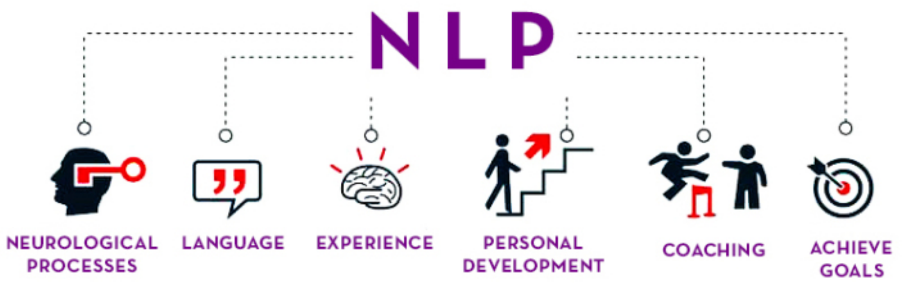
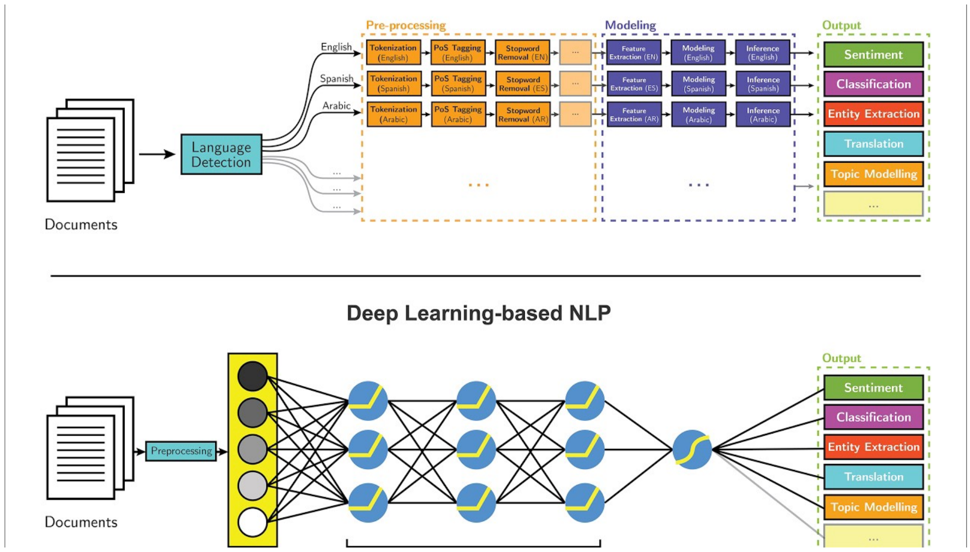
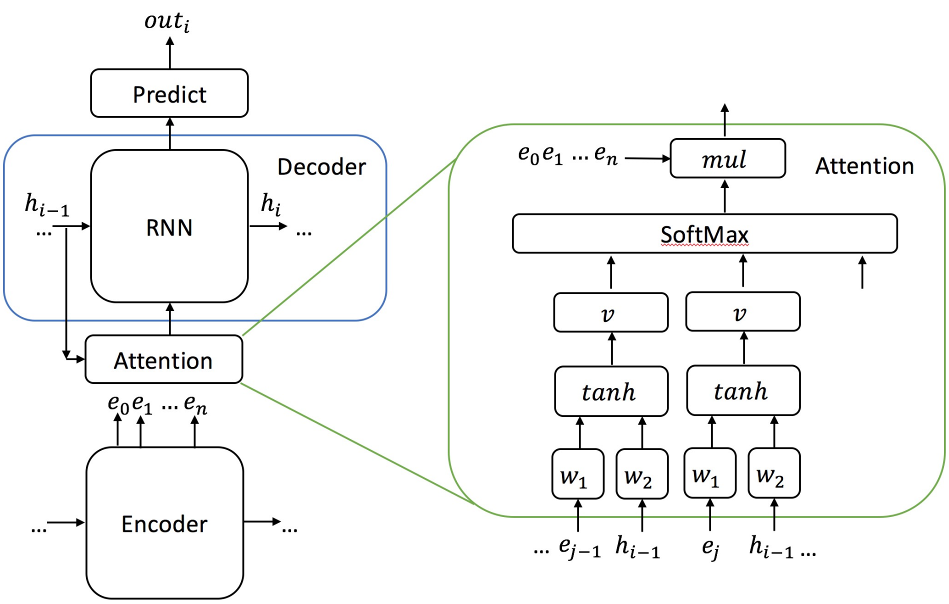
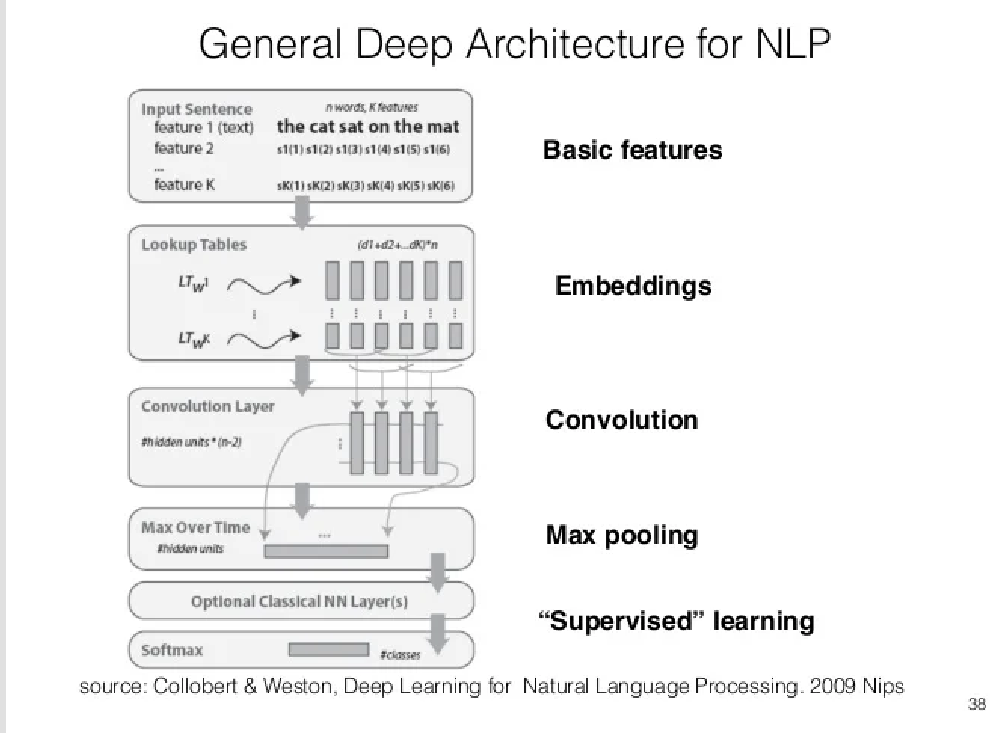
Memo - Natural Language Processing

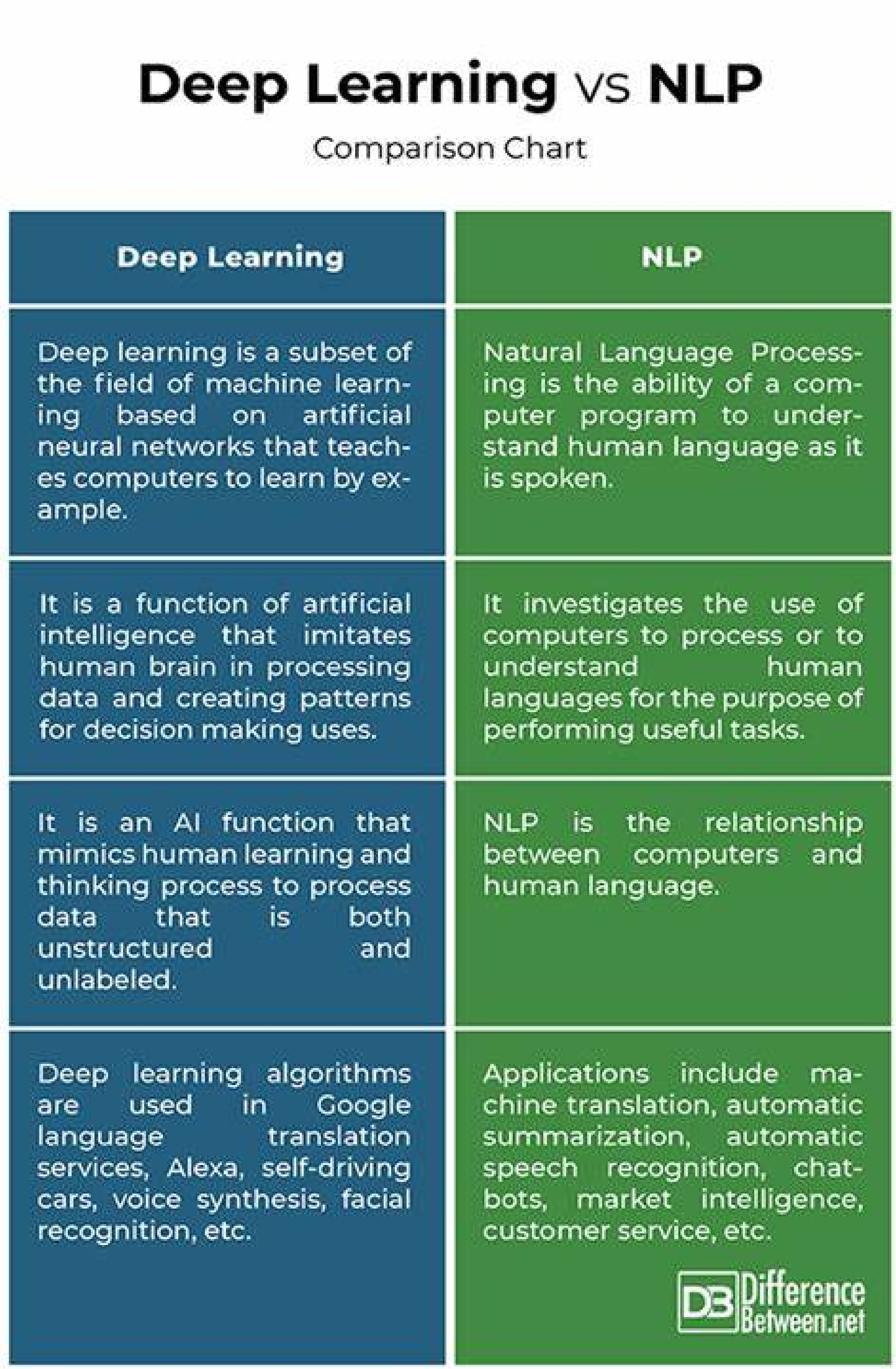
Natural Language Processing Introduction








General Deep Architecture for NLP
Input Sentence
n words, K features
feature 1 (text)
the cat sat on the mat
feature 2
s1(1) s1(2) s1(3) s1(4) s1(5) s1(6)
...
feature K
sK(1) sK(2) sK(3) sK(4) sK(5) sK(6)
Basic features
Lookup Tables
(d1+d2+...dK)*n
LT_{W^1}
LT_{W^K}
Embeddings
Convolution Layer
#hidden units * (n-2)
Convolution
Max Over Time
#hidden units
Max pooling
Optional Classical NN Layer(s)
Softmax
#classes
"Supervised" learning
source: Collobert & Weston, Deep Learning for Natural Language Processing. 2009 Nips
38

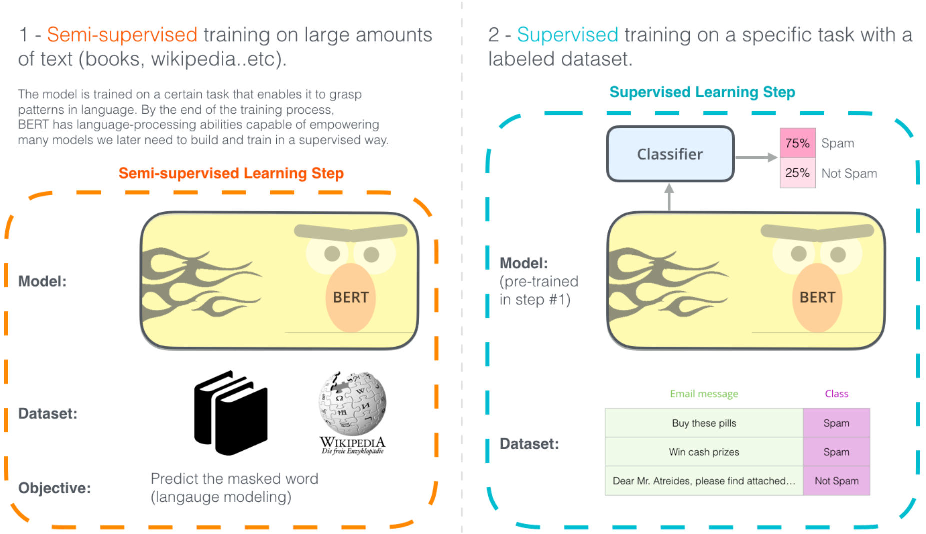


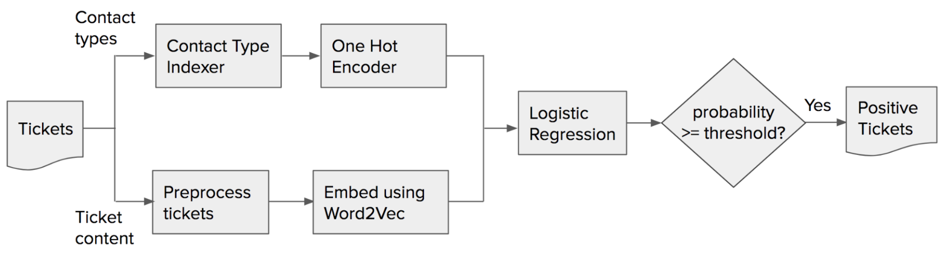
Deep Learning vs NLP

Comparison Chart

Deep Learning	NLP
Deep learning is a subset of the field of machine learning based on artificial neural networks that teaches computers to learn by example.	Natural Language Processing is the ability of a computer program to understand human language as it is spoken.
It is a function of artificial intelligence that imitates human brain in processing data and creating patterns for decision making uses.	It investigates the use of computers to process or to understand human languages for the purpose of performing useful tasks.
It is an AI function that mimics human learning and thinking process to process data that is both unstructured and unlabeled.	NLP is the relationship between computers and human language.
Deep learning algorithms are used in Google language translation services, Alexa, self-driving cars, voice synthesis, facial recognition, etc.	Applications include machine translation, automatic summarization, automatic speech recognition, chatbots, market intelligence, customer service, etc.

DifferenceBetween.net



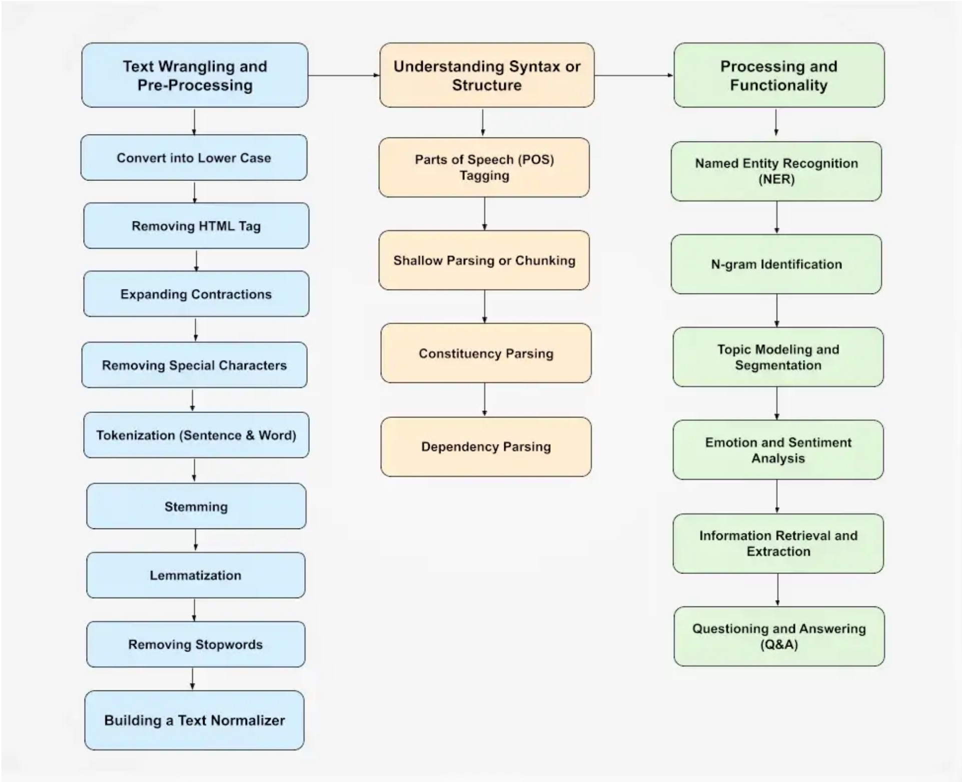


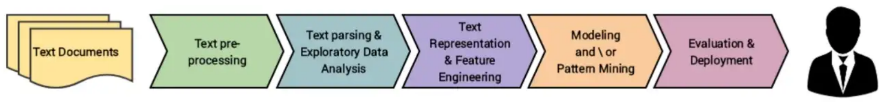
from google.colab import drive drive.mount('/content/drive', force_remount=True)

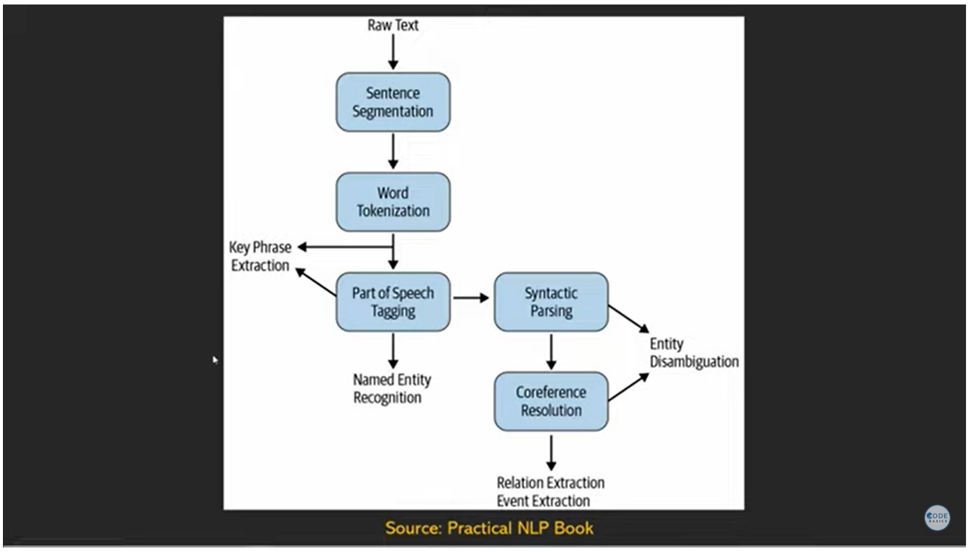
> # Model Resources
>
> TensorFlow Hub
> fastText
> GPT-3

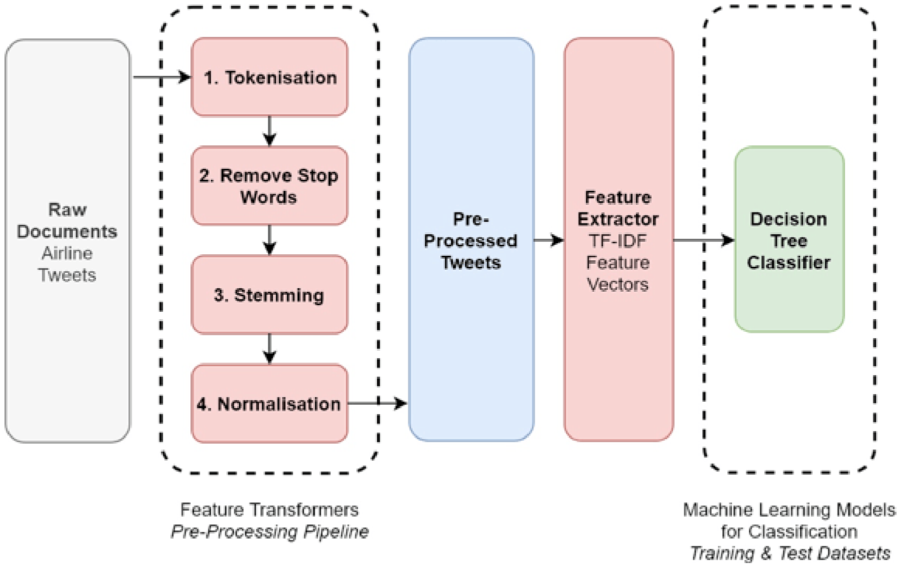
> # OpenSource Ecosytem - Libraries
>
> 
>

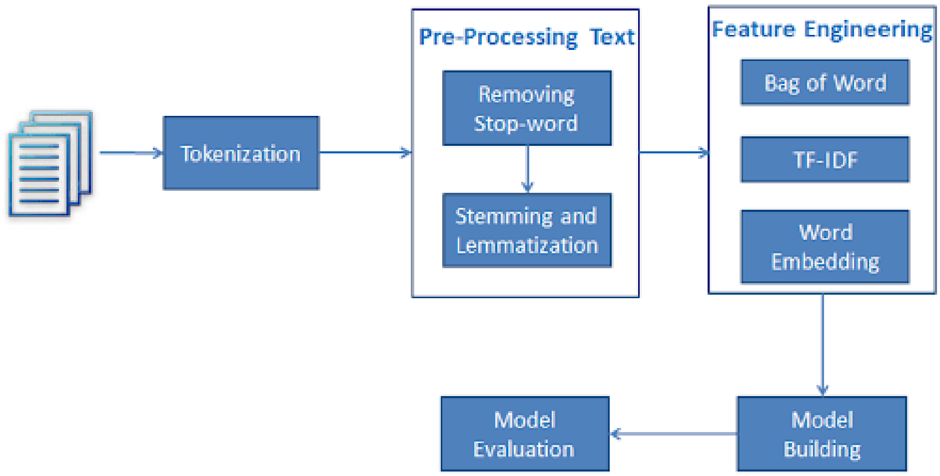
> # Workflow

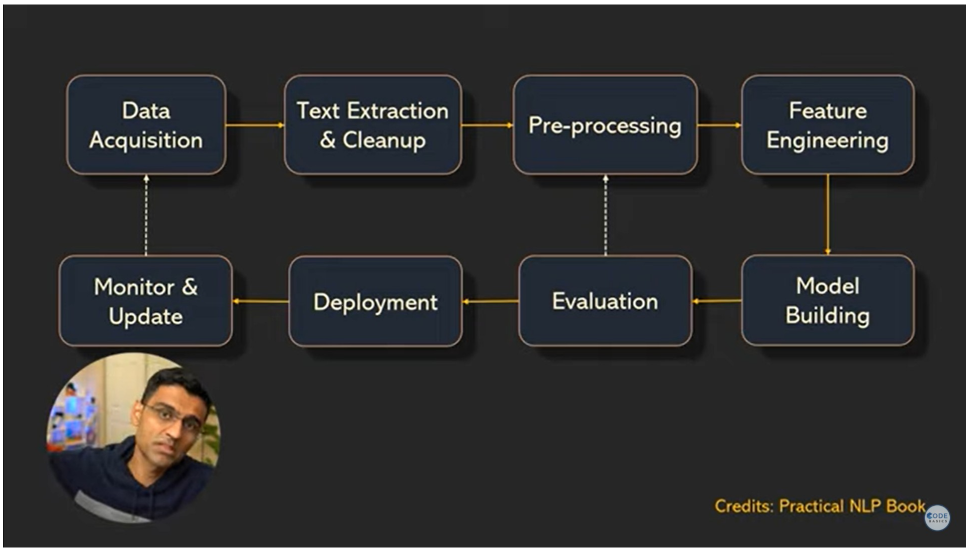

Text Wrangling and Pre-Processing
Convert into Lower Case
Removing HTML Tag
Expanding Contractions
Removing Special Characters
Tokenization (Sentence & Word)
Stemming
Lemmatization
Removing Stopwords
Building a Text Normalizer
Understanding Syntax or Structure
Parts of Speech (POS) Tagging
Shallow Parsing or Chunking
Constituency Parsing
Dependency Parsing
Processing and Functionality
Named Entity Recognition (NER)
N-gram Identification
Topic Modeling and Segmentation
Emotion and Sentiment Analysis
Information Retrieval and Extraction
Questioning and Answering (Q&A)


