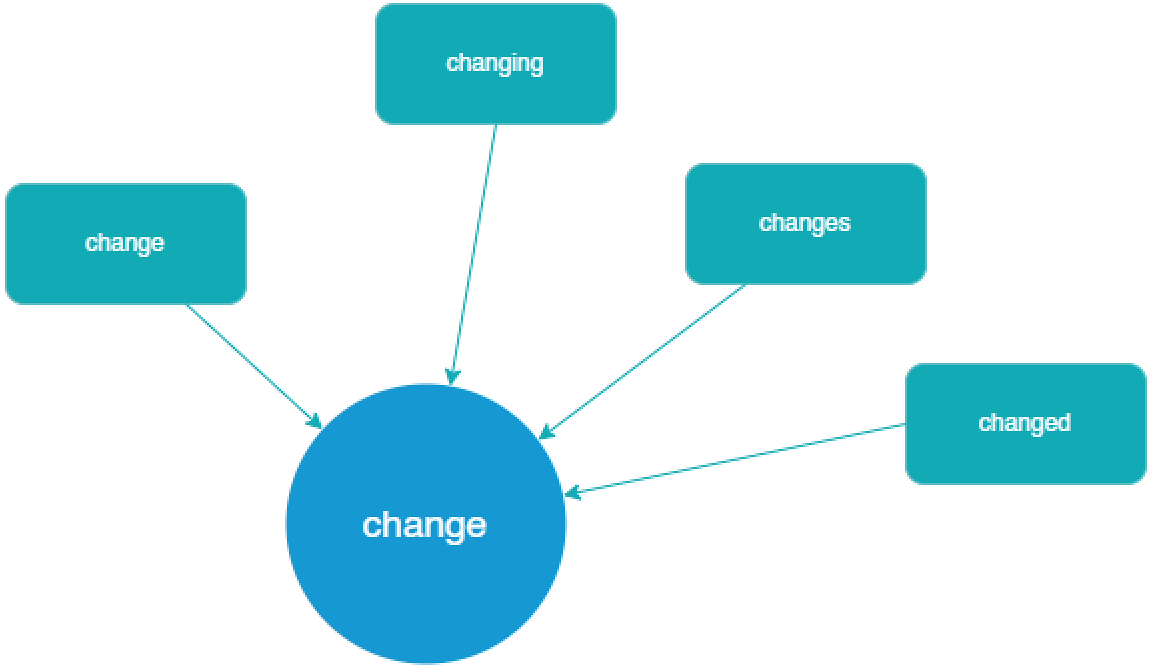
```
give:giv
giving:giv
given:giv
gave:gav
```

You can see, the stemmer stemmed all the words. As a result of it, you can conclude that *Lancaster Stemmer is more aggressive than Porter Stemmer*