Text Documents
Text pre-processing
Text parsing & Exploratory Data Analysis
Text Representation & Feature Engineering
Modeling and \ or Pattern Mining
Evaluation & Deployment

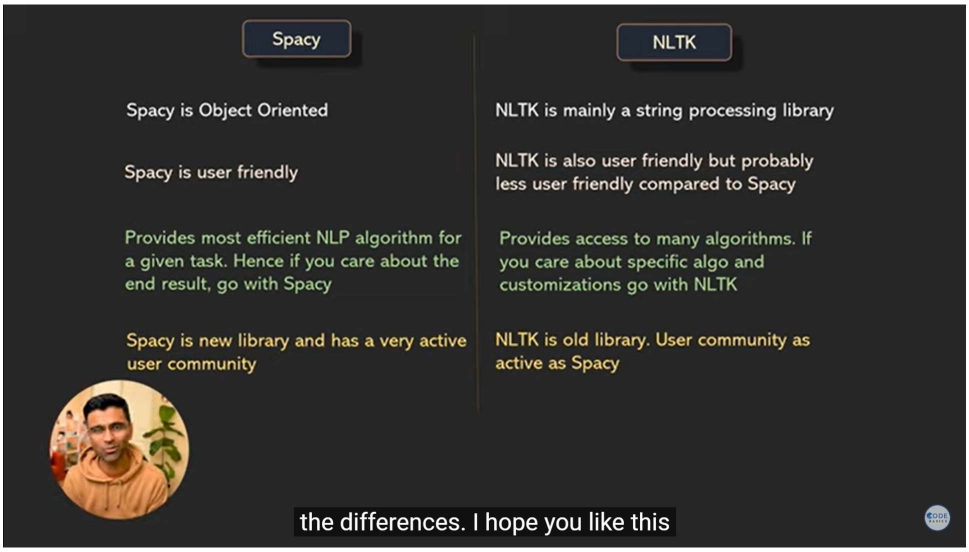

Raw Text
Sentence Segmentation
Word Tokenization
Key Phrase Extraction
Part of Speech Tagging
Syntactic Parsing
Entity Disambiguation
Named Entity Recognition
Coreference Resolution
Relation Extraction
Event Extraction
Source: Practical NLP Book







Pre-processing - Spacy vs NLTK



NLTK

In [ ]:
```
import nltk
```

In [ ]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\User\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[ ]:
```
True
```

In [ ]:
```
from nltk.tokenize import sent_tokenize
```

In [ ]:
```
sent_tokenize("An end to end NLP project consists of many steps. These steps togeth
```

Out[ ]:
```
['An end to end NLP project consists of many steps.',
 'These steps together forms an NLP pipeline.',
 'The pipeline has various stages such as data acquisition, data cleaning, pre-pro
cessing, model building, deployment, monitor and update etc.']
```

In [ ]:
```
from nltk.tokenize import word_tokenize
word_tokenize("An end to end NLP project consists of many steps.")
```

Out[ ]:
```
['An',
 'end',
 'to',
 'end',
 'NLP',
 'project',
 'consists',
 'of',
 'many',
 'steps',
 '.']
```

Spacy

In [ ]:
```
import spacy
```

In [ ]:
```
!python -m spacy download en
```

In [ ]:
```
nlp = spacy.load('en_core_web_sm')
#nlp = spacy.blank("en")
```

In [ ]:
```
type(nlp)
```

Out[ ]:
```
spacy.lang.en.English
```

In [ ]:
```
text = "An end to end NLP project consists of many steps. These steps together form
```

In [ ]:
```
doc = nlp(text)
type(doc)
```

Out[ ]:
```
spacy.tokens.doc.Doc
```

In [ ]:
```
for sentence in doc.sents:
  print(sentence)
```

```
An end to end NLP project consists of many steps.
These steps together forms an NLP pipeline.
The pipeline has various stages such as data acquisition, data cleaning, pre-proce
ssing, model building, deployment, monitor and update etc.
```

In [ ]:
```
for word in sentence:
  print(word)
```

```
The
pipeline
has
various
stages
such
as
data
acquisition
,
data
cleaning
,
pre
-
processing
,
model
building
,
deployment
,
monitor
and
update
etc
.
```

In [ ]:
```
for token in doc:
   print(token)
```

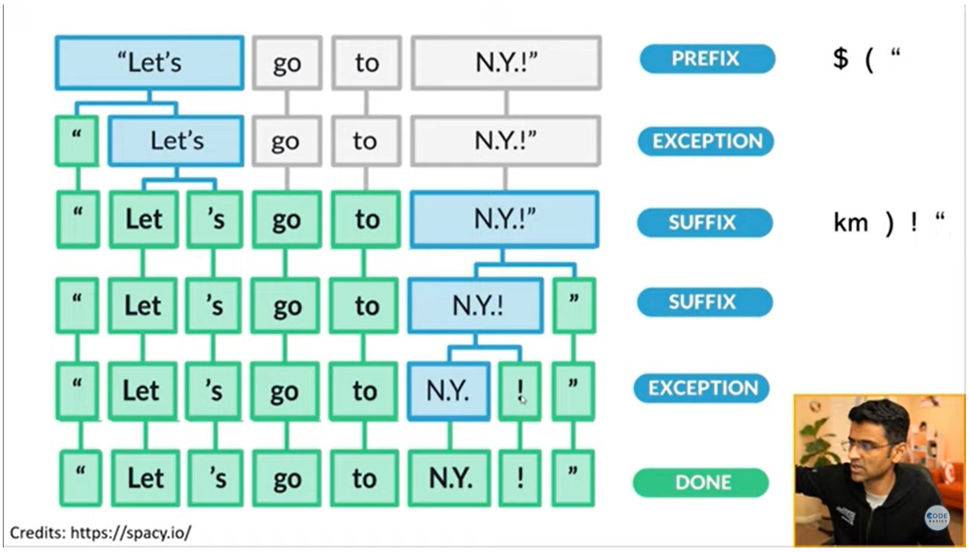
```
An
end
to
end
NLP
project
consists
of
many
steps
.
These
steps
together
forms
an
NLP
pipeline
.
The
pipeline
has
various
stages
such
as
data
acquisition
,
data
cleaning
,
pre
-
processing
,
model
building
,
deployment
,
monitor
and
update
etc
.
```

In [ ]:
```
doc[0]
```

Out[ ]:
```
An
```

In [ ]:
```
doc[-1]
```

Out[ ]:
```
.
```




```
In [ ]: token_0 = doc[0]
        token_0
```

Out[ ]: An

```
In [ ]: type(token_0)
```

Out[ ]: spacy.tokens.token.Token

```
In [ ]: dir(token_0)
```

Out[ ]:
```
['_',
 '__bytes__',
 '__class__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__pyx_vtable__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__unicode__',
 'ancestors',
 'check_flag',
 'children',
 'cluster',
 'conjuncts',
 'dep',
 'dep_',
 'doc',
 'ent_id',
 'ent_id_',
 'ent_iob',
 'ent_iob_',
 'ent_kb_id',
 'ent_kb_id_',
 'ent_type',
 'ent_type_',
 'get_extension',
 'has_dep',
 'has_extension',
 'has_head',
 'has_morph',
 'has_vector',
 'head',
 'i',
 'idx',
 'iob_strings',
 'is_alpha',
 'is_ancestor',
 'is_ascii',
 'is_bracket',
 'is_currency',
 'is_digit',
 'is_left_punct',
```

```
 'is_lower',
 'is_oov',
 'is_punct',
 'is_quote',
 'is_right_punct',
 'is_sent_end',
 'is_sent_start',
 'is_space',
 'is_stop',
 'is_title',
 'is_upper',
 'lang',
 'lang_',
 'left_edge',
 'lefts',
 'lemma',
 'lemma_',
 'lex',
 'lex_id',
 'like_email',
 'like_num',
 'like_url',
 'lower',
 'lower_',
 'morph',
 'n_lefts',
 'n_rights',
 'nbor',
 'norm',
 'norm_',
 'orth',
 'orth_',
 'pos',
 'pos_',
 'prefix',
 'prefix_',
 'prob',
 'rank',
 'remove_extension',
 'right_edge',
 'rights',
 'sent',
 'sent_start',
 'sentiment',
 'set_extension',
 'set_morph',
 'shape',
 'shape_',
 'similarity',
 'subtree',
 'suffix',
 'suffix_',
 'tag',
 'tag_',
 'tensor',
 'text',
 'text_with_ws',
 'vector',
 'vector_norm',
 'vocab',
 'whitespace_']
```

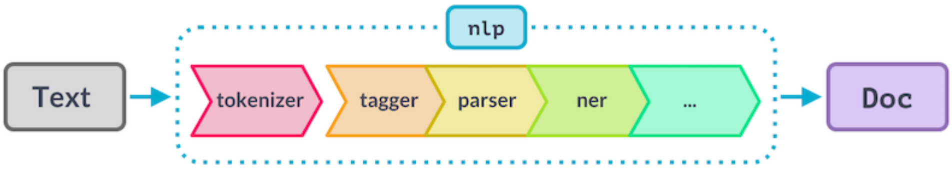
In [ ]:
```
token_0.text
```
Out[ ]:
```
'An'
```

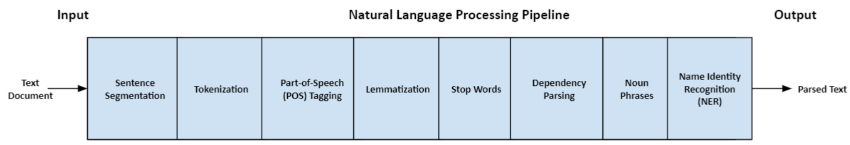
In [ ]:
```
token_0.like_num
```
Out[ ]:
```
False
```

Pipeline in Spacy

!pip intall spacy
!python -m spacy download en
import spacy
nlp = spacy.load("en_core_web_sm")

NLP Pipeline: Building an NLP Pipeline, Step-by-Step





NAME	COMPONENT	CREATES	DESCRIPTION
tokenizer	`Tokenizer`	`Doc`	Segment text into tokens.
tagger	`Tagger`	`Doc[i].tag`	Assign part-of-speech tags.
parser	`DependencyParser`	`Doc[i].head`, `Doc[i].dep`, `Doc.sents`, `Doc.noun_chunks`	Assign dependency labels.
ner	`EntityRecognizer`	`Doc.ents`, `Doc[i].ent_iob`, `Doc[i].ent_type`	Detect and label named entities.
textcat	`TextCategorizer`	`Doc.cats`	Assign document labels.
...	custom components	`Doc._.xxx`, `Token._.xxx`, `Span._.xxx`	Assign custom attributes, methods or properties.

In [ ]:
```
import spacy

nlp = spacy.blank("en")
```

In [ ]:
```
with open("data/student.txt", "r") as f:
    text = f.readlines()
print(text)
```

```
['Dayton high school, 8th grade students information\n', '=======================
=========================\n', '\n', 'Name\tbirth day   \temail\n', '-----\t------
------\t------\n', 'Virat   5 June, 1882    virat@kohli.com\n', 'Maria\t12 April,
2001  maria@sharapova.com\n', 'Serena  24 June, 1998   serena@williams.com \n', 'J
oe      1 May, 1997    joe@root.com']
```

In [ ]:
```
text = " ".join(text)
text
```

Out[ ]:
```
'Dayton high school, 8th grade students information\n ===========================
=====================\n \n Name\tbirth day   \temail\n -----\t------------\t-----
-\n Virat   5 June, 1882    virat@kohli.com\n Maria\t12 April, 2001  maria@sharapo
va.com\n Serena  24 June, 1998   serena@williams.com \n Joe      1 May, 1997    jo
e@root.com'
```

In [ ]:
```
doc = nlp(text)
emails = []
for token in doc:
    if token.like_email:
        emails.append(token)
emails
```

Out[ ]:
```
[virat@kohli.com, maria@sharapova.com, serena@williams.com, joe@root.com]
```

Customizing tokenizer

In [ ]:
```
doc = nlp("gimme double cheese extra large healthy pizza")
```

In [ ]:
```
tokens = [token for token in doc]
tokens
```

Out[ ]:
```
[gimme, double, cheese, extra, large, healthy, pizza]
```

In [ ]:
```
from spacy.symbols import ORTH

nlp.tokenizer.add_special_case("gimme", [
    {ORTH: "gim"},
    {ORTH: "me"},
])
doc = nlp("gimme double cheese extra large healthy pizza")
tokens = [token for token in doc]
tokens
```

Out[ ]:
```
[gim, me, double, cheese, extra, large, healthy, pizza]
```

In [ ]:
```
import spacy

nlp = spacy.blank("en")

doc = nlp("Captain america ate 100$ of samosa. Then he said I can do this all day.'
```

```
for token in doc:
    print(token)
```

```
Captain
america
ate
100
$
of
samosa
.
Then
he
said
I
can
do
this
all
day
.
```

In [ ]:
```
nlp = spacy.blank("en")

nlp.pipe_names
```

Out[ ]:
```
[]
```

In [ ]:
```
nlp = spacy.load('en_core_web_sm')

nlp.pipe_names
```

Out[ ]:
```
['tok2vec', 'tagger', 'parser', 'attribute_ruler', 'lemmatizer', 'ner']
```

In [ ]:
```
nlp.pipeline
```

Out[ ]:
```
[('tok2vec', <spacy.pipeline.tok2vec.Tok2Vec at 0x2351d837dc0>),
 ('tagger', <spacy.pipeline.tagger.Tagger at 0x2351d837fa0>),
 ('parser', <spacy.pipeline.dep_parser.DependencyParser at 0x2351d74a490>),
 ('attribute_ruler',
  <spacy.pipeline.attributeruler.AttributeRuler at 0x2351da6dd80>),
 ('lemmatizer', <spacy.lang.en.lemmatizer.EnglishLemmatizer at 0x2351da97980>),
 ('ner', <spacy.pipeline.ner.EntityRecognizer at 0x2351d74a420>)]
```

In [ ]:
```
doc = nlp("Captain america ate 100$ of samosa. Then he said I can do this all day.'

for token in doc:
    print(token, "  ", token.pos_, "  ", token.lemma_)
```

```
Captain     PROPN     Captain
america     PROPN     america
ate     VERB     eat
100     NUM     100
$     NOUN     $
of     ADP     of
samosa     PROPN     samosa
.     PUNCT     .
Then     ADV     then
he     PRON     he
said     VERB     say
I     PRON     I
can     AUX     can
do     VERB     do
this     PRON     this
all     DET     all
day     NOUN     day
.     PUNCT     .
```

In [ ]:
```
doc = nlp("Captain america ate 100$ of samosa. Then he said I can do this all day.'

for token in doc:
    print(token, "     ", spacy.explain(token.pos_), "    ", token.lemma_)
```

```
Captain      proper noun      Captain
america      proper noun      america
ate      verb      eat
100      numeral      100
$      noun      $
of      adposition      of
samosa      proper noun      samosa
.      punctuation      .
Then      adverb      then
he      pronoun      he
said      verb      say
I      pronoun      I
can      auxiliary      can
do      verb      do
this      pronoun      this
all      determiner      all
day      noun      day
.      punctuation      .
```

Adding a Customizing Component to a blank pipeline

In [ ]:
```
nlp_source = spacy.load("en_core_web_sm")

nlp = spacy.blank("en")

nlp.add_pipe("ner", source=nlp_source)

nlp.pipe_names
```

Out[ ]: ['ner']

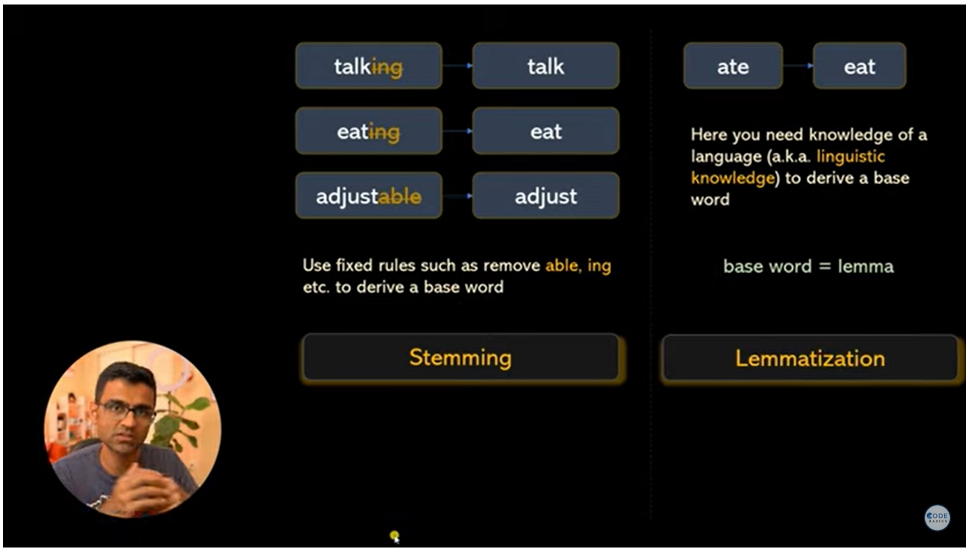
In [ ]:
```
nlp.pipe_names
```

Out[ ]: ['ner']

In [ ]:
```
nlp_source = spacy.load("en_core_web_sm")

nlp = spacy.blank("en")

nlp.add_pipe("ner", source=nlp_source)

doc = nlp("Tesla Inc is going to acquire twitter for $45 billion")

for entity in doc.ents:
  print(entity.text, "    ", entity.label_, "   ", spacy.explain(entity.label_))
```

```
Tesla Inc      ORG     Companies, agencies, institutions, etc.
$45 billion      MONEY     Monetary values, including unit
```

> ## Stemming and Lemmatization



In [ ]:
```
import nltk
import spacy
```

In [ ]:
```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
```

In [ ]:
```
words = ["eating", "eats", "eat", "ate", "adjustable", "rafting", "ability", "meet:

for word in words:
  print(word, "    ", stemmer.stem(word))
```

```
eating     eat
eats     eat
eat     eat
ate     ate
adjustable     adjust
rafting     raft
ability     abil
meeting     meet
```

In [ ]:
```
nlp = spacy.load("en_core_web_sm")

doc = nlp("eating eats eat ate adjustable rafting ability meeting better")

for token in doc:
  print(token, "    ", token.lemma_, "    ", token.lemma)
```

```
eating       eating       12092082220177030354
eats       eat       9837207709914848172
eat       eat       9837207709914848172
ate       eat       9837207709914848172
adjustable       adjustable       6033511944150694480
rafting       raft       7154368781129989833
ability       ability       11565809527369121409
meeting       meeting       14798207169164081740
better       well       4525988469032889948
```

In [ ]:
```
doc = nlp("Mando talked for 3 hours although talking isn't his thing")

for token in doc:
  print(token, "     ", token.lemma_, "     ", token.lemma)
```

```
Mando       mando       10991835832878170099
talked       talk       13939146775466599234
for       for       16037325823156266367
3       3       602994839685422785
hours       hour       9748623380567160636
although       although       343236316598008647
talking       talking       3577425109143670181
is       be       10382539506755952630
n't       not       447765159362469301
his       his       2661093235354845946
thing       thing       2473243759842082748
```

> # Adding a Customizing token Component to pipeline

In [ ]:
```
nlp.pipe_names
```

Out[ ]:
```
['tok2vec', 'tagger', 'parser', 'attribute_ruler', 'lemmatizer', 'ner']
```

In [ ]:
```
ar = nlp.get_pipe("attribute_ruler")

doc = nlp("Bro, you wanna go? Brah, don't say no! I am exhausted")

for token in doc:
  print(token, "     ", token.lemma_, "     ", token.lemma)
```

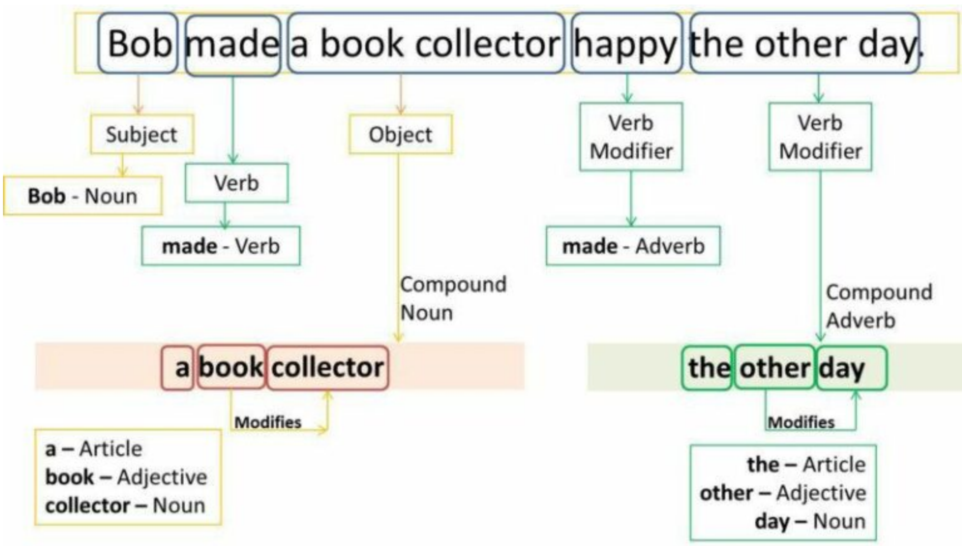
```
Bro       Bro       16427154408071002123
,       ,       2593208677638477497
you       you       7624161793554793053
wanna       wanna       13000462173222681081
go       go       8004577259940138793
?       ?       8205403955989537350
Brah       Brah       5645766505577852541
,       ,       2593208677638477497
do       do       2158845516055552166
n't       not       447765159362469301
say       say       8685289367999165211
no       no       13055779130471031426
!       !       17494803046312582752
I       I       4690420944186131903
am       be       10382539506755952630
exhausted       exhaust       5738807065439247694
```

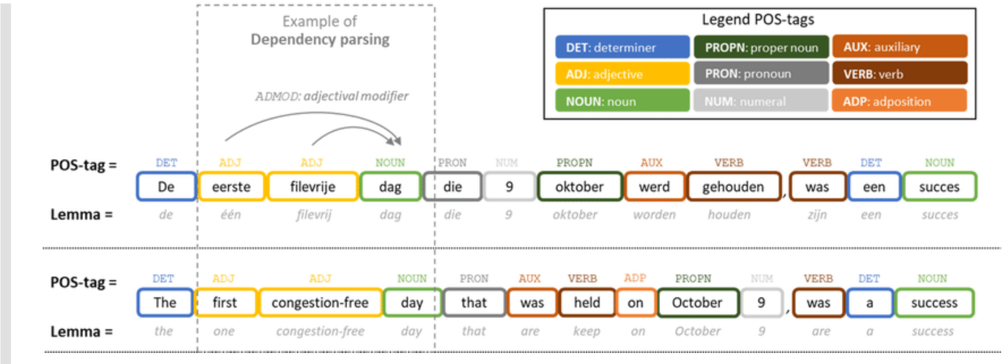
In [ ]:
```
ar = nlp.get_pipe("attribute_ruler")
```

```
ar.add([[{"TEXT":"Bro"}],[{"TEXT":"Brah"}]],{"LEMMA":"Brother"})

doc = nlp("Bro, you wanna go? Brah, don't say no! I am exhausted")

for token in doc:
  print(token, "     ", token.lemma_, "     ", token.lemma)
```

```
Bro       Brother      4347558510128575363
,       ,       2593208677638477497
you       you       7624161793554793053
wanna       wanna       13000462173222681081
go       go       8004577259940138793
?       ?       8205403955989537350
Brah       Brother       4347558510128575363
,       ,       2593208677638477497
do       do       2158845516055552166
n't       not       447765159362469301
say       say       8685289367999165211
no       no       13055779130471031426
!       !       17494803046312582752
I       I       4690420944186131903
am       be       10382539506755952630
exhausted       exhaust       5738807065439247694
```

Part Of Speech(POS) Tagging





In [ ]:
```
nlp = spacy.load("en_core_web_sm")
doc = nlp("Elon flew to mars yesterday. He carried biryani masala with him")

for token in doc:
    print(token,"   ", token.pos_, "    ", spacy.explain(token.pos_), "   ", token.
```

```
Elon     PROPN      proper noun     NNP      noun, proper singular
flew     VERB      verb     VBD      verb, past tense
to     ADP      adposition     IN      conjunction, subordinating or preposition
mars     NOUN      noun     NNS      noun, plural
yesterday     NOUN      noun     NN      noun, singular or mass
.     PUNCT      punctuation     .      punctuation mark, sentence closer
He     PRON      pronoun     PRP      pronoun, personal
carried     VERB      verb     VBD      verb, past tense
biryani     NOUN      noun     NN      noun, singular or mass
masala     NOUN      noun     NN      noun, singular or mass
with     ADP      adposition     IN      conjunction, subordinating or preposition
him     PRON      pronoun     PRP      pronoun, personal
```

In [ ]:
```
earnings_text="""Microsoft Corp. today announced the following results for the quar

•          Revenue was $51.7 billion and increased 20%
•          Operating income was $22.2 billion and increased 24%
•          Net income was $18.8 billion and increased 21%
•          Diluted earnings per share was $2.48 and increased 22%
“Digital technology is the most malleable resource at the world’s disposal to overc
“Solid commercial execution, represented by strong bookings growth driven by long-t

doc = nlp(earnings_text)

filted_tokens = []

for token in doc:
    if token.pos_ not in ["SACE", "PUNC"]:
        filted_tokens.append(token)

filted_tokens[:10]
```

Out[ ]:
```
[Microsoft,
 Corp.,
 today,
 announced,
 the,
 following,
 results,
 for,
 the,
 quarter]
```

In [ ]:
```
count = doc.count_by(spacy.attrs.POS)
count
```

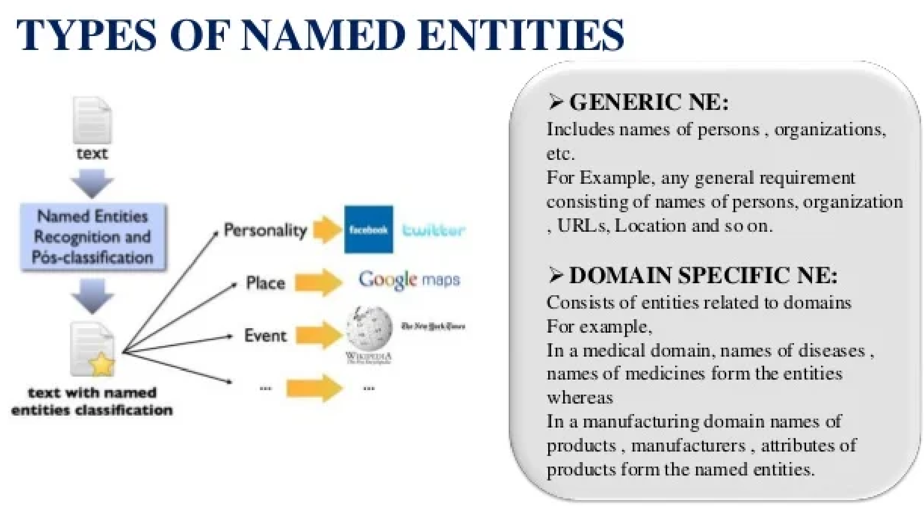
Out[ ]:
```
{96: 15,
 92: 45,
 100: 22,
 90: 9,
 85: 16,
 93: 16,
 97: 27,
 98: 1,
 84: 21,
 103: 10,
 87: 6,
 99: 5,
 89: 12,
 86: 3,
 94: 3,
 95: 2}
```

In [ ]:
```
for k, v in count.items():
    print(doc.vocab[k].text, "    ", v)
```

```
PROPN      15
NOUN      45
VERB      22
DET      9
ADP      16
NUM      16
PUNCT      27
SCONJ      1
ADJ      21
SPACE      10
AUX      6
SYM      5
CCONJ      12
ADV      3
PART      3
PRON      2
```

> ## Named Entity Recognition (NER)

> [Introduction to Named Entity Recognition]()
> [Hugging Face]()



In [ ]:
```python
import spacy
nlp = spacy.load("en_core_web_sm")
```

In [ ]:
```python
nlp.pipe_names
```

Out[ ]:
```
['tok2vec', 'tagger', 'parser', 'attribute_ruler', 'lemmatizer', 'ner']
```

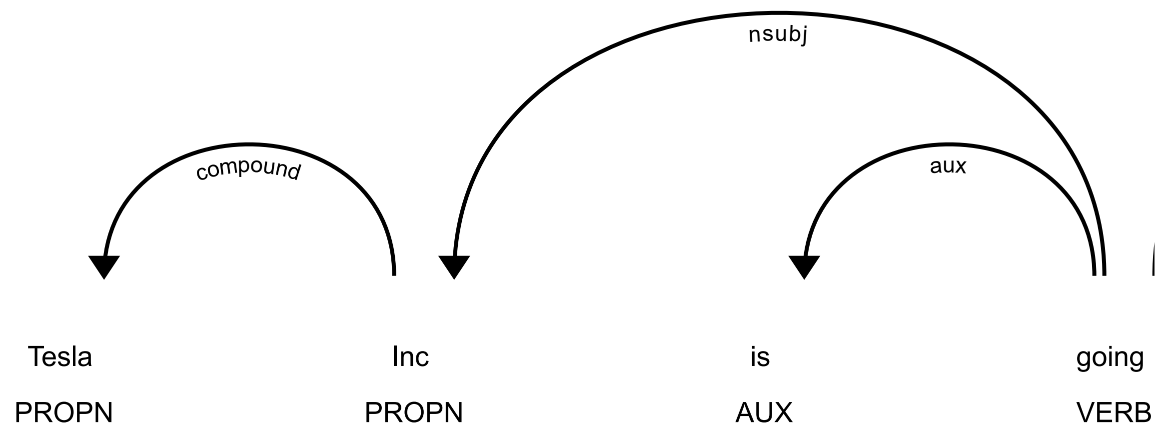
In [ ]:
```python
doc = nlp("Tesla Inc is going to acquire twitter for $45 billion")

for entity in doc.ents:
    print(entity.text, "    ", entity.label_, "    ", spacy.explain(entity.label_))
```

```
Tesla Inc      ORG     Companies, agencies, institutions, etc.
$45 billion      MONEY     Monetary values, including unit
```

In [ ]:
```python
from spacy import displacy

displacy.render(doc, style="ent")
```

Tesla Inc ORG is going to acquire twitter for $45 billion MONEY

In [ ]:
```python
from spacy import displacy

displacy.render(doc, style="dep")
```



In [ ]:
```
nlp.pipe_labels["ner"]
```

Out[ ]:
```
['CARDINAL',
 'DATE',
 'EVENT',
 'FAC',
 'GPE',
 'LANGUAGE',
 'LAW',
 'LOC',
 'MONEY',
 'NORP',
 'ORDINAL',
 'ORG',
 'PERCENT',
 'PERSON',
 'PRODUCT',
 'QUANTITY',
 'TIME',
 'WORK_OF_ART']
```

In [ ]:
```
doc = nlp("Michael Bloomberg founded Bloomberg in 1982")

for entity in doc.ents:
    print(entity.text, "    ", entity.label_, "   ", spacy.explain(entity.label_))
```

```
Michael Bloomberg      PERSON     People, including fictional
Bloomberg      GPE      Countries, cities, states
1982      DATE      Absolute or relative dates or periods
```

> ## Adding a Customizing NER Component to pipeline

In [ ]:
```
doc = nlp("Tesla is going to acquire Twitter for $45 billion")
```

```
for entity in doc.ents:
    print(entity.text, "    ", entity.label_, "   ", spacy.explain(entity.label_))
```

```
Twitter      PERSON     People, including fictional
$45 billion      MONEY      Monetary values, including unit
```

In [ ]:
```
type(doc[0])
```

Out[ ]:
```
spacy.tokens.token.Token
```

In [ ]:
```
type(doc[2:5])
```

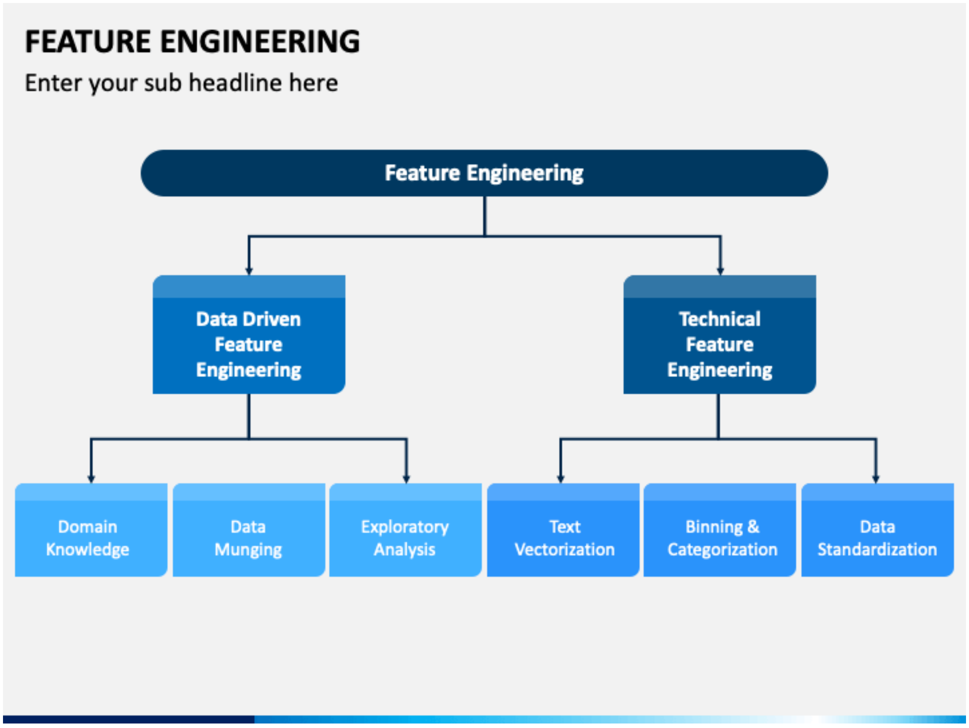
Out[ ]:
```
spacy.tokens.span.Span
```

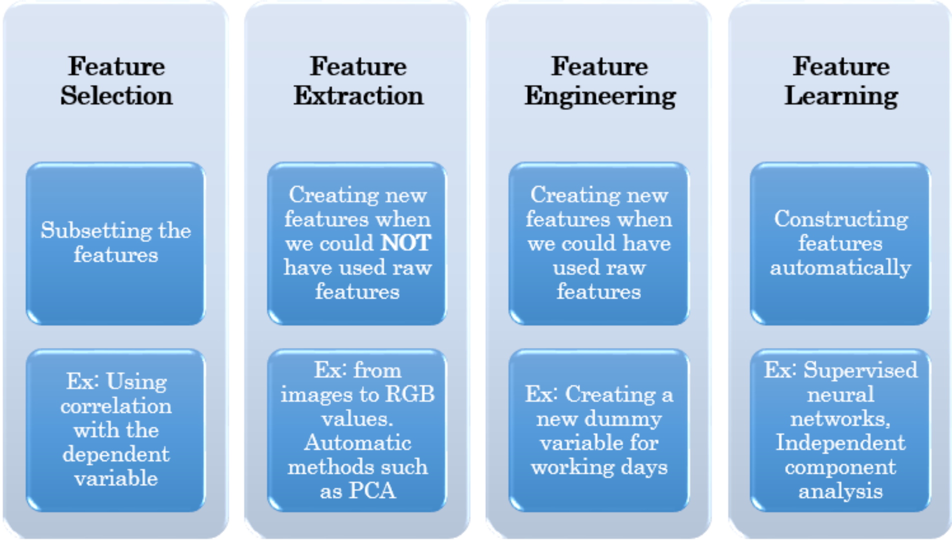
In [ ]:
```
from spacy.tokens import Span

s1 = Span(doc, 0, 1, label="ORG")
s2 = Span(doc, 5, 6, label="ORG")

doc.set_ents([s1, s2], default="unmodified")
```

In [ ]:
```
for entity in doc.ents:
    print(entity.text, "    ", entity.label_, "   ", spacy.explain(entity.label_))
```

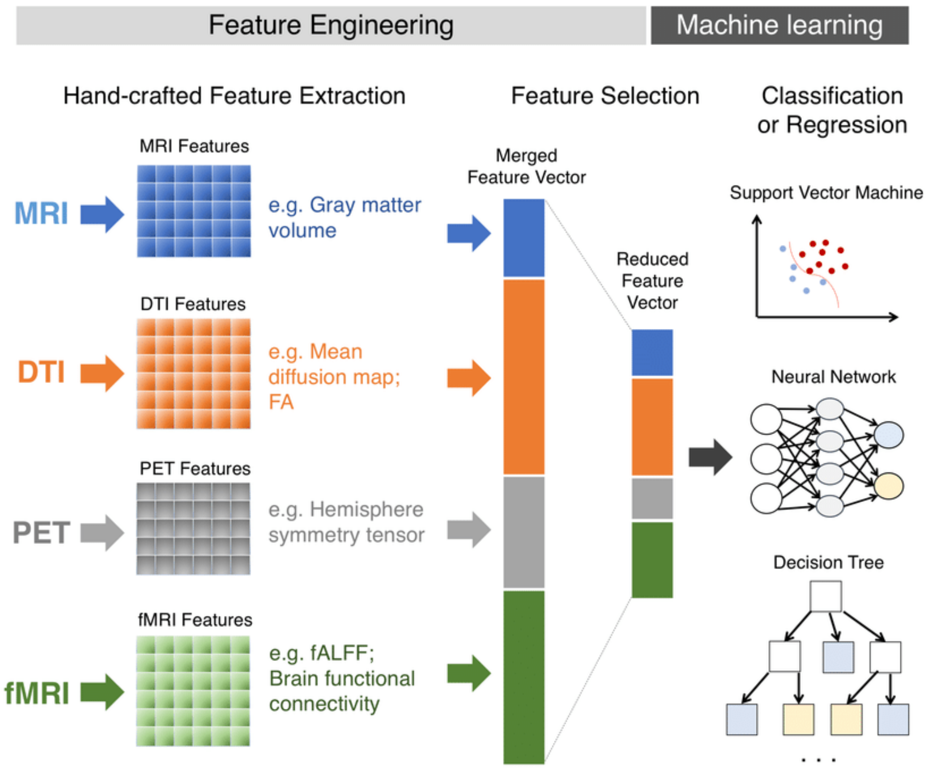
```
Tesla      ORG     Companies, agencies, institutions, etc.
Twitter      ORG     Companies, agencies, institutions, etc.
$45 billion      MONEY      Monetary values, including unit
```

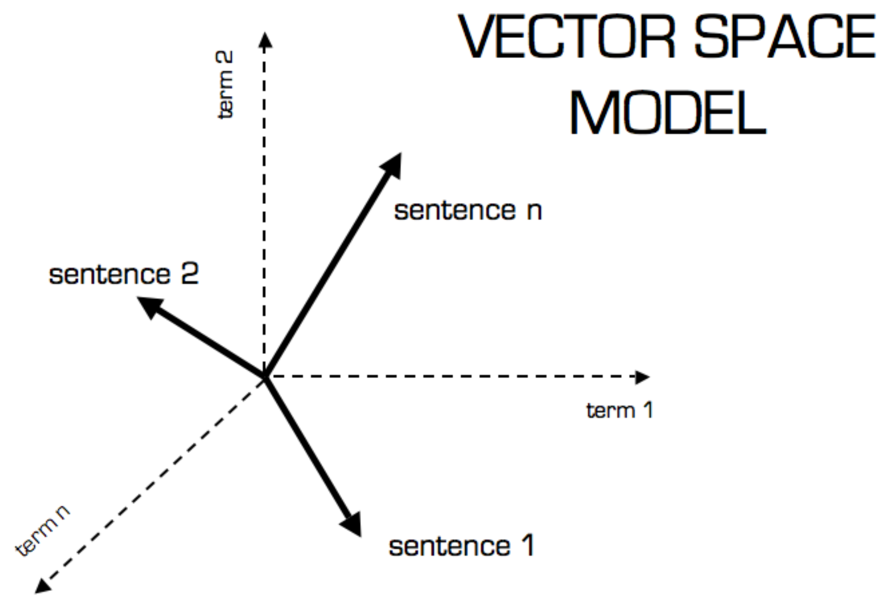
Feature Engineering





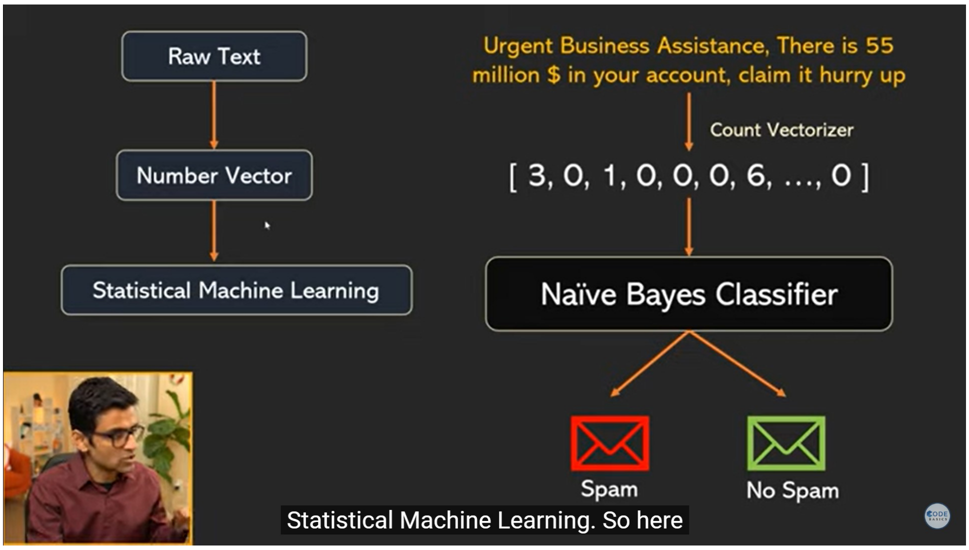
Convert Text to Vector

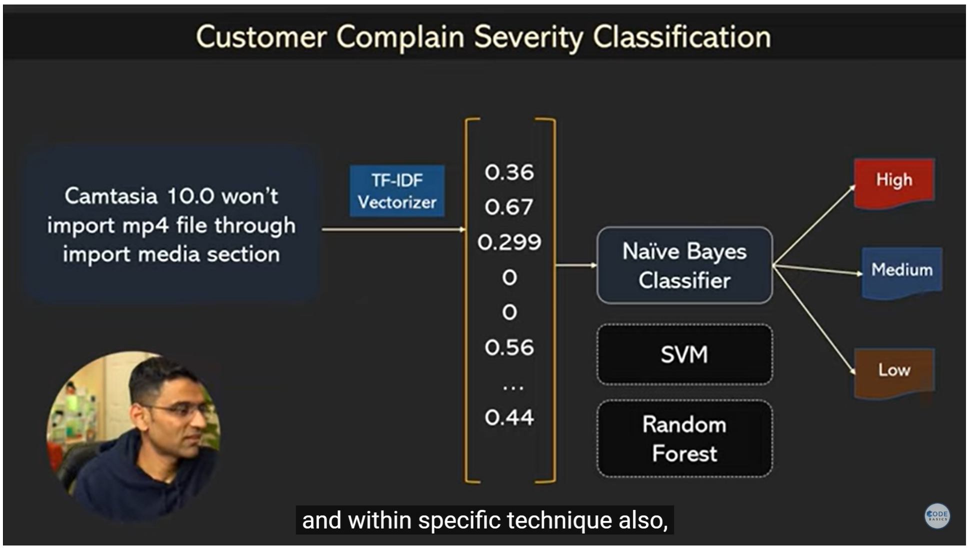



VECTOR SPACE MODEL
term 2
sentence n
sentence 2
term 1
term n
sentence 1

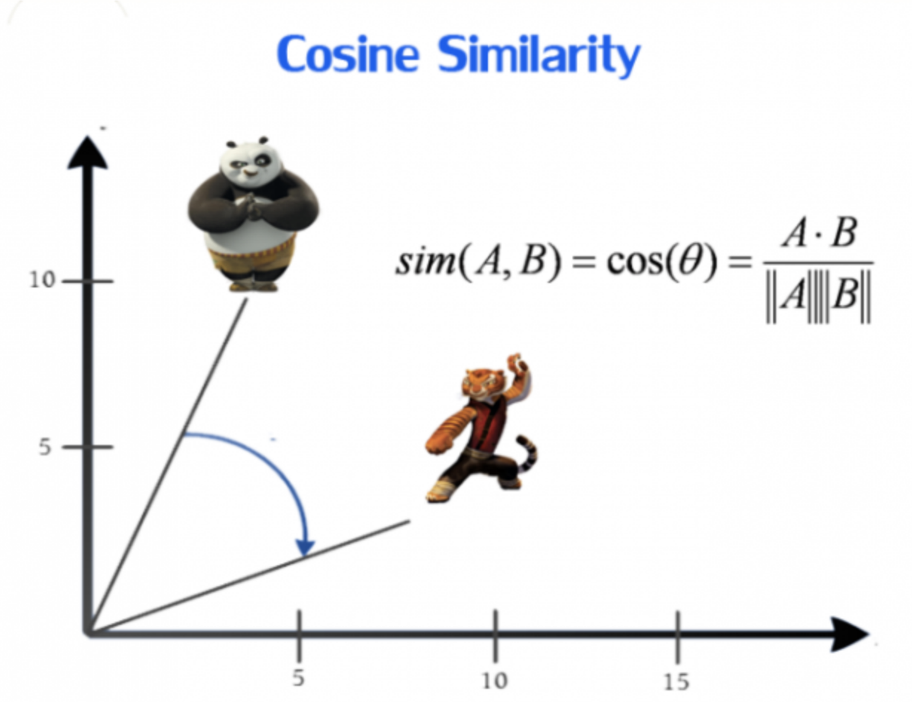
Text Classificatoin

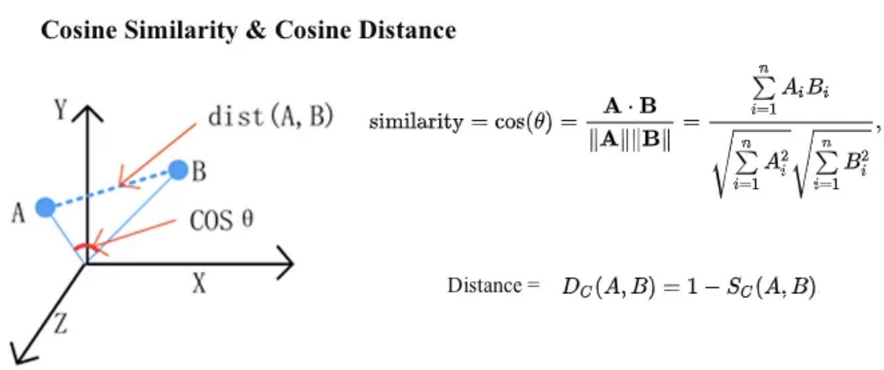
sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2





Cosine similarity, cosine distance explained





In [ ]:
```
from sklearn.metrics.pairwise import cosine_similarity, cosine_distances
```