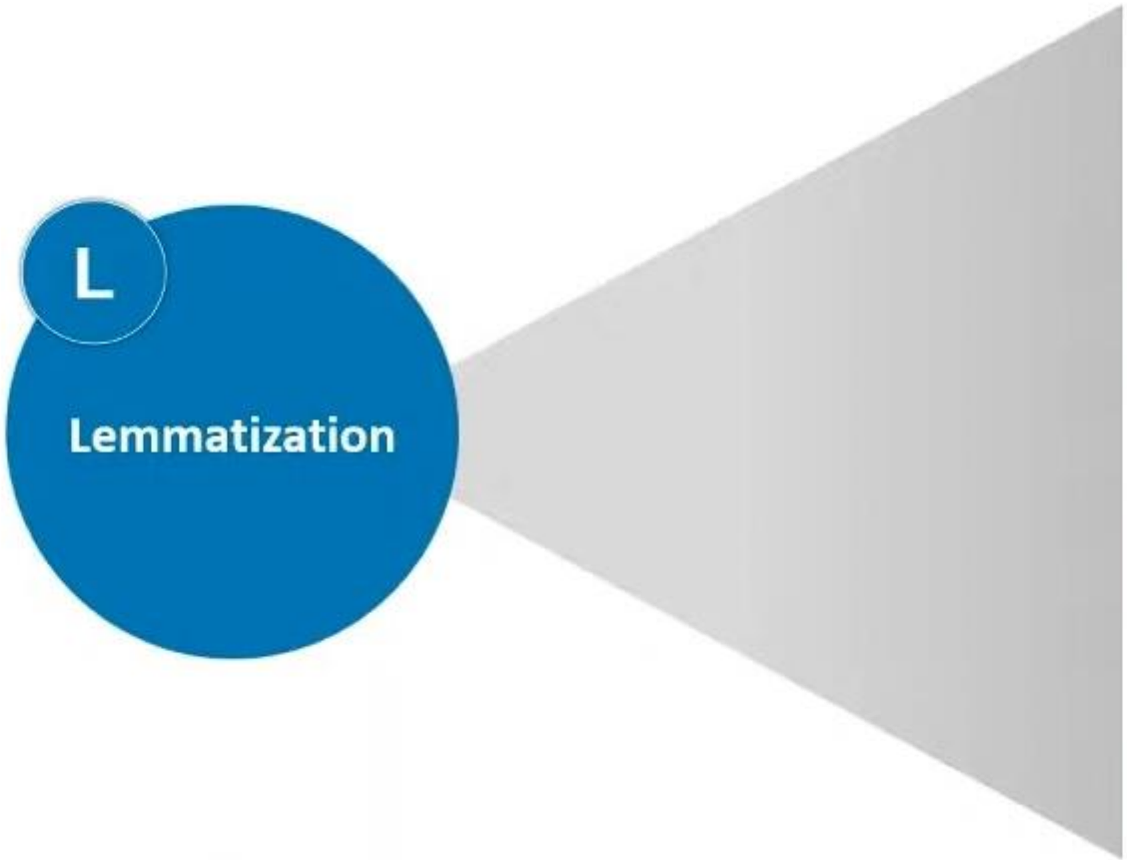
The use of each of the stemmers depends on the type of task, you want to perform. For eg: If you want to check, how many times the word ‘giv’ is used above: You can use **Lancaster Stemmer**

Lemmatization

The technique of collecting together the various inflected forms of a word so that they can be analysed as a single item is known as **Lemmatization**.



Lemmatization

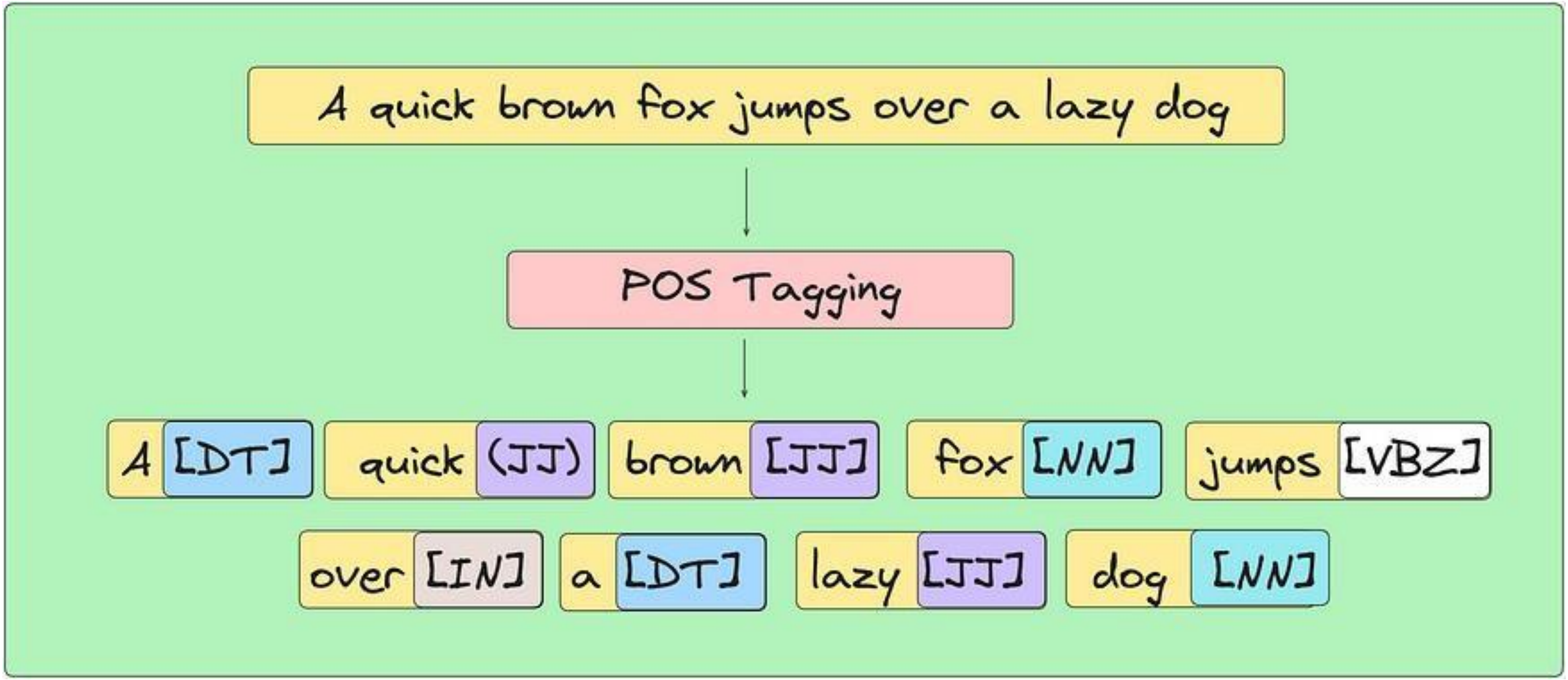


L

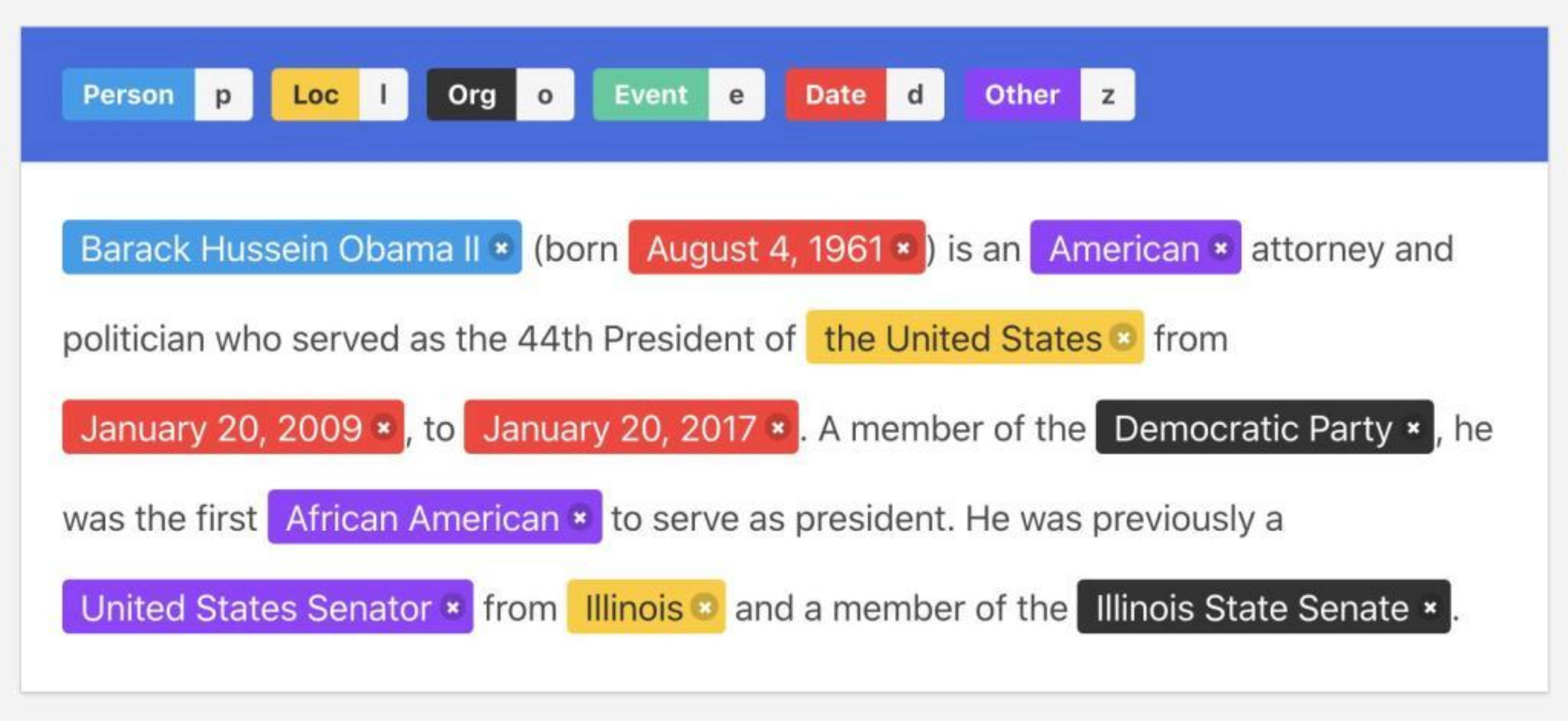
Lemmatization

- Groups together different inflected forms of a word, called Lemma
- Somehow similar to stemming, as it maps several words into one common root
- Output of Lemmatization is a proper word
- For example, a lemmatizer should map *gone*, *going* and *went* into *go*

POS Tagging


A quick brown fox jumps over a lazy dog
POS Tagging
A [DT]
quick (JJ)
brown [JJ]
fox [NN]
jumps [VBZ]
over [IN]
a [DT]
lazy [JJ]
dog [NN]