In [ ]:
```
cosine_similarity([[3,1]],[[6,2]])
```

Out[ ]: `array([[1.]])`

In [ ]:
```
cosine_distances([[3,1]],[[6,2]])
```

Out[ ]: `array([[1.11022302e-16]])`

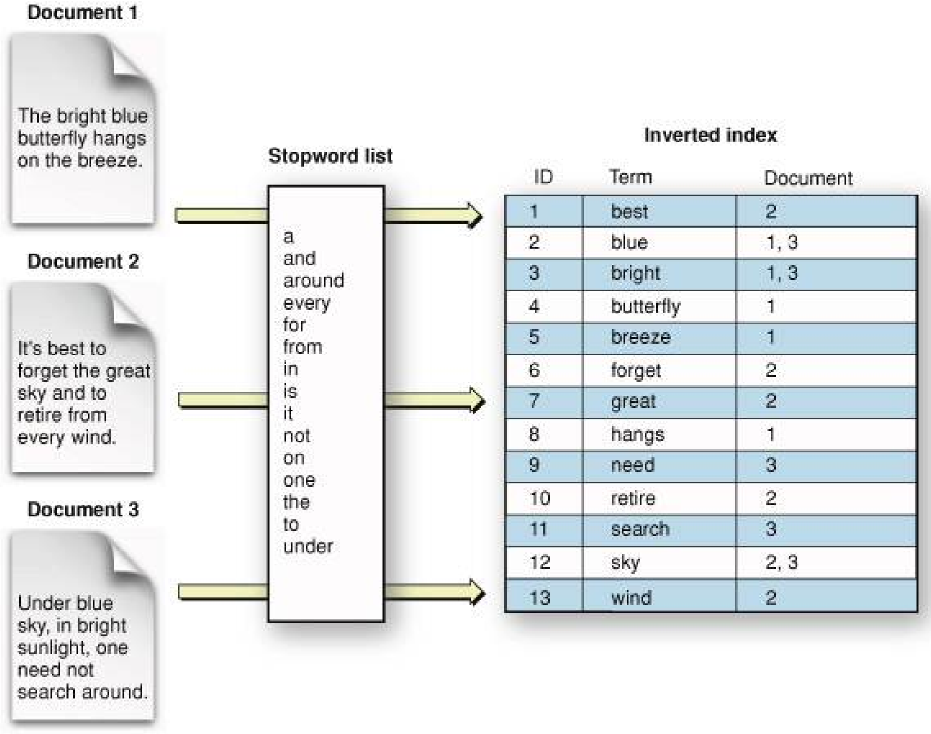
In [ ]:
```
cosine_similarity([[3,0]],[[0,8]])
```

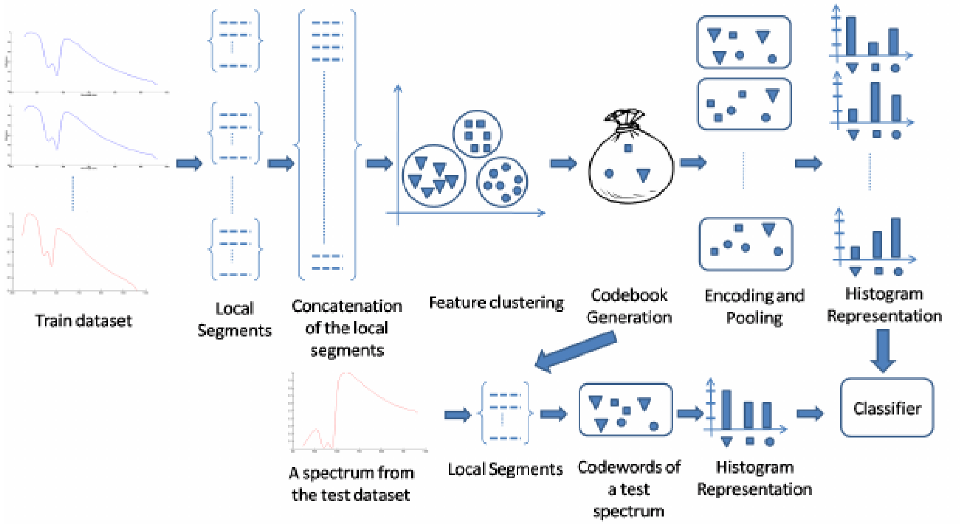
Out[ ]: `array([[0.]])`

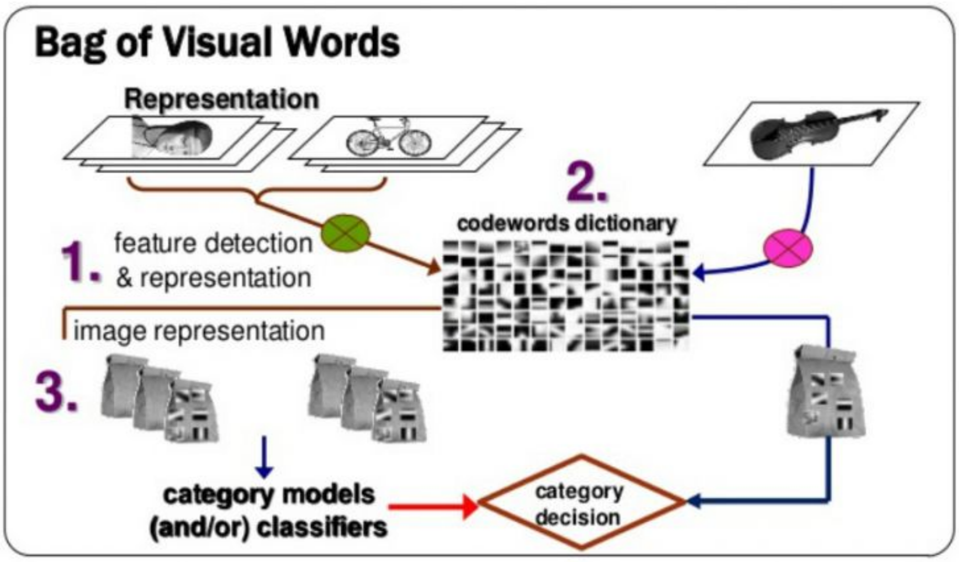
In [ ]:
```
cosine_distances([[3,0]],[[0,8]])
```

Out[ ]: `array([[1.]])`

Text Representation - Bag Of Words (BOW)








```
In [ ]: import numpy as np
        import pandas as pd
```

In [ ]:
```
df = pd.read_csv('data/spam.csv')
df.head()
```

Out[ ]:

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

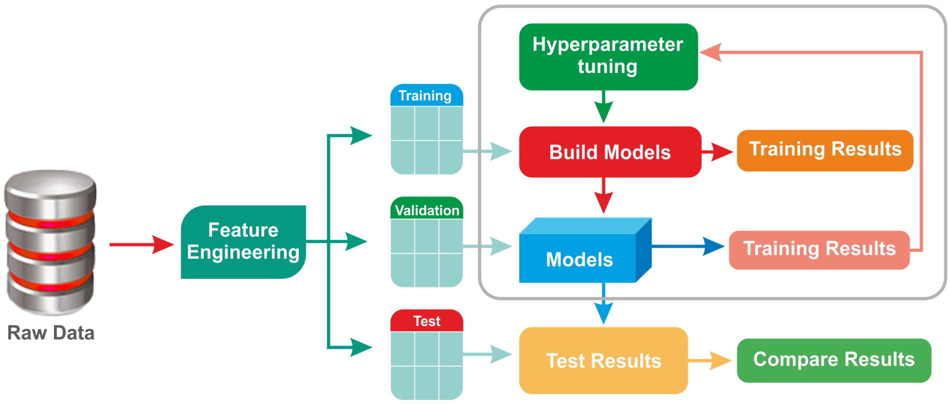
In [ ]:
```
df['Category'].value_counts()
```

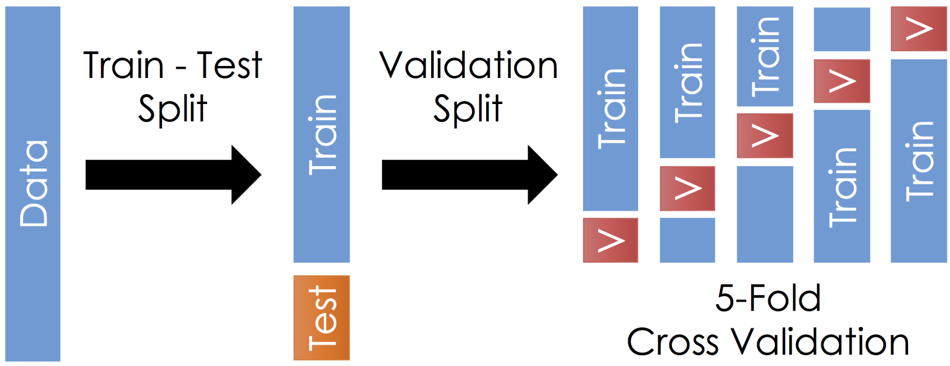
Out[ ]:
```
ham     4825
spam     747
Name: Category, dtype: int64
```

In [ ]:
```
df['Spam'] = df['Category'].apply(lambda x: 1 if x=='spam' else 0)
df.head()
```

Out[ ]:

	Category	Message	Spam
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

> ## Model Building and Training
>
> 
>



In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df['Message'], df['Spam'], test
```

In [ ]:
```
type(X_train)
```

Out[ ]: `pandas.core.series.Series`

In [ ]:
```
type(y_train)
```

Out[ ]: `pandas.core.series.Series`

In [ ]:
```
type(X_train.values)
```

Out[ ]: `numpy.ndarray`

In [ ]:
```
type(y_train.values)
```

Out[ ]: `numpy.ndarray`

Create bag of words representation using CountVectorizer

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer

v = CountVectorizer()

X_train_cv = v.fit_transform(X_train.values)
```

In [ ]:
```
type(X_train_cv)
```

Out[ ]: `scipy.sparse._csr.csr_matrix`

In [ ]:
```
X_train_cv.shape
```

Out[ ]: `(4457, 7799)`

In [ ]:
```
dir(v)
```

In [ ]:
```
v.vocabulary_
```

In [ ]:
```
v.get_feature_names_out()[1000:1010]
```

```
Out[ ]: array(['anything', 'anythingtomorrow', 'anytime', 'anyway', 'anyways',
               'anywhere', 'apart', 'apartment', 'apeshit', 'aphex'], dtype=object)
```

```
In [ ]: X_train_np = X_train_cv.toarray()
        X_train_np
```

```
Out[ ]: array([[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
In [ ]: type(X_train_np)
```

```
Out[ ]: numpy.ndarray
```

```
In [ ]: np.where(X_train_np[0] != 0)
```

```
Out[ ]: (array([3907, 4663, 4693, 4849, 5173, 5886, 6137, 6306, 6308, 6978, 7018,
                7262, 7541, 7696], dtype=int64),)
```

```
In [ ]: X_train_np[0][1054]
```

```
Out[ ]: 0
```

```
In [ ]: X_train[:4]
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_23912\2167669864.py:1: FutureWarning: T
he behavior of `series[i:j]` with an integer-dtype index is deprecated. In a futur
e version, this will be treated as *label-based* indexing, consistent with e.g. `s
eries[i]` lookups. To retain the old behavior, use `series.iloc[i:j]`. To get the
future behavior, use `series.loc[i:j]`.
  X_train[:4]
```

```
Out[ ]: 3292    I'm not smoking while people use "wylie smokes...
        723                                That is wondar full flim.
        5249        K I'm leaving soon, be there a little after 9
        5253    Please tell me not all of my car keys are in y...
        Name: Message, dtype: object
```

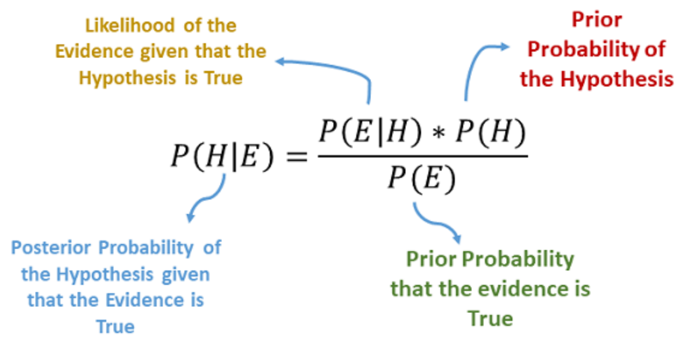
```
In [ ]: v.get_feature_names_out()[2333]
```

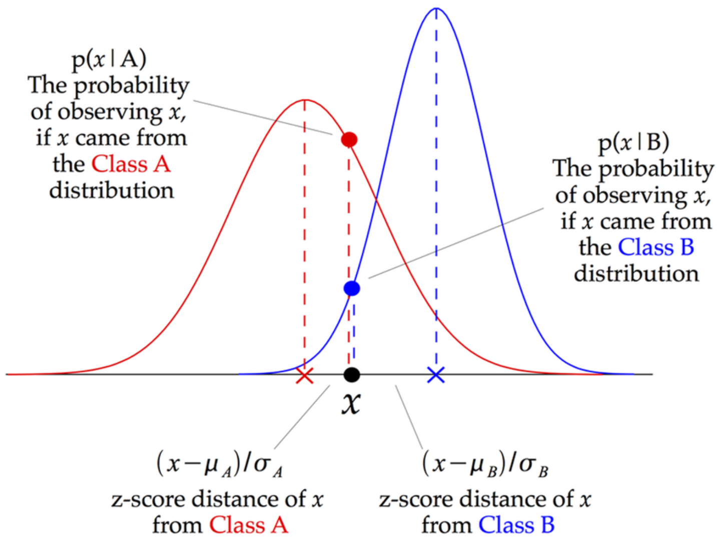
```
Out[ ]: 'different'
```

```
In [ ]: v.get_feature_names_out()[4539]
```

```
Out[ ]: 'missin'
```

Naive Bayes Classifier





In [ ]:
```
from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB()

model.fit(X_train_cv, y_train)
```

Out[ ]:
```
▾ MultinomialNB
MultinomialNB()
```

In [ ]:
```
X_test_cv = v.transform(X_test)
```

In [ ]:
```
y_preds = model.predict(X_test_cv)
```

In [ ]:
```
model.score(X_test_cv, y_test)
```

Out[ ]:
```
0.9847533632286996
```

In [ ]:
```
model.predict_proba(X_test_cv[:10])
```

Out[ ]:
```
array([[3.01293305e-02, 9.69870669e-01],
       [9.99410799e-01, 5.89201334e-04],
       [9.99999703e-01, 2.96768149e-07],
       [3.93373327e-11, 1.00000000e+00],
       [9.99999684e-01, 3.16156187e-07],
       [1.00000000e+00, 3.11023360e-15],
       [6.17426315e-15, 1.00000000e+00],
       [9.99999875e-01, 1.24821298e-07],
       [9.99999948e-01, 5.23724766e-08],
       [9.99999974e-01, 2.59966712e-08]])
```

In [ ]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       968
           1       0.97      0.91      0.94       147

    accuracy                           0.98      1115
   macro avg       0.98      0.95      0.97      1115
weighted avg       0.98      0.98      0.98      1115
```

In [ ]:
```
emails = [
    'Hey mohan, can we get together to watch footbal game tomorrow?',
    'Upto 20% discount on parking, exclusive offer just for you. Dont miss this rew
]

emails_cv = v.transform(emails)

model.predict(emails_cv)
```

Out[ ]:
```
array([0, 1], dtype=int64)
```

In [ ]:
```
from sklearn.pipeline import Pipeline

model = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('nb', MultinomialNB()),
])

model.fit(X_train, y_train)
```

Out[ ]:
```
Pipeline
  CountVectorizer
  MultinomialNB
```

In [ ]:
```
model.score(X_test, y_test)
```

Out[ ]:
```
0.9847533632286996
```

In [ ]:
```
from sklearn.metrics import classification_report

y_preds = model.predict(X_test)

print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       968
           1       0.97      0.91      0.94       147

    accuracy                           0.98      1115
   macro avg       0.98      0.95      0.97      1115
weighted avg       0.98      0.98      0.98      1115
```

Text Representation - Stop Words

In [ ]:
```
from spacy.lang.en.stop_words import STOP_WORDS
STOP_WORDS
```

In [ ]:
```
len(STOP_WORDS)
```

Out[ ]: 326

In [ ]:
```
nlp = spacy.load("en_core_web_sm")
```

In [ ]:
```
doc = nlp("We just opened our wings, the flying part is coming soon")

for token in doc:
    if not token.is_stop and not token.is_punct:
        print(token)
```

```
opened
wings
flying
coming
soon
```

In [ ]:
```
def preprocess(text):
    doc = nlp(text)
    no_stop_words = [token.text for token in doc if not token.is_stop and not toker
    return " ".join(no_stop_words)
```

In [ ]:
```
preprocess("We just opened our wings, the flying part is coming soon")
```

Out[ ]: 'opened wings flying coming soon'

In [ ]:
```
preprocess("Musk wants time to prepare for a trial over his")
```

Out[ ]: 'Musk wants time prepare trial'

In [ ]:
```
import numpy as np
import pandas as pd
```

In [ ]:
```
df = pd.read_json('data/doj_press.json', lines=True)
df.head()
```

Out[ ]:

	id	title	contents	date	topics	components
0	None	Convicted Bomb Plotter Sentenced to 30 Years	PORTLAND, Oregon. – Mohamed Osman Mohamud, 23,...	2014-10-01T00:00:00-04:00	[]	[National Security Division (NSD)]
1	12-919	$1 Million in Restitution Payments Announced t...	WASHINGTON – North Carolina's Waccamaw River...	2012-07-25T00:00:00-04:00	[]	[Environment and Natural Resources Division]
2	11-1002	$1 Million Settlement Reached for Natural Reso...	BOSTON– A $1-million settlement has been...	2011-08-03T00:00:00-04:00	[]	[Environment and Natural Resources Division]
3	10-015	10 Las Vegas Men Indicted \r\nfor Falsifying V...	WASHINGTON—A federal grand jury in Las Vegas...	2010-01-08T00:00:00-05:00	[]	[Environment and Natural Resources Division]
4	18-898	$100 Million Settlement Will Speed Cleanup Wor...	The U.S. Department of Justice, the U.S. Envir...	2018-07-09T00:00:00-04:00	[Environment]	[Environment and Natural Resources Division]

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13087 entries, 0 to 13086
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   id          12810 non-null  object
 1   title       13087 non-null  object
 2   contents    13087 non-null  object
 3   date        13087 non-null  object
 4   topics      13087 non-null  object
 5   components  13087 non-null  object
dtypes: object(6)
memory usage: 613.6+ KB
```

In [ ]:

```
df = df[ df['topics'].str.len() != 0]
df.head()
```

Out[ ]:

	id	title	contents	date	topics	components
4	18-898	$100 Million Settlement Will Speed Cleanup Wor...	The U.S. Department of Justice, the U.S. Envir...	2018-07-09T00:00:00-04:00	[Environment]	[Environment and Natural Resources Division]
7	14-1412	14 Indicted in Connection with New England Com...	A 131-count criminal indictment was unsealed t...	2014-12-17T00:00:00-05:00	[Consumer Protection]	[Civil Division]
19	17-1419	2017 Southeast Regional Animal Cruelty Prosecu...	The United States Attorney's Office for the Mi...	2017-12-14T00:00:00-05:00	[Environment]	[Environment and Natural Resources Division, U...
22	15-1562	21st Century Oncology to Pay $19.75 Million to...	21st Century Oncology LLC, has agreed to pay $...	2015-12-18T00:00:00-05:00	[False Claims Act, Health Care Fraud]	[Civil Division]
23	17-1404	21st Century Oncology to Pay $26 Million to Se...	21st Century Oncology Inc. and certain of its ...	2017-12-12T00:00:00-05:00	[Health Care Fraud, False Claims Act]	[Civil Division, USAO - Florida, Middle]

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4688 entries, 4 to 13086
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   id          4560 non-null   object
 1   title       4688 non-null   object
 2   contents    4688 non-null   object
 3   date        4688 non-null   object
 4   topics      4688 non-null   object
 5   components  4688 non-null   object
dtypes: object(6)
memory usage: 256.4+ KB
```

In [ ]:
```
df = df.head(100)
df["contents_new"] = df["contents"].apply(preprocess)
df.head()
```

Out[ ]:

	id	title	contents	date	topics	components	contents_new
4	18-898	$100 Million Settlement Will Speed Cleanup Wor...	The U.S. Department of Justice, the U.S. Envir...	2018-07-09T00:00:00-04:00	[Environment]	[Environment and Natural Resources Division]	U.S. Department Justice U.S. Environmental Pro...
7	14-1412	14 Indicted in Connection with New England Com...	A 131-count criminal indictment was unsealed t...	2014-12-17T00:00:00-05:00	[Consumer Protection]	[Civil Division]	131 count criminal indictment unsealed today B...
19	17-1419	2017 Southeast Regional Animal Cruelty Prosecu...	The United States Attorney's Office for the Mi...	2017-12-14T00:00:00-05:00	[Environment]	[Environment and Natural Resources Division, U...	United States Attorney Office Middle District ...
22	15-1562	21st Century Oncology to Pay $19.75 Million to...	21st Century Oncology LLC, has agreed to pay $...	2015-12-18T00:00:00-05:00	[False Claims Act, Health Care Fraud]	[Civil Division]	21st Century Oncology LLC agreed pay $ 19.75 m...
23	17-1404	21st Century Oncology to Pay $26 Million to Se...	21st Century Oncology Inc. and certain of its ...	2017-12-12T00:00:00-05:00	[Health Care Fraud, False Claims Act]	[Civil Division, USAO - Florida, Middle]	21st Century Oncology Inc. certain subsidiarie...

In [ ]:
```
len(preprocess(df['contents'].iloc[4]))
```

Out[ ]: 4217

In [ ]:
```
len(df['contents'].iloc[4])
```

Out[ ]: 5504

Text Representation - Bag Of n-grams

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer

v = CountVectorizer()

v.fit(["Thor Hathodawala is looking for a job"])
```

Out[ ]:
```
▾ CountVectorizer
CountVectorizer()
```

In [ ]:
```
v.vocabulary_
```

Out[ ]:
```
{'thor': 5, 'hathodawala': 1, 'is': 2, 'looking': 4, 'for': 0, 'job': 3}
```

In [ ]:
```
v = CountVectorizer(ngram_range=(2, 2))

v.fit(["Thor Hathodawala is looking for a job"])

v.vocabulary_
```

Out[ ]:
```
{'thor hathodawala': 4,
 'hathodawala is': 1,
 'is looking': 2,
 'looking for': 3,
 'for job': 0}
```

In [ ]:
```
import spacy

nlp = spacy.load("en_core_web_sm")

def preprocess(text):
    doc = nlp(text)
    filtered_tokens = []
    for token in doc:
        if token.is_stop or token.is_punct:
            continue
        filtered_tokens.append(token.lemma_)
    return " ".join(filtered_tokens)
```

In [ ]:
```
corpus = [
    "Thor ate pizza",
    "Loki is tall",
    "Loki is eating pizza"
]

corpus_processed = [preprocess(text) for text in corpus]
corpus_processed
```

Out[ ]:
```
['Thor eat pizza', 'Loki tall', 'Loki eat pizza']
```

In [ ]:
```
v = CountVectorizer(ngram_range=(1, 2))
v.fit(corpus_processed)
v.vocabulary_
```

Out[ ]:
```
{'thor': 7,
 'eat': 0,
 'pizza': 5,
 'thor eat': 8,
 'eat pizza': 1,
 'loki': 2,
 'tall': 6,
 'loki tall': 4,
 'loki eat': 3}
```

In [ ]:
```
v.transform(["Thor ate pizza"]).toarray()
```

Out[ ]:
```
array([[0, 0, 0, 0, 0, 1, 0, 1, 0]], dtype=int64)
```

Categories Classification

In [ ]:
```
import numpy as np
import pandas as pd
```

In [ ]:
```
df = pd.read_json('data/news_dataset.json')
df.head()
```

Out[ ]:

	text	category
0	Watching Schrödinger's Cat Die University of C...	SCIENCE
1	WATCH: Freaky Vortex Opens Up In Flooded Lake	SCIENCE
2	Entrepreneurs Today Don't Need a Big Budget to...	BUSINESS
3	These Roads Could Recharge Your Electric Car A...	BUSINESS
4	Civilian 'Guard' Fires Gun While 'Protecting' ...	CRIME