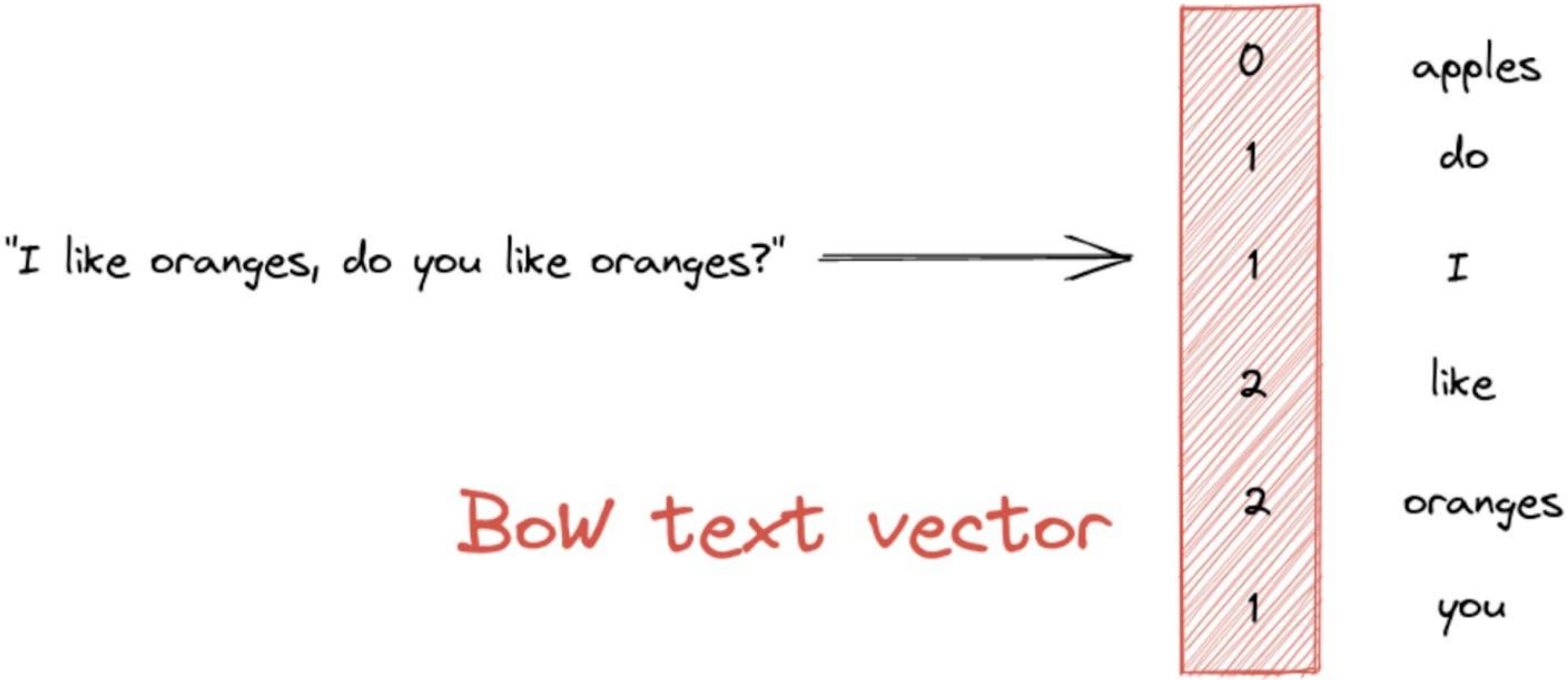
Named Entity Recognition



Barack Hussein Obama II (born August 4, 1961) is an American attorney and politician who served as the 44th President of the United States from January 20, 2009, to January 20, 2017. A member of the Democratic Party, he was the first African American to serve as president. He was previously a United States Senator from Illinois and a member of the Illinois State Senate.

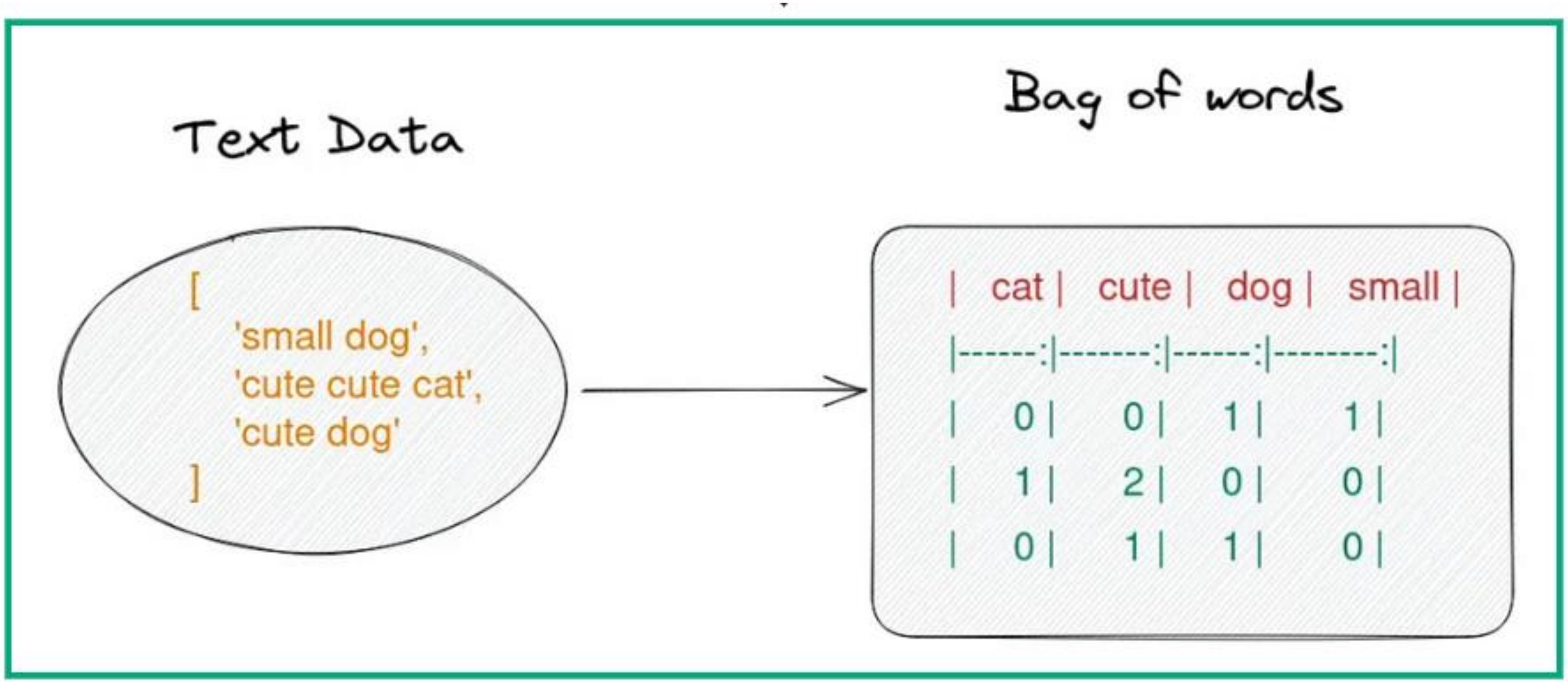
Text Vectorization / Feature Engineering

Text Vectorization used represent the text in the numeric vector in such a way that the ML algorithm can understand the text attribute.



0	apples
1	do
1	I
2	like
2	oranges
1	you

Text Vectorization - BoW



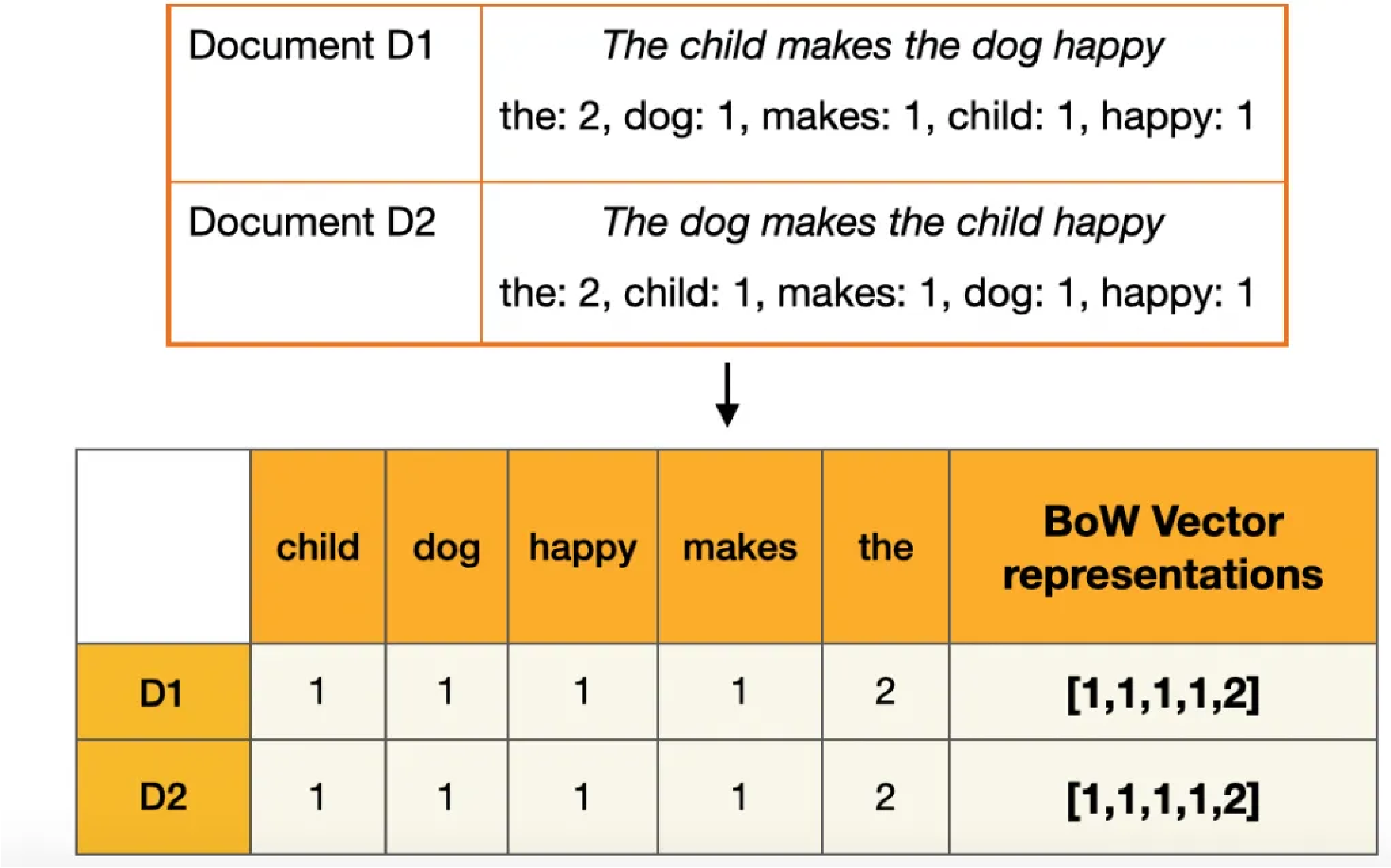
Text Data

```
[
    'small dog',
    'cute cute cat',
    'cute dog'
]
```