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12695 entries, 0 to 12694
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   text      12695 non-null  object
 1   category  12695 non-null  object
dtypes: object(2)
memory usage: 297.5+ KB
```

In [ ]:
```
df['category'].value_counts()
```

Out[ ]:
```
BUSINESS    4254
SPORTS      4167
CRIME       2893
SCIENCE     1381
Name: category, dtype: int64
```

In [ ]:
```
min_samples = 1381
df_BUSINESS = df[df['category']=="BUSINESS"].sample(min_samples, random_state=2022)
df_BUSINESS.shape
```

Out[ ]:
```
(1381, 2)
```

In [ ]:
```
df_SPORTS = df[df['category']=="SPORTS"].sample(min_samples, random_state=2022)
df_CRIME = df[df['category']=="CRIME"].sample(min_samples, random_state=2022)
df_SCIENCE = df[df['category']=="SCIENCE"].sample(min_samples, random_state=2022)
```

In [ ]:
```
df_balanced = pd.concat([df_BUSINESS, df_SPORTS, df_CRIME, df_SCIENCE], axis=0)
df_balanced['category'].value_counts()
```

Out[ ]:
```
BUSINESS    1381
SPORTS      1381
CRIME       1381
SCIENCE     1381
Name: category, dtype: int64
```

Model Build and Train

In [ ]:
```
df_balanced['category_new'] = df_balanced['category'].map({
    "BUSINESS": 0,
    "SPORTS" : 1,
    "CRIME": 3,
```

```
    "SCIENCE": 4,
})
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    df_balanced['text'],
    df_balanced['category_new'],
    test_size=0.2,
    random_state=2022,
    stratify=df_balanced['category_new']
)
```

In [ ]:
```
y_train.value_counts()
```

Out[ ]:
```
4    1105
3    1105
0    1105
1    1104
Name: category_new, dtype: int64
```

In [ ]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report

model = Pipeline([
    ('vectorizer_bow', CountVectorizer(ngram_range=(1, 2))),
    ('multi NB', MultinomialNB()),
])

model.fit(X_train, y_train)
```

Out[ ]:
```
▸ Pipeline
  ▸ CountVectorizer
  ▸ MultinomialNB
```

In [ ]:
```
model.score(X_test, y_test)
```

Out[ ]:
```
0.8244343891402715
```

In [ ]:
```
y_preds = model.predict(X_test)
```

In [ ]:
```
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.69      0.90      0.78       276
           1       0.95      0.74      0.83       277
           3       0.82      0.88      0.85       276
           4       0.92      0.78      0.84       276

    accuracy                           0.82      1105
   macro avg       0.85      0.82      0.83      1105
weighted avg       0.85      0.82      0.83      1105
```

Using preprocessed text to train

In [ ]:
```
df_balanced['text_processed'] = df_balanced['text'].apply(preprocess)
```

In [ ]:
```
df_balanced.head()
```

Out[ ]:

	text	category	category_new	text_processed
11967	GCC Business Leaders Remain Confident in the F...	BUSINESS	0	gcc Business leader remain Confident face Regi...
2912	From the Other Side; an Honest Review from Emp...	BUSINESS	0	Honest Review employee wake morning love impor...
3408	Mike McDerment, CEO of FreshBooks, Talks About...	BUSINESS	0	Mike McDerment ceo FreshBooks talk give build ...
502	How to Market Your Business While Traveling th...	BUSINESS	0	market business travel World recently amazing ...
5279	How to Leverage Intuition in Decision-making I...	BUSINESS	0	Leverage intuition decision making feel safe r...

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    df_balanced['text_processed'],
    df_balanced['category_new'],
    test_size=0.2,
    random_state=2022,
    stratify=df_balanced['category_new']
)
```

In [ ]:
```
model = Pipeline([
    ('vectorizer_bow', CountVectorizer(ngram_range=(1, 2))),
    ('multi NB', MultinomialNB()),
])

model.fit(X_train, y_train)
```

Out[ ]:

```
▸ Pipeline
  ▸ CountVectorizer
  ▸ MultinomialNB
```

In [ ]:
```
model.score(X_test, y_test)
```

Out[ ]:
```
0.860633484162896
```

In [ ]:
```
y_preds = model.predict(X_test)
print(classification_report(y_test, y_preds))
```

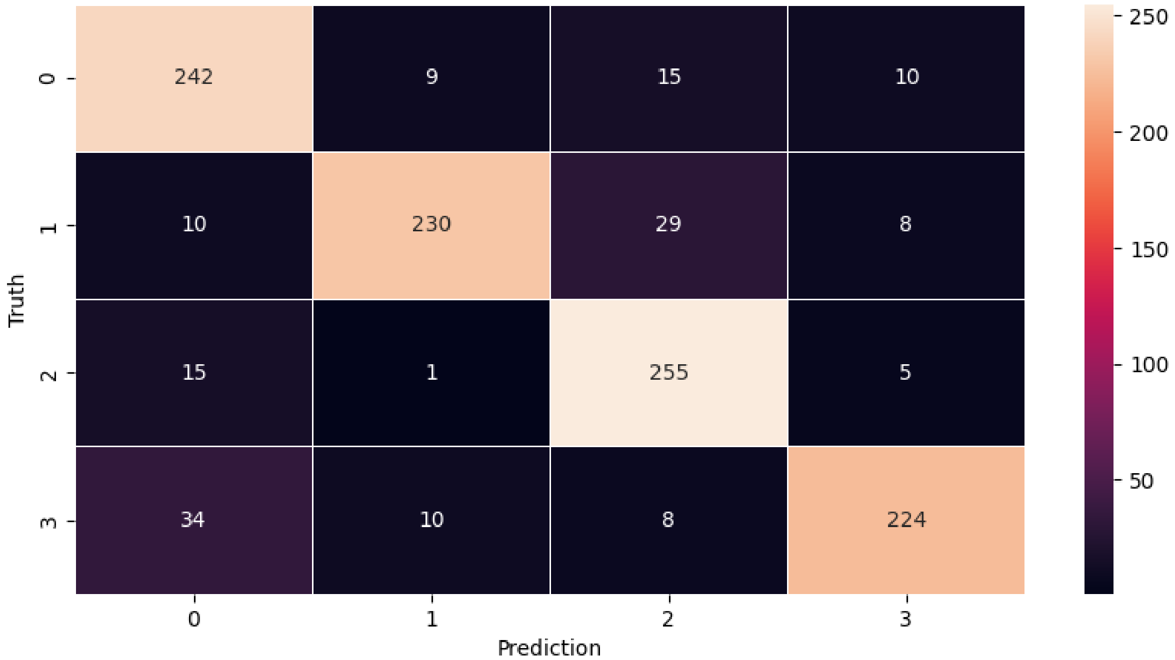
```
              precision    recall  f1-score   support

           0       0.80      0.88      0.84       276
           1       0.92      0.83      0.87       277
           3       0.83      0.92      0.87       276
           4       0.91      0.81      0.86       276

    accuracy                           0.86      1105
   macro avg       0.87      0.86      0.86      1105
weighted avg       0.87      0.86      0.86      1105
```

In [ ]:
```python
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_preds)
cm
```

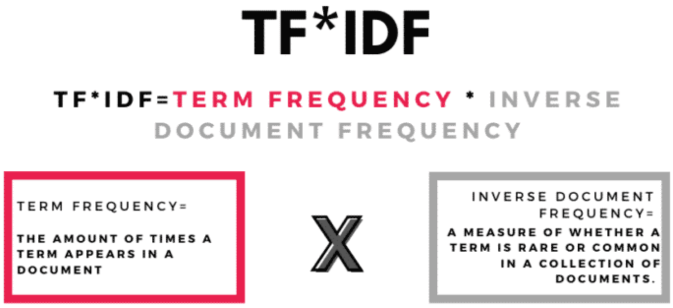
Out[ ]:
```
array([[242,   9,  15,  10],
       [ 10, 230,  29,   8],
       [ 15,   1, 255,   5],
       [ 34,  10,   8, 224]], dtype=int64)
```

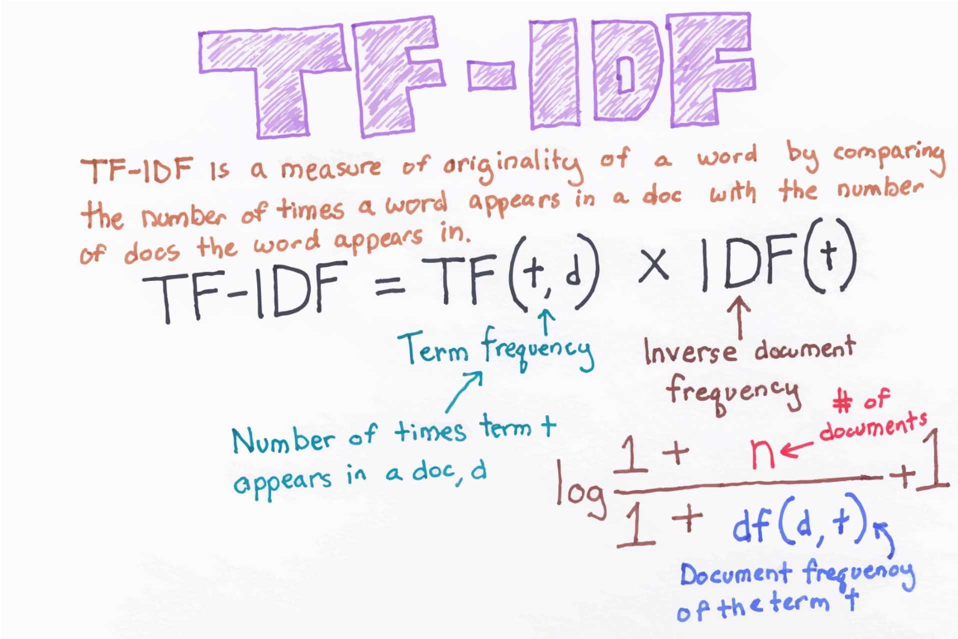
In [ ]:
```python
import matplotlib.pyplot as plt
import seaborn as sn

plt.figure(figsize=(10,5))
sn.heatmap(
    cm,
    annot=True,
    fmt='d',
    linewidth=0.5
)
plt.xlabel('Prediction')
plt.ylabel('Truth')
```

Out[ ]: Text(95.72222222222221, 0.5, 'Truth')



Text Representation - TF-IDF






```
In [ ]: from sklearn.feature_extraction.text import TfidfVectorizer

        v = TfidfVectorizer()

        dir(v)
```

```
In [ ]: corpus = [
            "Thor eating pizza, Loki is eating pizza, Ironman ate pizza already",
            "Apple is announcing new iphone tomorrow",
            "Tesla is announcing new model-3 tomorrow",
            "Google is announcing new pixel-6 tomorrow",
            "Microsoft is announcing new surface tomorrow",
            "Amazon is announcing new eco-dot tomorrow",
            "I am eating biryani and you are eating grapes"
        ]
```

```
In [ ]: output_transformed = v.fit_transform(corpus)
```

```
In [ ]: type(output_transformed)
```

```
Out[ ]: scipy.sparse._csr.csr_matrix
```

```
In [ ]: v.vocabulary_
```

Out[ ]:
```
{'thor': 25,
 'eating': 10,
 'pizza': 22,
 'loki': 17,
 'is': 16,
 'ironman': 15,
 'ate': 7,
 'already': 0,
 'apple': 5,
 'announcing': 4,
 'new': 20,
 'iphone': 14,
 'tomorrow': 26,
 'tesla': 24,
 'model': 19,
 'google': 12,
 'pixel': 21,
 'microsoft': 18,
 'surface': 23,
 'amazon': 2,
 'eco': 11,
 'dot': 9,
 'am': 1,
 'biryani': 8,
 'and': 3,
 'you': 27,
 'are': 6,
 'grapes': 13}
```

In [ ]:
```
all_feature_names = v.get_feature_names_out()
all_feature_names
```

Out[ ]:
```
array(['already', 'am', 'amazon', 'and', 'announcing', 'apple', 'are',
       'ate', 'biryani', 'dot', 'eating', 'eco', 'google', 'grapes',
       'iphone', 'ironman', 'is', 'loki', 'microsoft', 'model', 'new',
       'pixel', 'pizza', 'surface', 'tesla', 'thor', 'tomorrow', 'you'],
      dtype=object)
```

In [ ]:
```
v.idf_
```

Out[ ]:
```
array([2.38629436, 2.38629436, 2.38629436, 2.38629436, 1.28768207,
       2.38629436, 2.38629436, 2.38629436, 2.38629436, 2.38629436,
       1.98082925, 2.38629436, 2.38629436, 2.38629436, 2.38629436,
       2.38629436, 1.13353139, 2.38629436, 2.38629436, 2.38629436,
       1.28768207, 2.38629436, 2.38629436, 2.38629436, 2.38629436,
       2.38629436, 1.28768207, 2.38629436])
```

In [ ]:
```
for word in all_feature_names:
    index = v.vocabulary_.get(word)
    print(f"{word}:  {v.idf_[index]}")
```

```
already:  2.386294361119891
am:  2.386294361119891
amazon:  2.386294361119891
and:  2.386294361119891
announcing:  1.2876820724517808
apple:  2.386294361119891
are:  2.386294361119891
ate:  2.386294361119891
biryani:  2.386294361119891
dot:  2.386294361119891
eating:  1.9808292530117262
eco:  2.386294361119891
google:  2.386294361119891
grapes:  2.386294361119891
iphone:  2.386294361119891
ironman:  2.386294361119891
is:  1.1335313926245225
loki:  2.386294361119891
microsoft:  2.386294361119891
model:  2.386294361119891
new:  1.2876820724517808
pixel:  2.386294361119891
pizza:  2.386294361119891
surface:  2.386294361119891
tesla:  2.386294361119891
thor:  2.386294361119891
tomorrow:  1.2876820724517808
you:  2.386294361119891
```

In [ ]:
```
output_transformed[:2]
```

Out[ ]:
```
<2x28 sparse matrix of type '<class 'numpy.float64'>'
	with 14 stored elements in Compressed Sparse Row format>
```

In [ ]:
```
output_transformed.toarray()[:2]
```

Out[ ]:
```
array([[0.24266547, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.24266547, 0.        , 0.        ,
        0.40286636, 0.        , 0.        , 0.        , 0.        ,
        0.24266547, 0.11527033, 0.24266547, 0.        , 0.        ,
        0.        , 0.        , 0.72799642, 0.        , 0.        ,
        0.24266547, 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.30652086,
        0.5680354 , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.5680354 ,
        0.        , 0.26982671, 0.        , 0.        , 0.        ,
        0.30652086, 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.30652086, 0.        ]])
```

In [ ]:
```
corpus[:2]
```

Out[ ]:
```
['Thor eating pizza, Loki is eating pizza, Ironman ate pizza already',
 'Apple is announcing new iphone tomorrow']
```

In [ ]:
```
df = pd.read_csv("data/Ecommerce_data.csv")
df.head()
```

Out[ ]:

	Text	label
0	Urban Ladder Eisner Low Back Study-Office Comp...	Household
1	Contrast living Wooden Decorative Box,Painted ...	Household
2	IO Crest SY-PCI40010 PCI RAID Host Controller ...	Electronics
3	ISAKAA Baby Socks from Just Born to 8 Years- P...	Clothing & Accessories
4	Indira Designer Women's Art Mysore Silk Saree ...	Clothing & Accessories

In [ ]:
```
df['label'].value_counts()
```

Out[ ]:
```
Household                 6000
Electronics               6000
Clothing & Accessories    6000
Books                     6000
Name: label, dtype: int64
```

In [ ]:
```
df['label_num'] = df['label'].map({
    'Household' : 0,
    'Books': 1,
    'Electronics': 2,
    'Clothing & Accessories': 3
})
df.head()
```

Out[ ]:

	Text	label	label_num
0	Urban Ladder Eisner Low Back Study-Office Comp...	Household	0
1	Contrast living Wooden Decorative Box,Painted ...	Household	0
2	IO Crest SY-PCI40010 PCI RAID Host Controller ...	Electronics	2
3	ISAKAA Baby Socks from Just Born to 8 Years- P...	Clothing & Accessories	3
4	Indira Designer Women's Art Mysore Silk Saree ...	Clothing & Accessories	3

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(
    df['Text'],
    df['label_num'],
    test_size=0.2, # 20% samples will go to test dataset
    random_state=2022,
    stratify=df.label_num
)
```

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline

model = Pipeline([
    ('vectorizer_tfidf', TfidfVectorizer()),
    ('KNN', KNeighborsClassifier()),
])

model.fit(X_train, y_train)
```

Out[ ]:

```
Pipeline
  TfidfVectorizer
  KNeighborsClassifier
```

In [ ]:

```
model.score(X_test, y_test)
```

Out[ ]:

```
0.9641666666666666
```

In [ ]:

```
from sklearn.metrics import classification_report

y_preds = model.predict(X_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      1200
           1       0.97      0.95      0.96      1200
           2       0.97      0.97      0.97      1200
           3       0.97      0.98      0.97      1200

    accuracy                           0.96      4800
   macro avg       0.96      0.96      0.96      4800
weighted avg       0.96      0.96      0.96      4800
```

In [ ]:

```
y_test[:5]
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_23912\1754177261.py:1: FutureWarning: T
he behavior of `series[i:j]` with an integer-dtype index is deprecated. In a futur
e version, this will be treated as *label-based* indexing, consistent with e.g. `s
eries[i]` lookups. To retain the old behavior, use `series.iloc[i:j]`. To get the
future behavior, use `series.loc[i:j]`.
  y_test[:5]
```

Out[ ]:

```
20706    0
19166    2
15209    3
2462     1
6621     3
Name: label_num, dtype: int64
```

In [ ]:

```
y_preds[:5]
```

Out[ ]:

```
array([0, 2, 3, 1, 0], dtype=int64)
```

In [ ]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline

model = Pipeline([
    ('vectorizer_tfidf', TfidfVectorizer()),
    ('Multi NB', MultinomialNB()),
])

model.fit(X_train, y_train)
```

Out[ ]:
```
Pipeline
  TfidfVectorizer
  MultinomialNB
```

In [ ]:
```
model.score(X_test, y_test)
```

Out[ ]: 0.9602083333333333

In [ ]:
```
from sklearn.metrics import classification_report

y_preds = model.predict(X_test)
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      1200
           1       0.98      0.92      0.95      1200
           2       0.97      0.97      0.97      1200
           3       0.97      0.99      0.98      1200

    accuracy                           0.96      4800
   macro avg       0.96      0.96      0.96      4800
weighted avg       0.96      0.96      0.96      4800
```

In [ ]:
```
df['Text_processed'] = df['Text'].apply(preprocess)
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(
    df['Text_processed'],
    df['label_num'],
    test_size=0.2, # 20% samples will go to test dataset
    random_state=2022,
    stratify=df.label_num
)
```

In [ ]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline

model = Pipeline([
    ('vectorizer_tfidf', TfidfVectorizer()),
    ('Multi NB', MultinomialNB()),
])

model.fit(X_train, y_train)
```

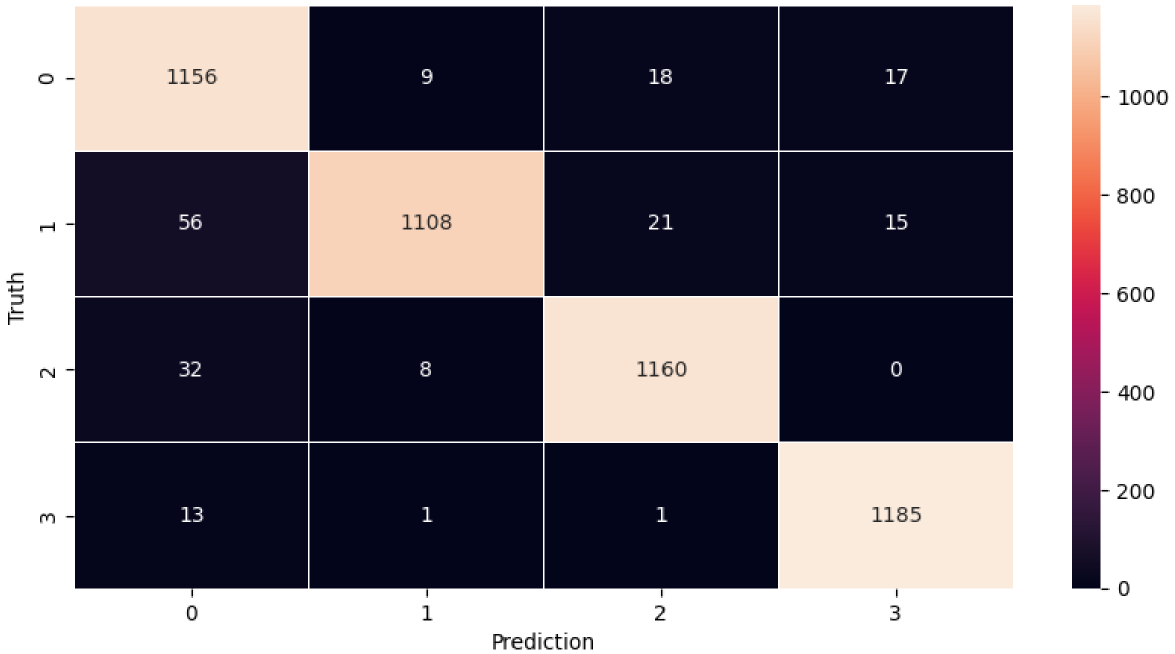
Out[ ]:
```
Pipeline
  TfidfVectorizer
  MultinomialNB
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_preds)
cm
```

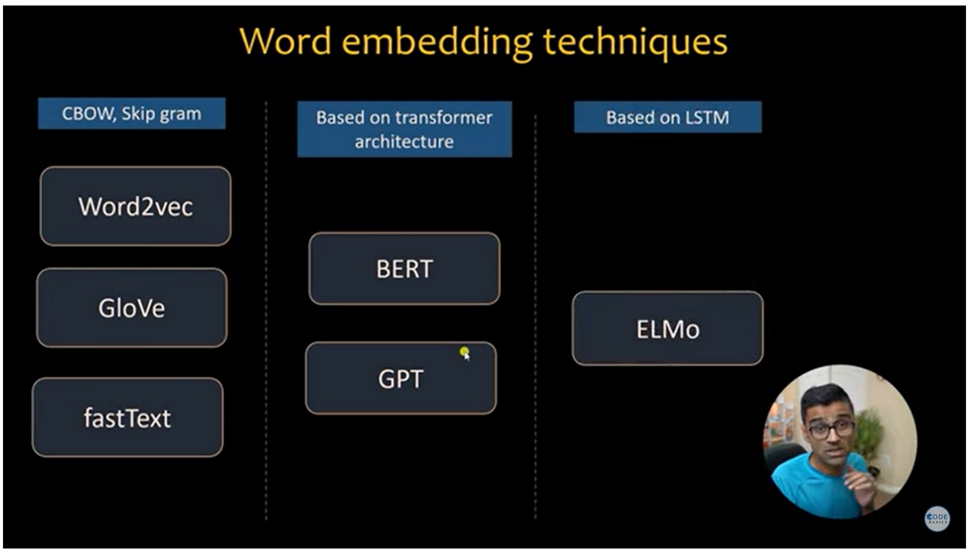
```
Out[ ]: array([[1156,    9,   18,   17],
               [  56, 1108,   21,   15],
               [  32,    8, 1160,    0],
               [  13,    1,    1, 1185]], dtype=int64)
```

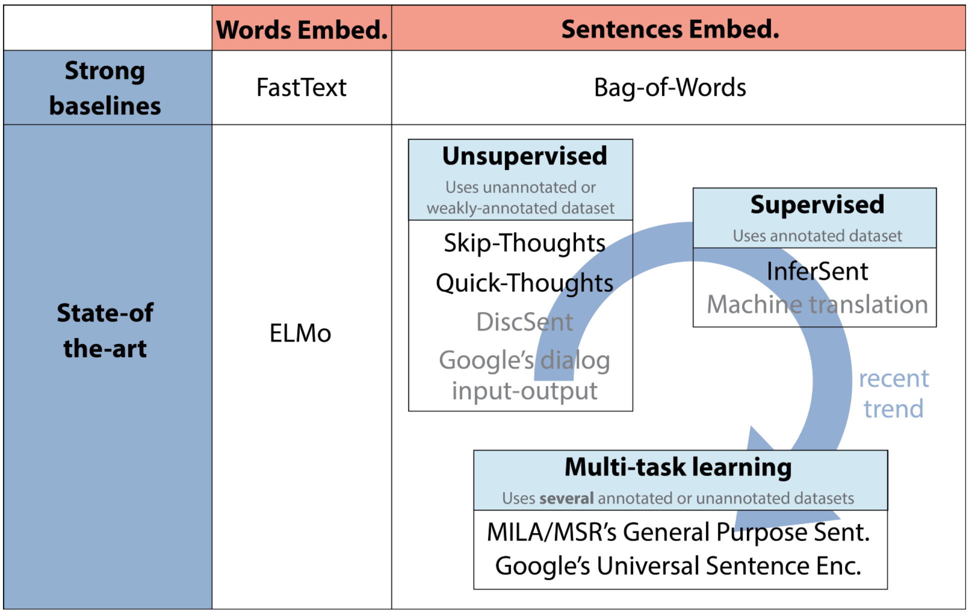
```
In [ ]: import matplotlib.pyplot as plt
        import seaborn as sn

        plt.figure(figsize=(10,5))
        sn.heatmap(
            cm,
            annot=True,
            fmt='d',
            linewidth=0.5
        )
        plt.xlabel('Prediction')
        plt.ylabel('Truth')
```

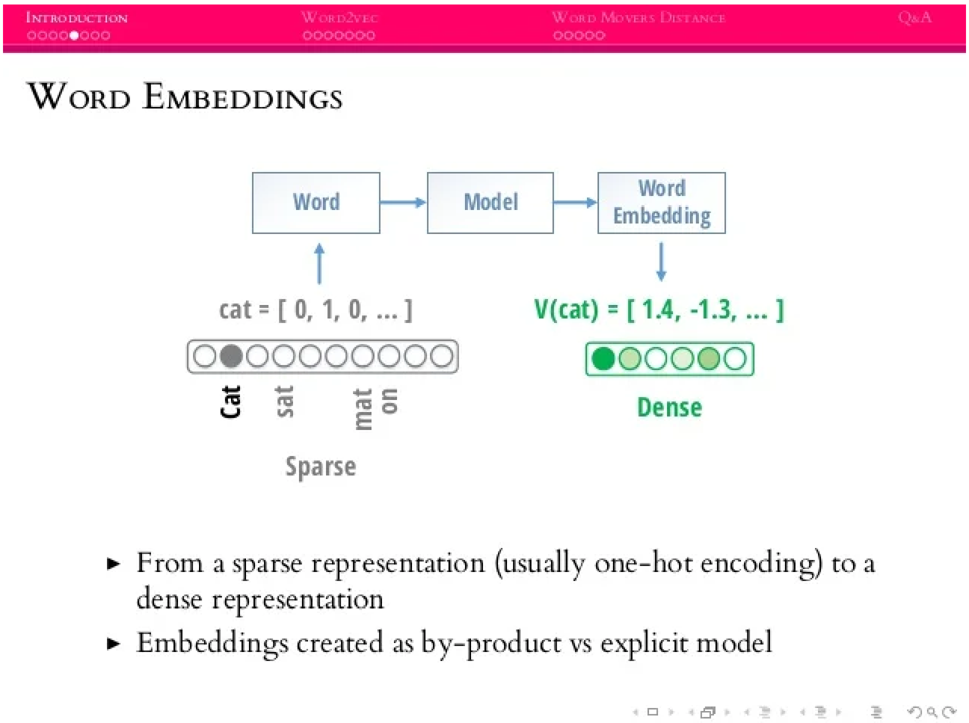
```
Out[ ]: Text(95.72222222222221, 0.5, 'Truth')
```



Text Representation - Word Embeddings







!python -m spacy download en_core_web_lg

In [ ]:
```
nlp = spacy.load("en_core_web_lg")
```

In [ ]:
```
doc = nlp("dog cat banana abcde")

for token in doc:
    print (token, " Vector: ", token.has_vector, " OOV: ", token.is_oov)
```

```
dog  Vector:  True  OOV:  False
cat  Vector:  True  OOV:  False
banana  Vector:  True  OOV:  False
abcde  Vector:  False  OOV:  True
```

In [ ]:
```
doc[0].vector.shape
```

Out[ ]: (300,)

In [ ]:
```
token_base = nlp("bread")
token_base.vector.shape
```

Out[ ]:
```
(300,)
```

In [ ]:
```
doc = nlp("bread sandwich burger car tiger human wheat")

for token in doc:
    print(f"{token.text} <-> {token_base.text}, Similarity:", token.similarity(toke
```

```
bread <-> bread, Similarity: 1.0
sandwich <-> bread, Similarity: 0.6341067010130894
burger <-> bread, Similarity: 0.47520687769584247
car <-> bread, Similarity: 0.06451533308853552
tiger <-> bread, Similarity: 0.04764611675903374
human <-> bread, Similarity: 0.2151154210812192
wheat <-> bread, Similarity: 0.6150360888607199
```

In [ ]:
```
def print_similarity(word_base, words_to_compare):
    token_base = nlp(word_base)
    doc = nlp(words_to_compare)
    for token in doc:
        print(f"{token.text} <-> {token_base.text}, Similarity:", token.similarity(
```

In [ ]:
```
print_similarity("iphone", "apple samsung iphone dog kitten")
```

```
apple <-> iphone, Similarity: 0.4387907401919904
samsung <-> iphone, Similarity: 0.670859081425417
iphone <-> iphone, Similarity: 1.0
dog <-> iphone, Similarity: 0.08211864228011527
kitten <-> iphone, Similarity: 0.10222317834969896
```

In [ ]:
```
king = nlp.vocab["king"].vector
man = nlp.vocab["man"].vector
woman = nlp.vocab["woman"].vector
queen = nlp.vocab["queen"].vector

result = king - man + woman
```

In [ ]:
```
from sklearn.metrics.pairwise import cosine_similarity

cosine_similarity([result], [queen])
```

Out[ ]:
```
array([[0.61780137]], dtype=float32)
```

Text Classification Using Spacy Word Embeddings

In [ ]:
```
import numpy as np
import pandas as pd
```

In [ ]:
```
df = pd.read_csv("data/Fake_Real_Data.csv")
df.head()
```

Out[ ]:

	Text	label
0	Top Trump Surrogate BRUTALLY Stabs Him In The...	Fake
1	U.S. conservative leader optimistic of common ...	Real
2	Trump proposes U.S. tax overhaul, stirs concer...	Real
3	Court Forces Ohio To Allow Millions Of Illega...	Fake
4	Democrats say Trump agrees to work on immigrat...	Real

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9900 entries, 0 to 9899
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Text    9900 non-null   object
 1   label   9900 non-null   object
dtypes: object(2)
memory usage: 154.8+ KB
```

In [ ]:
```
df['label'].value_counts()
```

Out[ ]:
```
Fake    5000
Real    4900
Name: label, dtype: int64
```

In [ ]:
```
df['label_num'] = df['label'].map({'Fake': 0, 'Real': 1})
```

In [ ]:
```
df.sample(5)
```

Out[ ]:

	Text	label	label_num
8961	Gala glitz masks Asia's tensions as Trump wind...	Real	1
3230	Kellyanne Conway Announces Trump's HUGE 'Than...	Fake	0
2944	Trump Boasts About Opening Of First New Coal ...	Fake	0
8488	U.S. taxpayers rush to claim deductions under ...	Real	1
1661	Trump faces obstacles in bid to re-shape key U...	Real	1

In [ ]:
```
nlp = spacy.load("en_core_web_lg")
```

In [ ]:
```
df['vector'] = df['Text'].apply(lambda text: nlp(text).vector)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9900 entries, 0 to 9899
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Text       9900 non-null   object
 1   label      9900 non-null   object
 2   label_num  9900 non-null   int64 
 3   vector     9900 non-null   object
dtypes: int64(1), object(3)
memory usage: 309.5+ KB
```

Train-Test splitting

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    df['vector'].values,
    df['label_num'],
    test_size=0.2,
    random_state=2022,
)
```

In [ ]:
```
X_train[0].shape
```

Out[ ]:
```
(300,)
```

Reshaping the X_train and X_test so as to fit for models

In [ ]:
```
X_train_2d = np.stack(X_train)
X_test_2d = np.stack(X_test)
```

In [ ]:
```
X_train_2d.shape
```

Out[ ]:
```
(7920, 300)
```

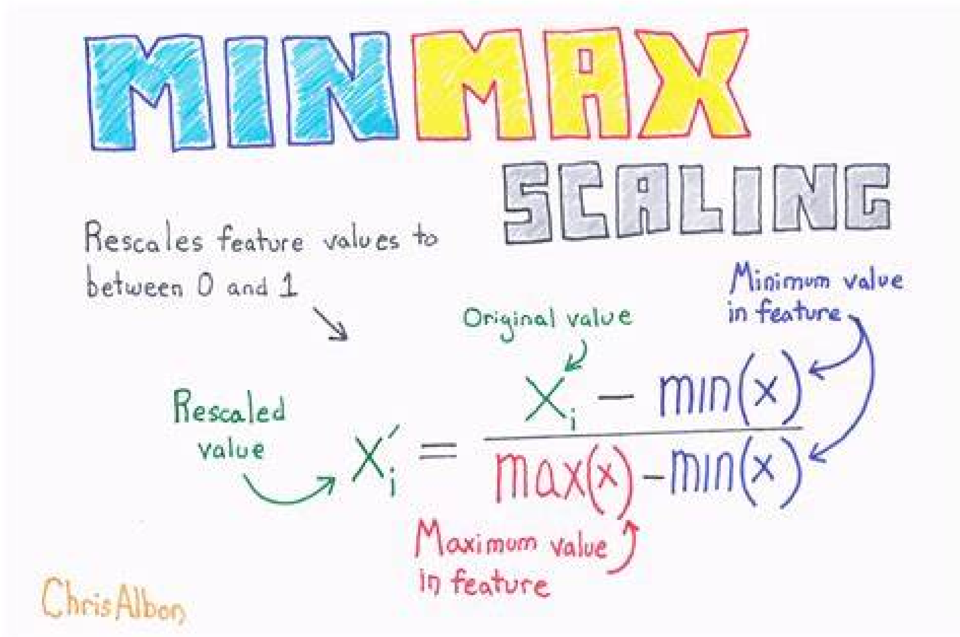
In [ ]:
```
X_test_2d.shape
```

Out[ ]:
```
(1980, 300)
```

Model Building and Training

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.naive_bayes import MultinomialNB


scaler = MinMaxScaler()
scaled_train_embed = scaler.fit_transform(X_train_2d)
scaled_test_embed = scaler.transform(X_test_2d)

model = MultinomialNB()
model.fit(scaled_train_embed, y_train)
```

Out[ ]:
```
▾ MultinomialNB
MultinomialNB()
```



$$X_i' = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$

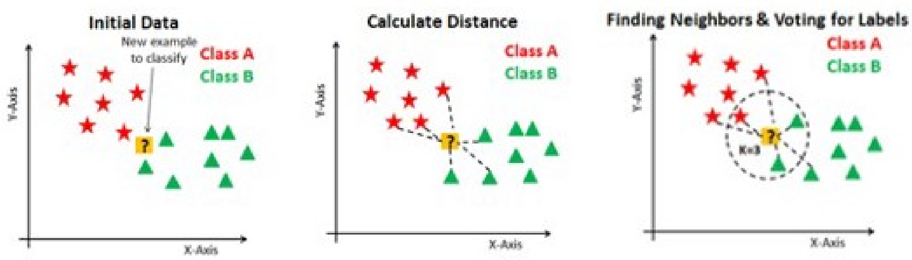
In [ ]:
```
model.score(scaled_test_embed, y_test)
```

Out[ ]: 0.9439393939393394

In [ ]:
```
y_preds = model.predict(scaled_test_embed)

print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      1024
           1       0.94      0.95      0.94       956

    accuracy                           0.94      1980
   macro avg       0.94      0.94      0.94      1980
weighted avg       0.94      0.94      0.94      1980
```

KNN K nearest neighbors Classification



In [ ]:
```
from  sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors = 5, metric = 'euclidean')
```

```
model.fit(X_train_2d, y_train)
```

Out[ ]:

```
KNeighborsClassifier
KNeighborsClassifier(metric='euclidean')
```

In [ ]:

```
model.score(X_test_2d, y_test)
```

Out[ ]: 0.990909090909091

In [ ]:

```
y_preds = model.predict(X_test_2d)

print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      1024
           1       0.99      0.99      0.99       956

    accuracy                           0.99      1980
   macro avg       0.99      0.99      0.99      1980
weighted avg       0.99      0.99      0.99      1980
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_preds)
cm
```

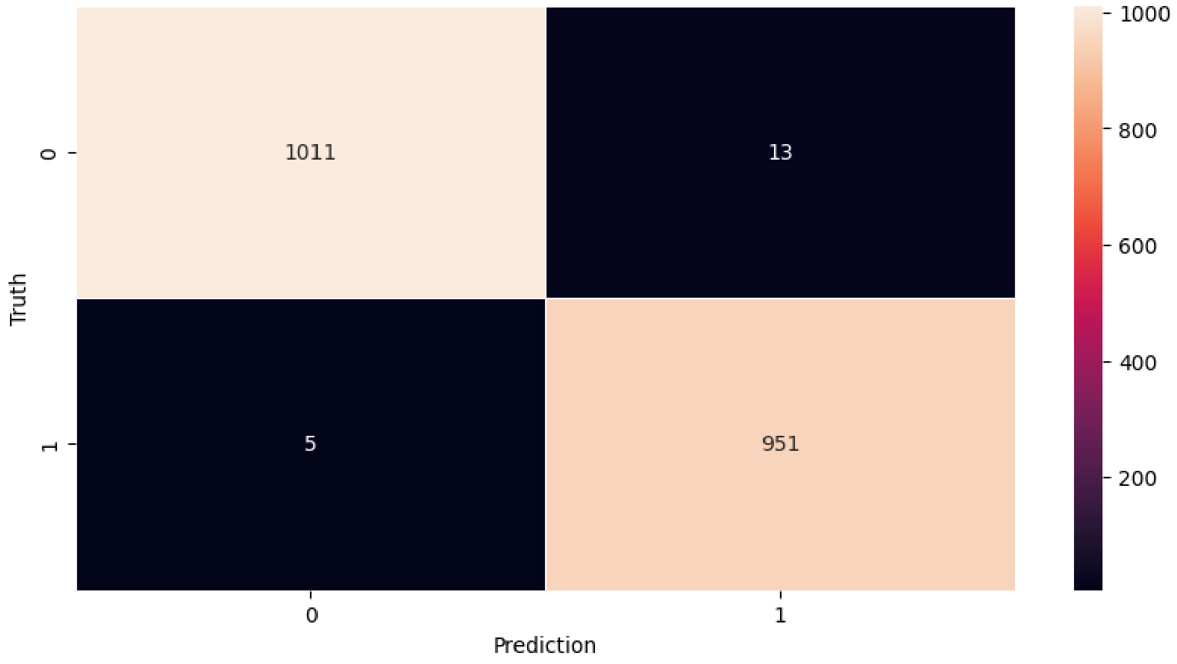
Out[ ]:

```
array([[1011,   13],
       [   5,  951]], dtype=int64)
```

In [ ]:

```
plt.figure(figsize=(10,5))

sn.heatmap(
    cm,
    annot=True,
    fmt='d',
    linewidth=0.5
)

plt.xlabel('Prediction')
plt.ylabel('Truth')
```

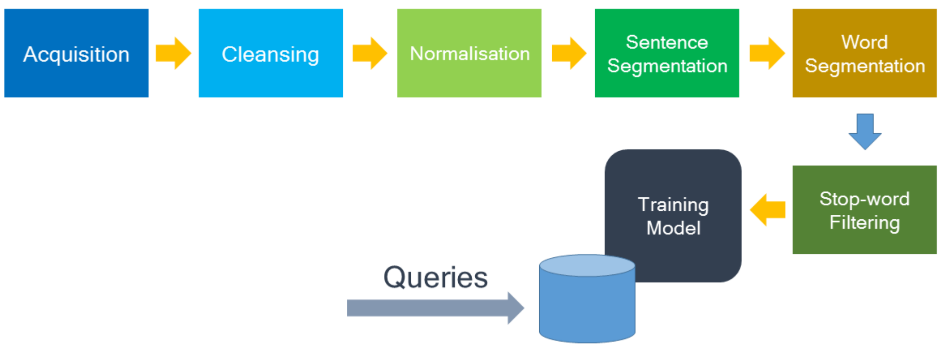
Out[ ]: Text(95.72222222222221, 0.5, 'Truth')

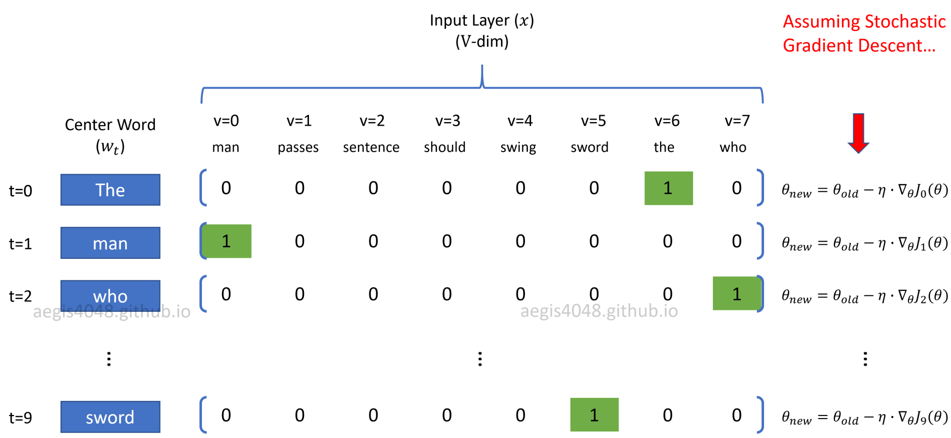


Word Vectors Overview Using Gensim Library

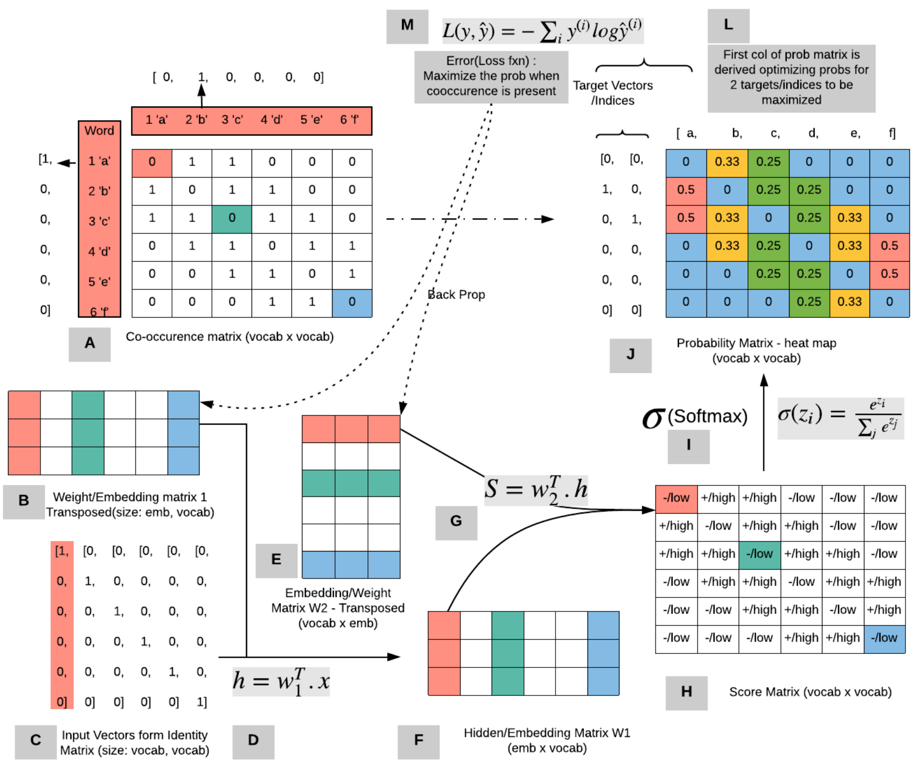
gensim models

!pip install gensim





Word2Vec - Information Flow




```
In [ ]: import gensim.downloader as api
```

```
In [ ]: wv = api.load('word2vec-google-news-300')
```

```
In [ ]: wv.similarity(w1="great", w2="good")
```

```
Out[ ]: 0.729151
```

```
In [ ]: wv.most_similar("good")
```

```
Out[ ]: [('great', 0.7291510105133057),
         ('bad', 0.7190051078796387),
         ('terrific', 0.6889115571975708),
         ('decent', 0.6837348341941833),
         ('nice', 0.6836092472076416),
         ('excellent', 0.644292950630188),
         ('fantastic', 0.6407778263092041),
         ('better', 0.6120728850364685),
         ('solid', 0.5806034803390503),
         ('lousy', 0.576420247554779)]
```

```
In [ ]: wv.most_similar(positive=['king', 'woman'], negative=['man'])
```

Out[ ]:
```
[('queen', 0.7118193507194519),
 ('monarch', 0.6189674139022827),
 ('princess', 0.5902431011199951),
 ('crown_prince', 0.5499460697174072),
 ('prince', 0.5377321839332581),
 ('kings', 0.5236844420433044),
 ('Queen_Consort', 0.5235945582389832),
 ('queens', 0.5181134343147278),
 ('sultan', 0.5098593831062317),
 ('monarchy', 0.5087411999702454)]
```

In [ ]:
```
wv.most_similar(positive=['king', 'woman'], negative=['man'], topn=5)
```

Out[ ]:
```
[('queen', 0.7118193507194519),
 ('monarch', 0.6189674139022827),
 ('princess', 0.5902431011199951),
 ('crown_prince', 0.5499460697174072),
 ('prince', 0.5377321839332581)]
```

In [ ]:
```
wv.most_similar(positive=['France', 'Berlin'], negative=['Paris'], topn=5)
```

Out[ ]:
```
[('Germany', 0.7901254892349243),
 ('Austria', 0.6026812195777893),
 ('German', 0.6004959940910339),
 ('Germans', 0.5851002931594849),
 ('Poland', 0.5847075581550598)]
```

In [ ]:
```
wv.doesnt_match(["facebook", "cat", "google", "microsoft"])
```

Out[ ]:
```
'cat'
```

In [ ]:
```
wv.doesnt_match(["dog", "cat", "google", "mouse"])
```

Out[ ]:
```
'google'
```

> ## Gensim: Glove
>
> GloVe

In [ ]:
```
glv = api.load("glove-twitter-25")
```

In [ ]:
```
glv.most_similar("good")
```

Out[ ]:
```
[('too', 0.9648017287254333),
 ('day', 0.9533665180206299),
 ('well', 0.9503170847892761),
 ('nice', 0.9438973665237427),
 ('better', 0.9425962567329407),
 ('fun', 0.9418926239013672),
 ('much', 0.9413353800773621),
 ('this', 0.9387555122375488),
 ('hope', 0.9383506774902344),
 ('great', 0.9378516674041748)]
```