Bag of words

cat	cute	dog	small
0	0	1	1
1	2	0	0
0	1	1	0

Text Vectorization - BoW



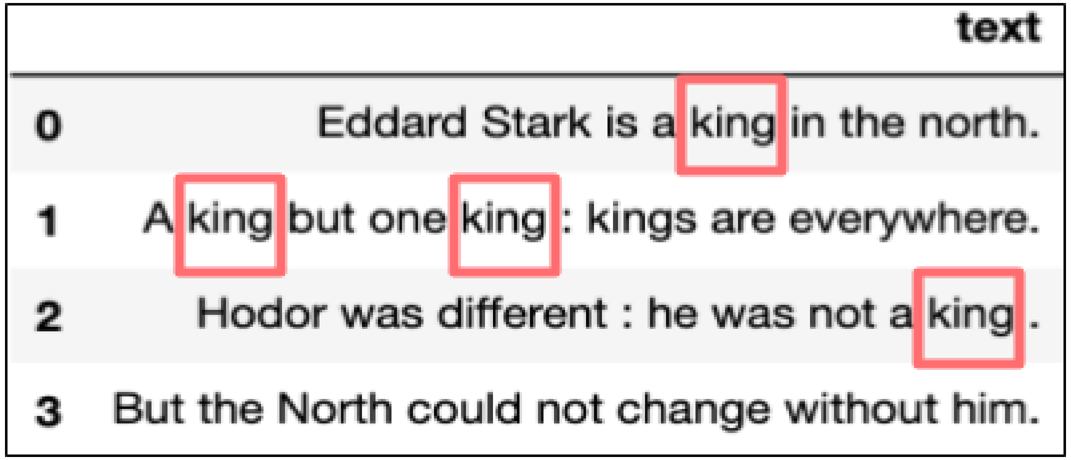
Document D1	*The child makes the dog happy* the: 2, dog: 1, makes: 1, child: 1, happy: 1
Document D2	*The dog makes the child happy* the: 2, child: 1, makes: 1, dog: 1, happy: 1

↓

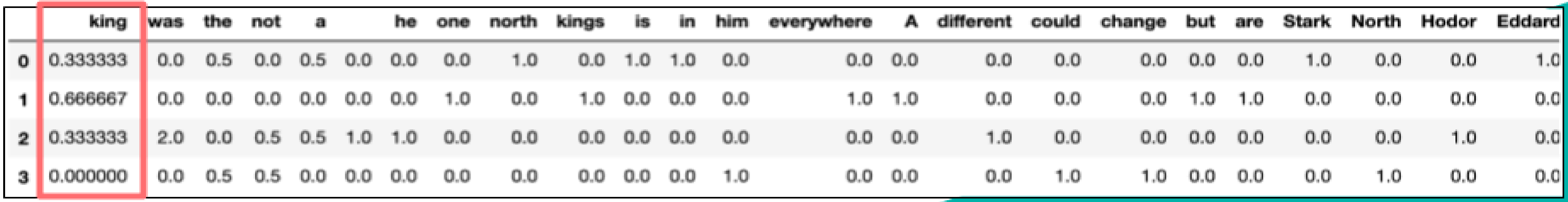
	child	dog	happy	makes	the	**BoW Vector representations**
D1	1	1	1	1	2	**[1,1,1,1,2]**
D2	1	1	1	1	2	**[1,1,1,1,2]**

Text Vectorization – TF-IDF

TF-IDF (term frequency-inverse document frequency) vectorization is a statistical method that converts text documents into numerical vectors based on the importance of words in a document