In [ ]:
```
glv.doesnt_match("breakfast cereal dinner lunch".split())
```

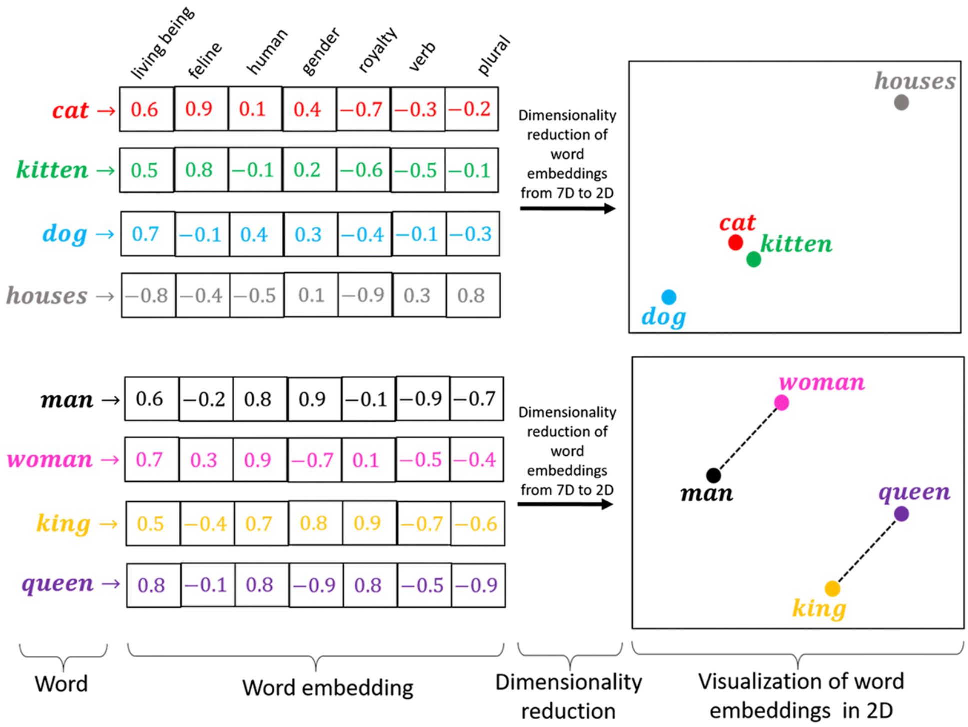
Out[ ]:
```
'cereal'
```

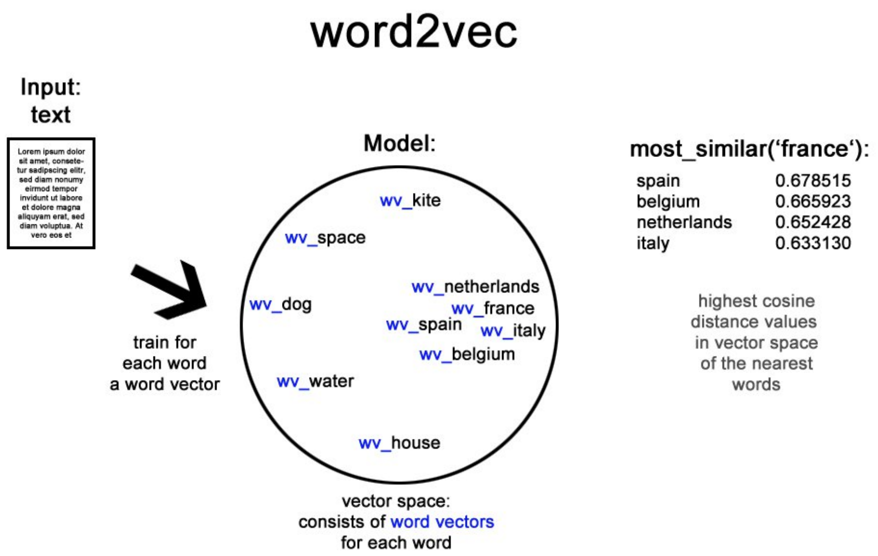
In [ ]:
```
glv.doesnt_match("facebook cat google microsoft".split())
```

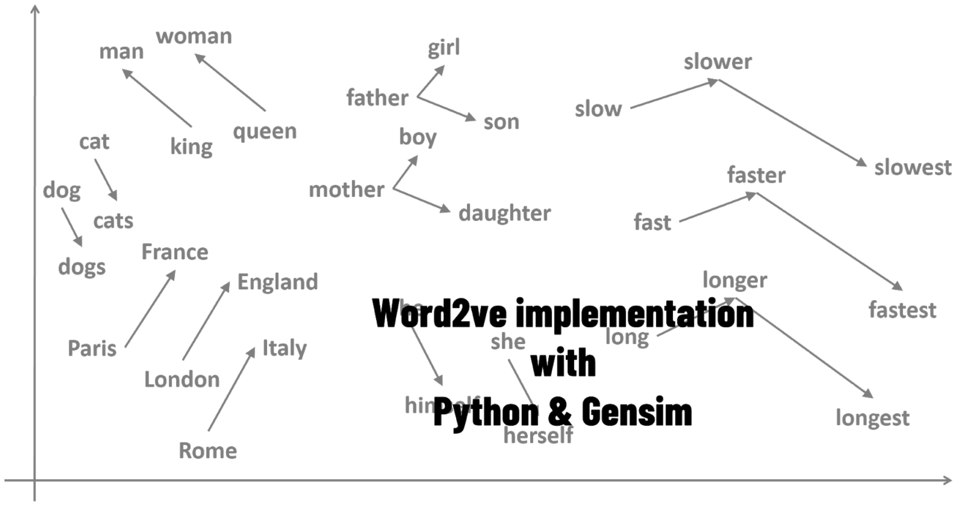
Out[ ]: 'cat'

Text Classification USing Gensim Word Embeddings

Dataset







Using gensim's word2vec embeddings to convert text into vector

1. Preprocess the text to remove stop words, punctuations and get lemma for each word
2. Get word vectors for each of the words in a pre-processed sentece
3. Take a mean of all word vectors to derive the numeric representation of the entire news article

> import gensim.downloader as api
> wv = api.load('word2vec-google-news-300')

In [ ]:
```
wv.similarity(w1="great", w2="good")
```
Out[ ]: 0.729151

In [ ]:
```
wv_great = wv["great"]
wv_good = wv["good"]
```

In [ ]:
```
wv_great.shape
```
Out[ ]: (300,)

In [ ]:
```
type(wv_great)
```
Out[ ]: numpy.ndarray

Get Data

In [ ]:
```
df = pd.read_csv("data/fake_and_real_news.csv")
df.head()
```

Out[ ]:

	Text	label
0	Top Trump Surrogate BRUTALLY Stabs Him In The...	Fake
1	U.S. conservative leader optimistic of common ...	Real
2	Trump proposes U.S. tax overhaul, stirs concer...	Real
3	Court Forces Ohio To Allow Millions Of Illega...	Fake
4	Democrats say Trump agrees to work on immigrat...	Real

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9900 entries, 0 to 9899
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Text    9900 non-null   object
 1   label   9900 non-null   object
dtypes: object(2)
memory usage: 154.8+ KB
```

In [ ]:
```
df['label'].value_counts()
```

Out[ ]:
```
Fake    5000
Real    4900
Name: label, dtype: int64
```

In [ ]:
```
df['label_num'] = df['label'].map({
    "Fake": 0,
    "Real": 1
})
df.head()
```

Out[ ]:

	Text	label	label_num
0	Top Trump Surrogate BRUTALLY Stabs Him In The...	Fake	0
1	U.S. conservative leader optimistic of common ...	Real	1
2	Trump proposes U.S. tax overhaul, stirs concer...	Real	1
3	Court Forces Ohio To Allow Millions Of Illega...	Fake	0
4	Democrats say Trump agrees to work on immigrat...	Real	1

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9900 entries, 0 to 9899
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Text       9900 non-null   object
 1   label      9900 non-null   object
 2   label_num  9900 non-null   int64 
dtypes: int64(1), object(2)
memory usage: 232.2+ KB
```

> ## convert the text into a vector

In [ ]:
```
import spacy
nlp = spacy.load("en_core_web_lg")

def preprocess_and_vectorize(text):
    doc = nlp(text)

    filtered_tokens = []

    for token in doc:
        if token.is_punct or token.is_stop:
            continue
        filtered_tokens.append(token.lemma_)

    return filtered_tokens
```

In [ ]:
```
preprocess_and_vectorize("Don't worry if you don't understand")
```

Out[ ]:
```
['worry', 'understand']
```

In [ ]:
```
wv.get_mean_vector(['worry', 'understand'], pre_normalize=False)[:5]
```

Out[ ]:
```
array([ 0.00976562, -0.00561523, -0.08905029,  0.01330566, -0.2709961 ],
      dtype=float32)
```

In [ ]:
```
v1 = wv['worry']
v2 = wv['understand']
np.mean([v1, v2], axis=0)[:5]
```

Out[ ]:
```
array([ 0.00976562, -0.00561523, -0.08905029,  0.01330566, -0.2709961 ],
      dtype=float32)
```

In [ ]:
```
import spacy
nlp = spacy.load("en_core_web_lg")

def preprocess_and_vectorize(text):
    doc = nlp(text)

    filtered_tokens = []

    for token in doc:
        if token.is_punct or token.is_stop:
            continue
        filtered_tokens.append(token.lemma_)

    return wv.get_mean_vector(filtered_tokens)
```

In [ ]:
```
v = preprocess_and_vectorize("Don't worry if you don't understand")
v.shape
```

Out[ ]:
```
(300,)
```

In [ ]:
```
df['vector'] = df['Text'].apply(preprocess_and_vectorize)
#df['vector_text'] = df['Text'].apply(lambda text: preprocess_and_vectorize(text))
```

In [ ]:
```
df.head()
```

Out[ ]:

	Text	label	label_num	vector
0	Top Trump Surrogate BRUTALLY Stabs Him In The...	Fake	0	[0.008145372, 0.019952843, -0.00989356, 0.0344...
1	U.S. conservative leader optimistic of common ...	Real	1	[0.00861828, 0.007408227, 0.0007675802, 0.0138...
2	Trump proposes U.S. tax overhaul, stirs concer...	Real	1	[0.01823076, 0.0063306373, -0.0058634086, 0.03...
3	Court Forces Ohio To Allow Millions Of Illega...	Fake	0	[0.012453172, 0.0122098895, 6.3027373e-06, 0.0...
4	Democrats say Trump agrees to work on immigrat...	Real	1	[-0.0022669104, 0.011340516, 0.003596399, 0.02...

Train-Test splitting

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    df['vector'].values,
    df['label_num'],
    test_size=0.2,
    random_state=2022,
    stratify=df['label_num']
)
```

Reshaping the X_train and X_test to fit model

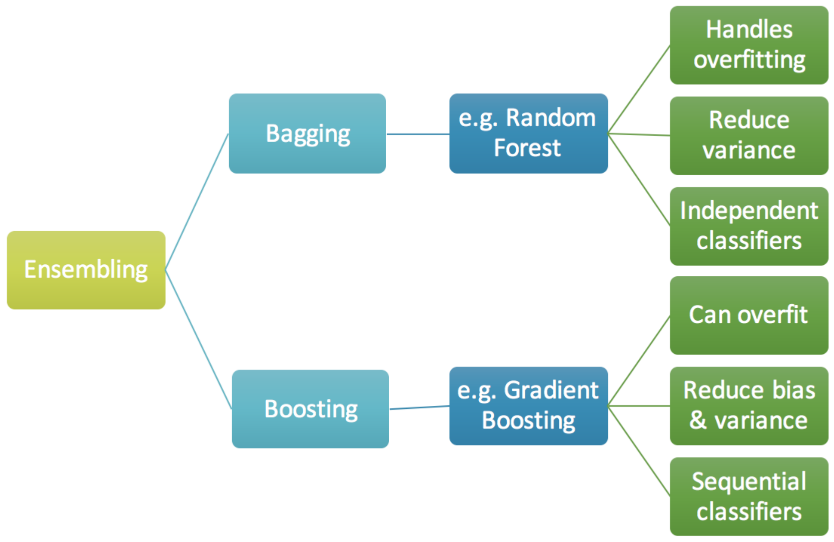
In [ ]:
```
X_train[:3]
```

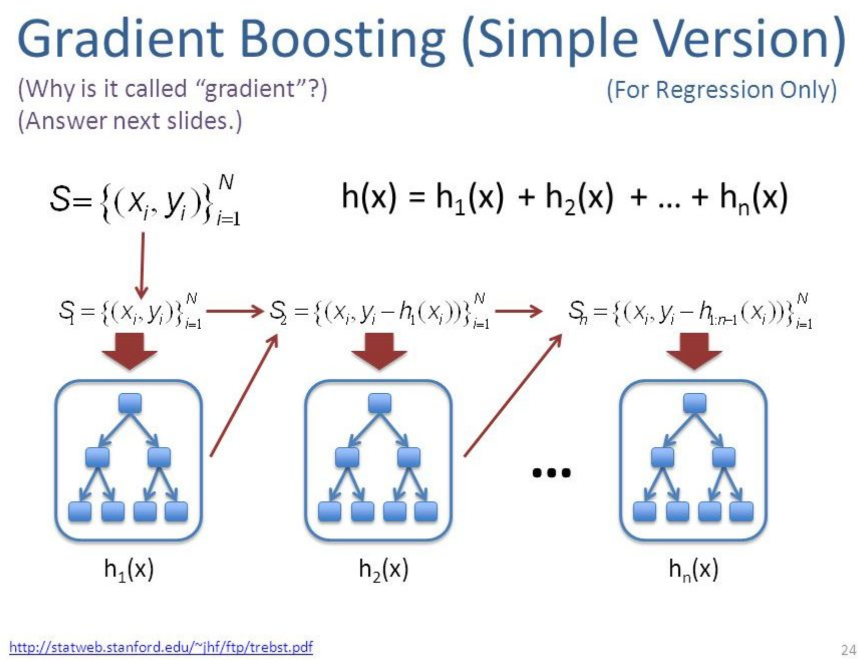
In [ ]:
```
X_train_2d = np.stack(X_train)
X_test_2d =  np.stack(X_test)
```

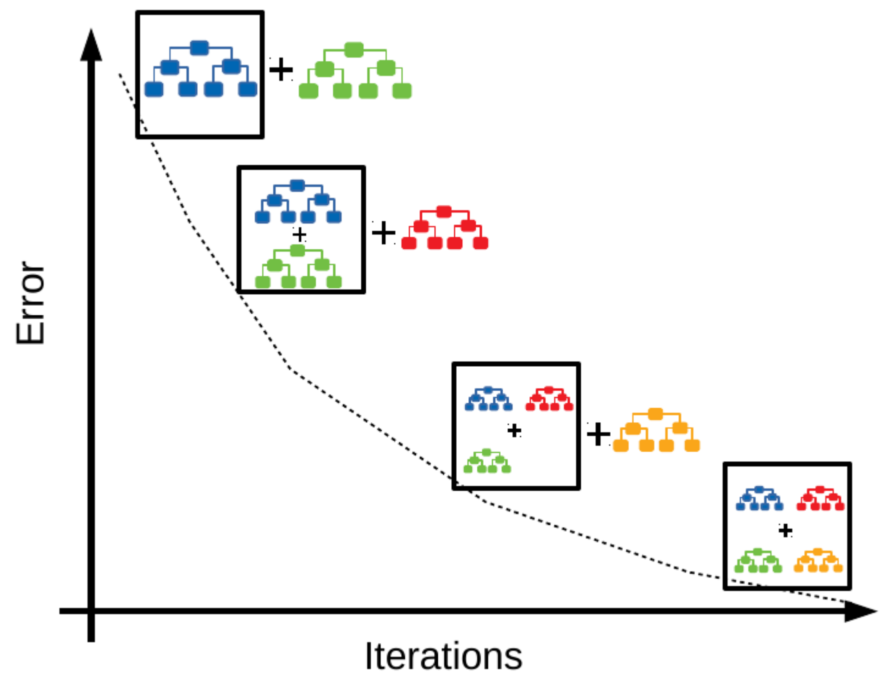
In [ ]:
```
X_train_2d[:3]
```

Model Building and Training

Gradient Boosting Classifier


Handles overfitting
Bagging
e.g. Random Forest
Reduce variance
Independent classifiers
Ensembling
Can overfit
Boosting
e.g. Gradient Boosting
Reduce bias & variance
Sequential classifiers


Gradient Boosting (Simple Version)
(Why is it called "gradient"?)
(Answer next slides.)
(For Regression Only)
$S=\{(x_i, y_i)\}_{i=1}^{N}$
h(x) = h_1(x) + h_2(x) + ... + h_n(x)
$S_1=\{(x_i, y_i)\}_{i=1}^{N}$
$S_2=\{(x_i, y_i-h_1(x_i))\}_{i=1}^{N}$
$S_n=\{(x_i, y_i-h_{1:n-1}(x_i))\}_{i=1}^{N}$
...
h_1(x)
h_2(x)
h_n(x)
http://statweb.stanford.edu/~jhf/ftp/trebst.pdf
24




```
In [ ]: from sklearn.ensemble import GradientBoostingClassifier

        model = GradientBoostingClassifier()

        model.fit(X_train_2d, y_train)
```

```
Out[ ]: ▾ GradientBoostingClassifier
        GradientBoostingClassifier()
```

```
In [ ]: model.score(X_test_2d, y_test)
```

```
Out[ ]: 0.9782828282828283
```

```
In [ ]: y_preds = model.predict(X_test_2d)
```

```
In [ ]: from sklearn.metrics import classification_report
        print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      1000
           1       0.97      0.98      0.98       980

    accuracy                           0.98      1980
   macro avg       0.98      0.98      0.98      1980
weighted avg       0.98      0.98      0.98      1980
```

Test

```
In [ ]: test_news = [
            "Michigan governor denies misleading U.S. House on Flint water (Reuters) - Mich
            " WATCH: Fox News Host Loses Her Sh*t, Says Investigating Russia For Hacking Ou
            " Sarah Palin Celebrates After White Man Who Pulled Gun On Black Protesters Goe
        ]
```

In [ ]:
```
test_news_vectors = [preprocess_and_vectorize(n) for n in test_news]
```

In [ ]:
```
model.predict(test_news_vectors)
```

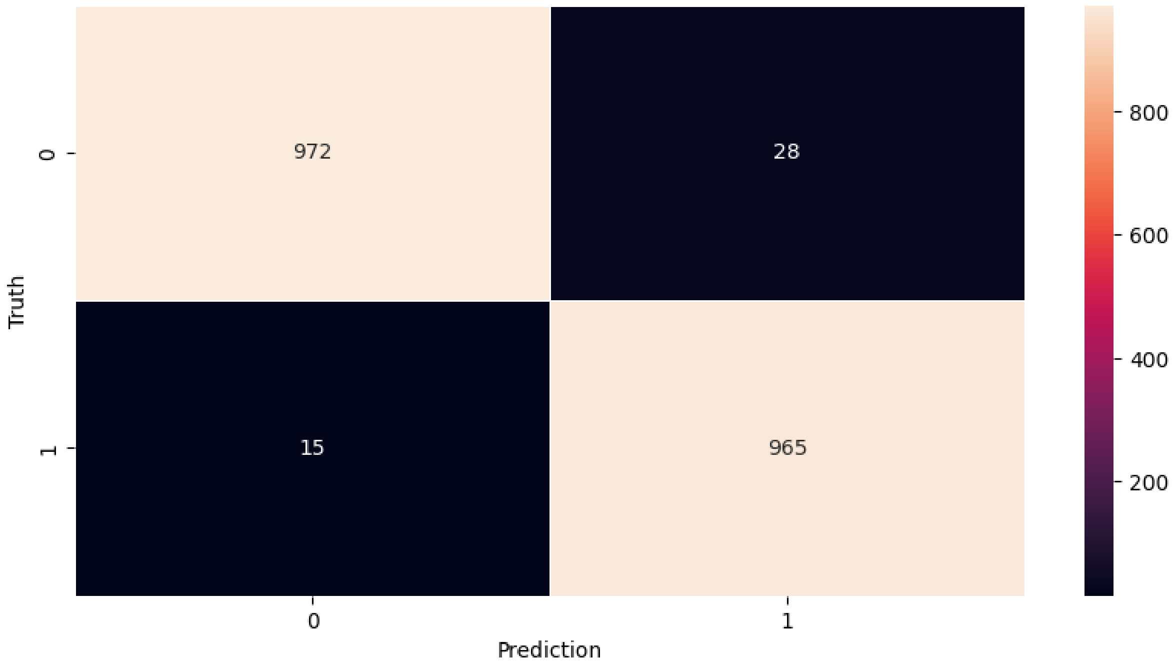
Out[ ]:
```
array([1, 0, 0], dtype=int64)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_preds)
cm
```

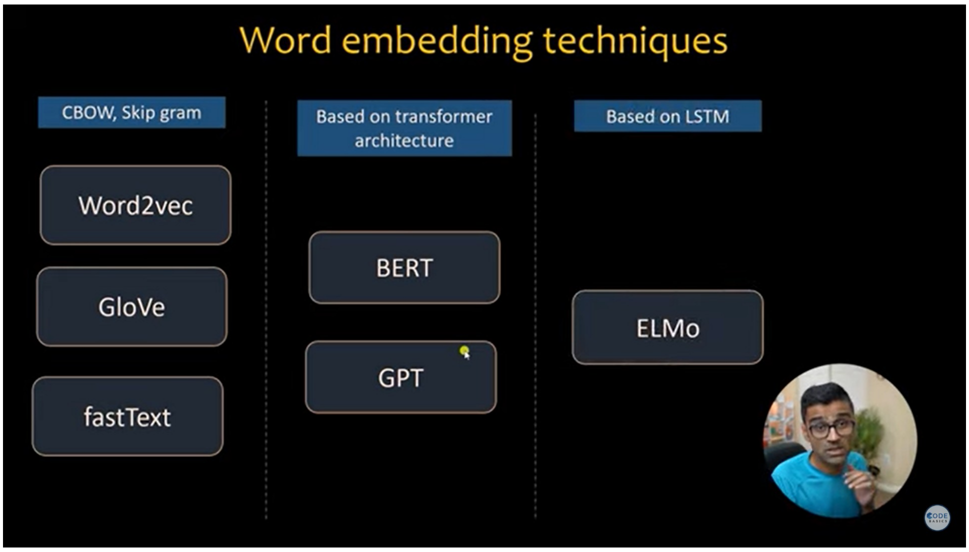
Out[ ]:
```
array([[972,  28],
       [ 15, 965]], dtype=int64)
```

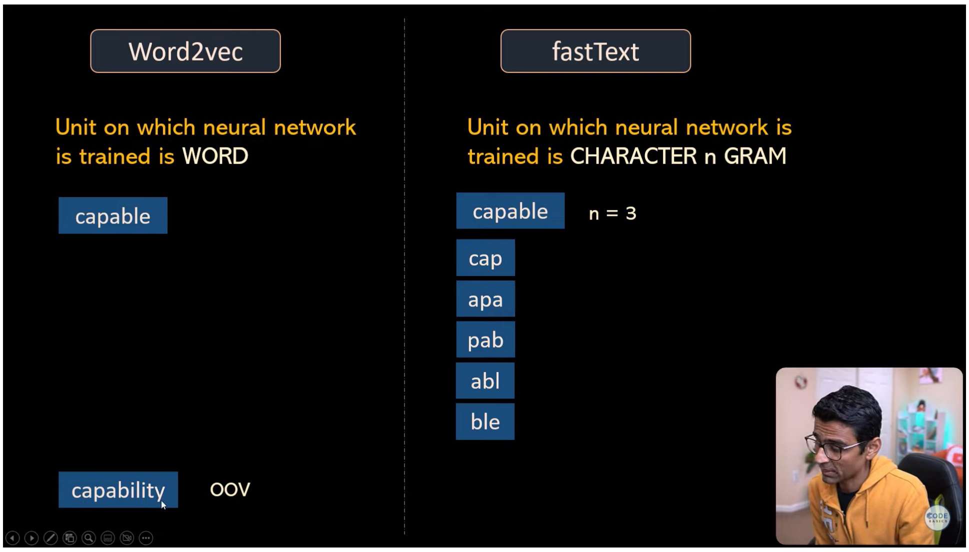
In [ ]:
```
plt.figure(figsize=(10,5))

sn.heatmap(
    cm,
    annot=True,
    fmt='d',
    linewidth=0.5
)

plt.xlabel('Prediction')
plt.ylabel('Truth')
```

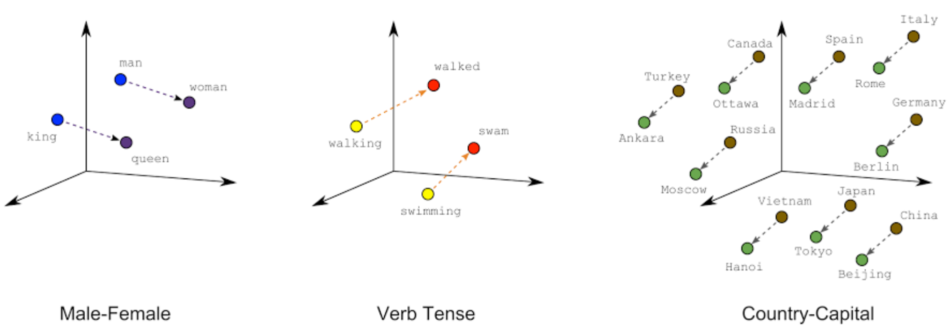
Out[ ]:
```
Text(95.72222222222221, 0.5, 'Truth')
```



> ## Train Custom Word Vectors in [fastText](#)
>
> git clone https://github.com/facebookresearch/fastText.git
>
> cd fastText
>
> pip install .
>
> pip install fasttext pip install fastText







In [ ]:
```
df = pd.read_csv("data/Cleaned_Indian_Food_Dataset.csv")
df.head(3)
```

Out[ ]:

	TranslatedRecipeName	TranslatedIngredients	TotalTimeInMins	Cuisine	TranslatedInstructions
0	Masala Karela Recipe	1 tablespoon Red Chilli powder,3 tablespoon Gr...	45	Indian	To begin making the Masala Karela Recipe,de-se...
1	Spicy Tomato Rice (Recipe)	2 teaspoon cashew - or peanuts, 1/2 Teaspoon ...	15	South Indian Recipes	To make tomato puliogere, first cut the tomato...
2	Ragi Semiya Upma Recipe - Ragi Millet Vermicel...	1 Onion - sliced,1 teaspoon White Urad Dal (Sp...	50	South Indian Recipes	To begin making the Ragi Vermicelli Recipe, fi...

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5938 entries, 0 to 5937
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   TranslatedRecipeName    5938 non-null   object
 1   TranslatedIngredients   5938 non-null   object
 2   TotalTimeInMins         5938 non-null   int64 
 3   Cuisine                 5938 non-null   object
 4   TranslatedInstructions  5938 non-null   object
 5   URL                     5938 non-null   object
 6   Cleaned-Ingredients     5938 non-null   object
 7   image-url               5938 non-null   object
 8   Ingredient-count        5938 non-null   int64 
dtypes: int64(2), object(7)
memory usage: 417.6+ KB
```

In [ ]:

```
df['TranslatedInstructions'][0]
```

Out[ ]:

```
'To begin making the Masala Karela Recipe,de-seed the karela and slice.\r\nDo not
remove the skin as the skin has all the nutrients.\r\nAdd the karela to the pressu
re cooker with 3 tablespoon of water, salt and turmeric powder and pressure cook f
or three whistles.\r\nRelease the pressure immediately and open the lids.\r\nKeep
aside.Heat oil in a heavy bottomed pan or a kadhai.\r\nAdd cumin seeds and let it
sizzle.Once the cumin seeds have sizzled, add onions and saute them till it turns
golden brown in color.Add the karela, red chilli powder, amchur powder, coriander
powder and besan.\r\nStir to combine the masalas into the karela.Drizzle a little
extra oil on the top and mix again.\r\nCover the pan and simmer Masala Karela stir
ring occasionally until everything comes together well.\r\nTurn off the heat.Trans
fer Masala Karela into a serving bowl and serve.Serve Masala Karela along with Pan
chmel Dal and Phulka for a weekday meal with your family.\r\n'
```

> # Pre-procesing - Use Regular Expression to Clean Data

> 1.Remove punctuation
>
> 2.Remove extra space
>
> 3.Make the entire sentence lower case

In [ ]:
```
import re

def preprocess(text):
    text = re.sub(r'[^\w\s]', ' ', text)
    text = re.sub(r'[ \r\n]+', ' ', text)
    return text.strip().lower()
```

In [ ]:
```
text = "To begin making the Masala Karela Recipe,de-seed the karela and slice.\r\nD
```

In [ ]:
```
preprocess(text)
```

Out[ ]:
```
'to begin making the masala karela recipe de seed the karela and slice do not remo
ve the skin as the skin has all the nutrients add the karela to the pressure cooke
r with 3 tablespoon of water salt and turmeric powder and pressure cook for three
whistles release the pressure immediately and open the lids keep aside heat oil in
a heavy bottomed pan or a kadhai add cumin seeds and let it sizzle once the cumin
seeds have sizzled add onions and saute them till it turns golden brown in color a
dd the karela red chilli powder amchur powder coriander powder and besan stir to c
ombine the masalas into the karela drizzle a little extra oil on the top and mix a
gain cover the pan and simmer masala karela stirring occasionally until everything
comes together well turn off the heat transfer masala karela into a serving bowl a
nd serve serve masala karela along with panchmel dal and phulka for a weekday meal
with your family'
```

In [ ]:
```
df['TranslatedInstructions'] = df['TranslatedInstructions'].apply(preprocess)
```

In [ ]:
```
df.head()
```

Out[ ]:

	TranslatedRecipeName	TranslatedIngredients	TotalTimeInMins	Cuisine	TranslatedInstructions
0	Masala Karela Recipe	1 tablespoon Red Chilli powder,3 tablespoon Gr...	45	Indian	to begin making the masala karela recipe de se...
1	Spicy Tomato Rice (Recipe)	2 teaspoon cashew - or peanuts, 1/2 Teaspoon ...	15	South Indian Recipes	to make tomato puliogere first cut the tomatoe...
2	Ragi Semiya Upma Recipe - Ragi Millet Vermicel...	1 Onion - sliced,1 teaspoon White Urad Dal (Sp...	50	South Indian Recipes	to begin making the ragi vermicelli recipe fir...
3	Gongura Chicken Curry Recipe - Andhra Style Go...	1/2 teaspoon Turmeric powder (Haldi),1 tablesp...	45	Andhra	to begin making gongura chicken curry recipe f...
4	Andhra Style Alam Pachadi Recipe - Adrak Chutn...	oil - as per use, 1 tablespoon coriander seed...	30	Andhra	to make andhra style alam pachadi first heat o...

In [ ]:
```
df['TranslatedInstructions'][0]
```

Out[ ]:
```
'to begin making the masala karela recipe de seed the karela and slice do not remove the skin as the skin has all the nutrients add the karela to the pressure cooker with 3 tablespoon of water salt and turmeric powder and pressure cook for three whistles release the pressure immediately and open the lids keep aside heat oil in a heavy bottomed pan or a kadhai add cumin seeds and let it sizzle once the cumin seeds have sizzled add onions and saute them till it turns golden brown in color add the karela red chilli powder amchur powder coriander powder and besan stir to combine the masalas into the karela drizzle a little extra oil on the top and mix again cover the pan and simmer masala karela stirring occasionally until everything comes together well turn off the heat transfer masala karela into a serving bowl and serve serve masala karela along with panchmel dal and phulka for a weekday meal with your family'
```

In [ ]:
```python
df.to_csv("data/food_receipes.csv", columns=['TranslatedInstructions'], header=None
```

In [ ]:
```python
import fasttext

model = fasttext.train_unsupervised("data/food_receipes.csv")
```

In [ ]:
```python
model.get_nearest_neighbors("paneer")
```

Out[ ]:
```
[(0.6278824806213379, 'bhurji'),
 (0.6183227896690369, 'tikka'),
 (0.6128366589546204, 'makhanwala'),
 (0.5976285934448242, 'tikkas'),
 (0.5888250470161438, 'makhane'),
 (0.5729339718818665, 'tandoori'),
 (0.568075954914093, 'reshmi'),
 (0.5664334297180176, 'makhani'),
 (0.5645010471343994, 'burji'),
 (0.5590397715568542, 'satay')]
```

In [ ]:
```python
model.get_word_vector("dosa").shape
```

Out[ ]:
```
(100,)
```

Text Classification Using fastText

In [ ]:
```python
df= pd.read_csv("data/ecommerce_dataset.csv", header=None)
df.head()
```

Out[ ]:

	0	1
0	Household	Paper Plane Design Framed Wall Hanging Motivat...
1	Household	SAF 'Floral' Framed Painting (Wood, 30 inch x ...
2	Household	SAF 'UV Textured Modern Art Print Framed' Pain...
3	Household	SAF Flower Print Framed Painting (Synthetic, 1...
4	Household	Incredible Gifts India Wooden Happy Birthday U...

In [ ]:
```python
df.columns=["category", "description"]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50425 entries, 0 to 50424
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   category     50425 non-null  object
 1   description  50424 non-null  object
dtypes: object(2)
memory usage: 788.0+ KB
```

In [ ]:
```
df['category'].value_counts()
```

Out[ ]:
```
Household                 19313
Books                     11820
Electronics               10621
Clothing & Accessories     8671
Name: category, dtype: int64
```

In [ ]:
```
df.dropna(inplace=True)
df.shape
```

Out[ ]:
```
(50424, 2)
```

In [ ]:
```
df['category'].replace("Clothing & Accessories", "Clothing_Accessories", inplace=Tı
df['category'].unique()
```

Out[ ]:
```
array(['Household', 'Books', 'Clothing_Accessories', 'Electronics'],
      dtype=object)
```

In [ ]:
```
df['category'] = "__label__" + df['category'].astype(str)
df.head()
```

Out[ ]:

	category	description
0	__label__Household	Paper Plane Design Framed Wall Hanging Motivat...
1	__label__Household	SAF 'Floral' Framed Painting (Wood, 30 inch x ...
2	__label__Household	SAF 'UV Textured Modern Art Print Framed' Pain...
3	__label__Household	SAF Flower Print Framed Painting (Synthetic, 1...
4	__label__Household	Incredible Gifts India Wooden Happy Birthday U...

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 50424 entries, 0 to 50424
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   category     50424 non-null  object
 1   description  50424 non-null  object
dtypes: object(2)
memory usage: 1.2+ MB
```

In [ ]:
```
df['category_description'] = df['category'] + ' ' + df['description']
df.head()
```

Out[ ]:

	category	description	category_description
0	__label__Household	Paper Plane Design Framed Wall Hanging Motivat...	__label__Household Paper Plane Design Framed W...
1	__label__Household	SAF 'Floral' Framed Painting (Wood, 30 inch x ...	__label__Household SAF 'Floral' Framed Paintin...
2	__label__Household	SAF 'UV Textured Modern Art Print Framed' Pain...	__label__Household SAF 'UV Textured Modern Art...
3	__label__Household	SAF Flower Print Framed Painting (Synthetic, 1...	__label__Household SAF Flower Print Framed Pai...
4	__label__Household	Incredible Gifts India Wooden Happy Birthday U...	__label__Household Incredible Gifts India Wood...

In [ ]:
```
df['category_description'][0]
```

Out[ ]:
```
'__label__Household Paper Plane Design Framed Wall Hanging Motivational Office Dec
or Art Prints (8.7 X 8.7 inch) - Set of 4 Painting made up in synthetic frame with
uv textured print which gives multi effects and attracts towards it. This is an sp
ecial series of paintings which makes your wall very beautiful and gives a royal t
ouch. This painting is ready to hang, you would be proud to possess this unique pa
inting that is a niche apart. We use only the most modern and efficient printing t
echnology on our prints, with only the and inks and precision epson, roland and hp
printers. This innovative hd printing technique results in durable and spectacular
looking prints of the highest that last a lifetime. We print solely with top-notch
100% inks, to achieve brilliant and true colours. Due to their high level of uv re
sistance, our prints retain their beautiful colours for many years. Add colour and
style to your living space with this digitally printed painting. Some are for plea
sure and some for eternal bliss.so bring home this elegant print that is lushed wi
th rich colors that makes it nothing but sheer elegance to be to your friends and
family.it would be treasured forever by whoever your lucky recipient is. Liven up
your place with these intriguing paintings that are high definition hd graphic dig
ital prints for home, office or any room.'
```

> ## Pre-procesing - Use Regular Expression to Clean Data

> 1.Remove punctuation
>
> 2.Remove extra space
>
> 3.Make the entire sentence lower case

In [ ]:
```
df['category_description'] = df['category_description'].apply(preprocess)
```

In [ ]:
```
df['category_description'][0]
```

Out[ ]:
```
'__label__household paper plane design framed wall hanging motivational office dec
or art prints 8 7 x 8 7 inch set of 4 painting made up in synthetic frame with uv
textured print which gives multi effects and attracts towards it this is an specia
l series of paintings which makes your wall very beautiful and gives a royal touch
this painting is ready to hang you would be proud to possess this unique painting
that is a niche apart we use only the most modern and efficient printing technolog
y on our prints with only the and inks and precision epson roland and hp printers
this innovative hd printing technique results in durable and spectacular looking p
rints of the highest that last a lifetime we print solely with top notch 100 inks
to achieve brilliant and true colours due to their high level of uv resistance our
prints retain their beautiful colours for many years add colour and style to your
living space with this digitally printed painting some are for pleasure and some f
or eternal bliss so bring home this elegant print that is lushed with rich colors
that makes it nothing but sheer elegance to be to your friends and family it would
be treasured forever by whoever your lucky recipient is liven up your place with t
hese intriguing paintings that are high definition hd graphic digital prints for h
ome office or any room'
```

Train Test Split

In [ ]:
```python
from sklearn.model_selection import train_test_split

train, test = train_test_split(df, test_size=0.2)
```

In [ ]:
```python
train.shape, test.shape
```

Out[ ]:
```
((40339, 3), (10085, 3))
```

In [ ]:
```python
train.to_csv("data/ecommerce_train.csv", columns=["category_description"], index=Fa
test.to_csv("data/ecommerce_test.csv", columns=["category_description"], index=Fals
```

In [ ]:
```python
import fasttext

model = fasttext.train_supervised(input="data/ecommerce_train.csv")
```

In [ ]:
```python
model.test("data/ecommerce_test.csv")
```

Out[ ]:
```
(10085, 0.9699553792761527, 0.9699553792761527)
```

> - First parameter (10085) is test size. Second and third parameters are precision and recall respectively.

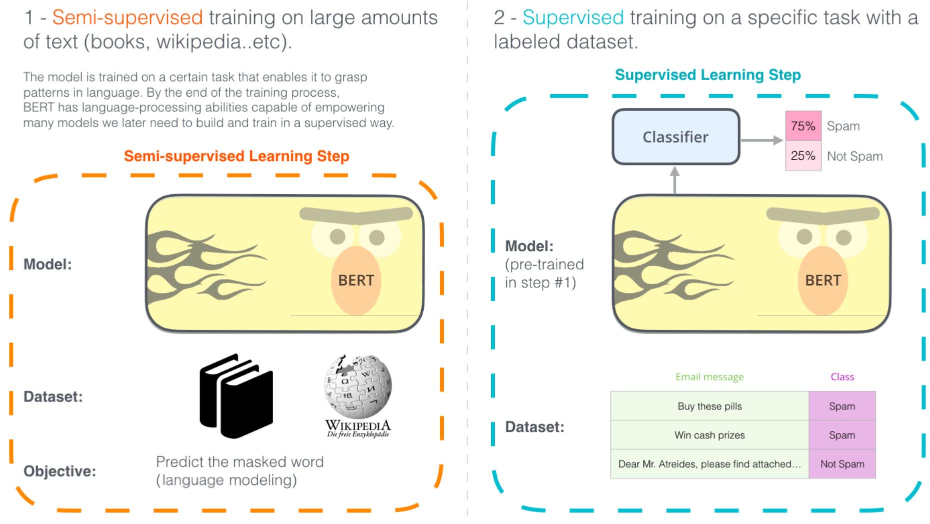
In [ ]:
```python
model.predict("wintech assemble desktop pc cpu 500 gb sata hdd 4 gb ram intel c2d p
```

Out[ ]:
```
(('__label__electronics',), array([0.99772298]))
```

In [ ]:
```python
model.predict("ockey men's cotton t shirt fabric details 80 cotton 20 polyester sup
```

Out[ ]:
```
(('__label__clothing_accessories',), array([1.00000703]))
```

BERT (Bidirectional Encoder Representations From Transformers)



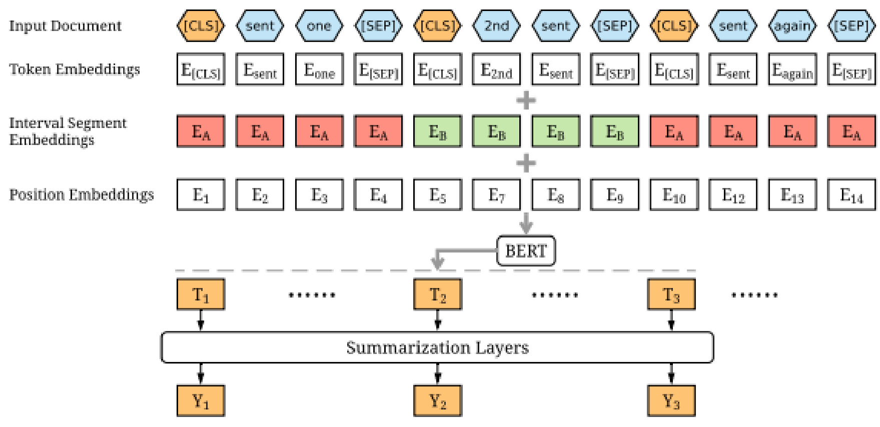
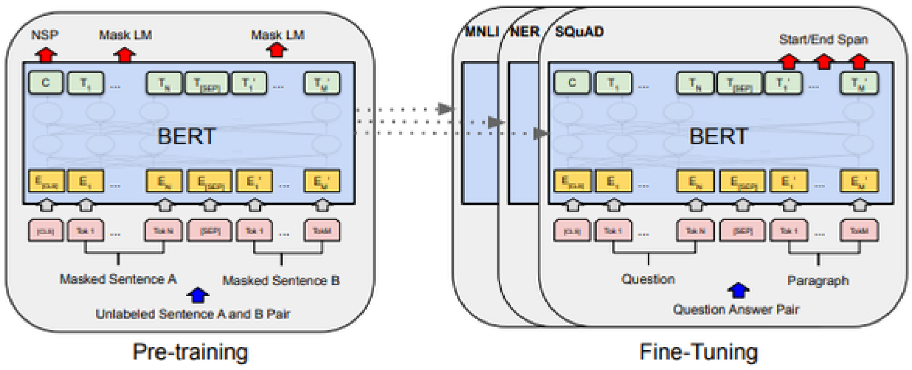


Figure 1: The overview architecture of the BERTSUM model.



Regular Expression Tutorial

regular expressions 101

Tesla Company Filings

In [ ]:
```
import re
```

In [ ]:
```
text = '''
Elon musk's phone number is 9991116666, call him if you have any questions on dodge
Tesla's CFO number (999)-333-7777
'''
```

In [ ]:
```
pattern = '\(\d{3}\)\-\d{3}\-\d{4}|\d{10}'
matches = re.findall(pattern, text)
matches
```

Out[ ]: ['9991116666', '(999)-333-7777']

In [ ]:
```
text = '''
Note 1 - Overview
Tesla, Inc. (“Tesla”, the “Company”, “we”, “us” or “our”) was incorporated in the S
products. Our Chief Executive Officer, as the chief operating decision maker (“CODM
Beginning in the first quarter of 2021, there has been a trend in many parts of the
against COVID-19, as well as an easing of restrictions on social, business, travel
rates and regulations continue to fluctuate in various regions and there are ongoin
and increases in costs for logistics and supply chains, such as increased port cong
supply. We have also previously been affected by temporary manufacturing closures,
administrative activities supporting our product deliveries and deployments.
Note 2 - Summary of Significant Accounting Policies
Unaudited Interim Financial Statements
The consolidated balance sheet as of September 30, 2021, the consolidated statement
comprehensive income, the consolidated statements of redeemable noncontrolling inte
30, 2021 and 2020 and the consolidated statements of cash flows for the nine months
disclosed in the accompanying notes, are unaudited. The consolidated balance sheet
consolidated financial statements as of that date. The interim consolidated financi
conjunction with the annual consolidated financial statements and the accompanying
ended December 31, 2020.
'''
```

In [ ]:
```
pattern = 'Note \d - ([^\n]*)'
matches = re.findall(pattern, text)
matches
```

Out[ ]: ['Overview', 'Summary of Significant Accounting Policies']

In [ ]:
```
text = '''
The gross cost of operating lease vehicles in FY2021 Q1 was $4.85 billion.
In previous quarter i.e. FY2020 Q4 it was $3 billion.
'''
```

In [ ]:
```
pattern = 'FY\d{4} Q[1-4]'
matches = re.findall(pattern, text)
matches
```

Out[ ]: ['FY2021 Q1', 'FY2020 Q4']

In [ ]:
```
text = '''
The gross cost of operating lease vehicles in FY2021 Q1 was $4.85 billion.
In previous quarter i.e. fy2020 Q4 it was $3 billion.
'''
```

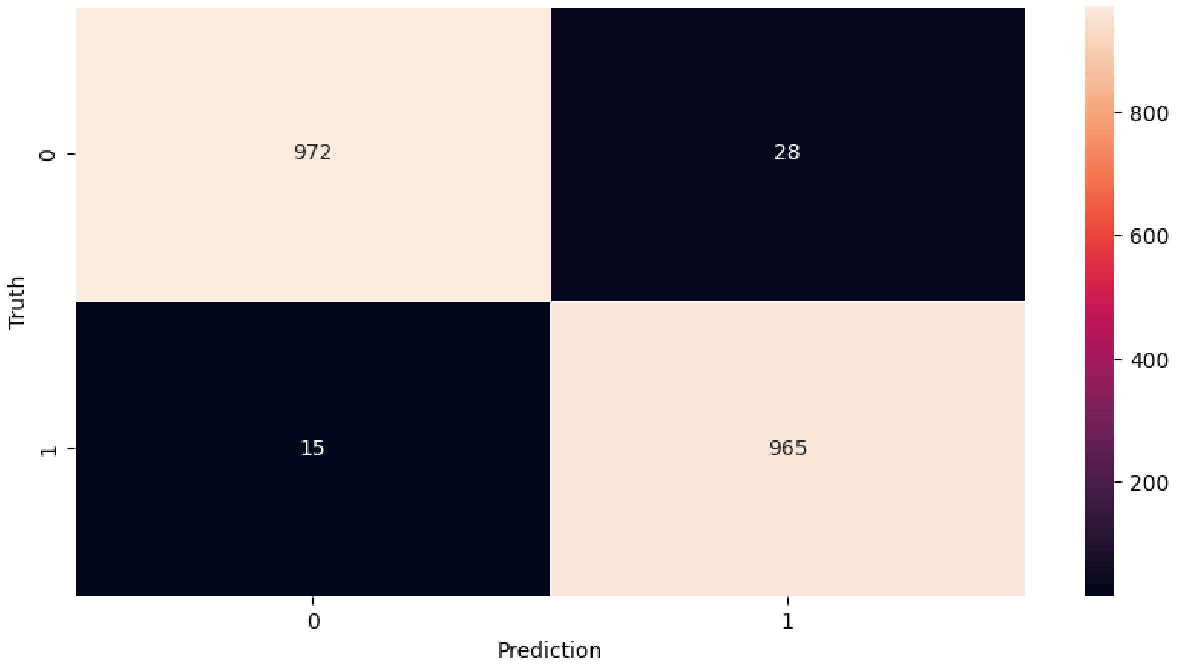
In [ ]:
```
pattern = 'FY\d{4} Q[1-4]'
matches = re.findall(pattern, text)
matches
```

Out[ ]: ['FY2021 Q1']

In [