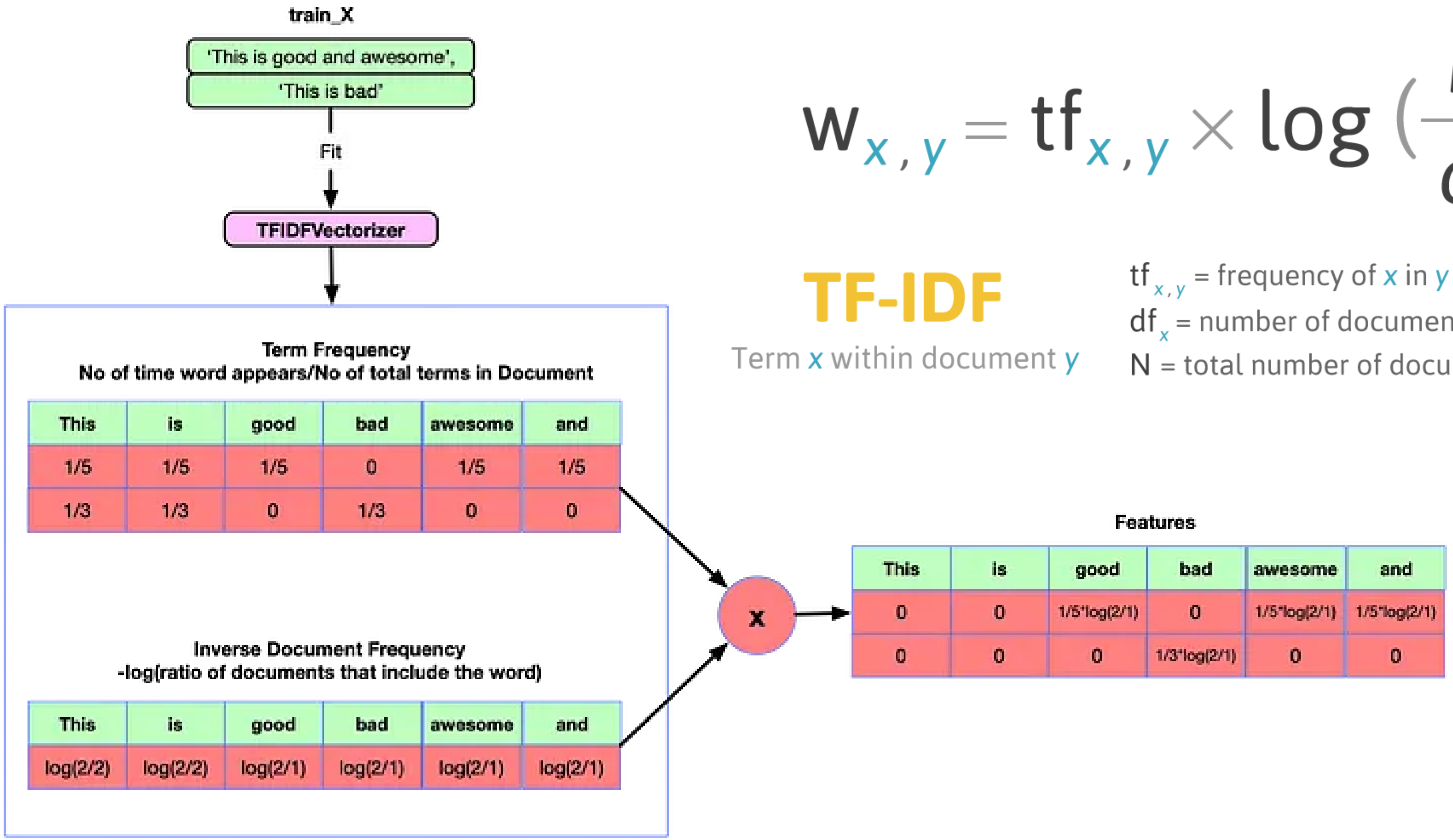


	text	tf	idf
0	Eddard Stark is a king in the north.	1	3
1	A king but one king : kings are everywhere.	2	3
2	Hodor was different : he was not a king .	1	3
3	But the North could not change without him.	0	3



	king	was	the	not	a		he	one	north	kings	is	in	him	everywhere	A	different	could	change	but	are	Stark	North	Hodor	Eddard
0	0.333333	0.0	0.5	0.0	0.5	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
1	0.666667	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0
2	0.333333	2.0	0.0	0.5	0.5	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.000000	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0

Text Vectorization – TF-IDF



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)

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

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents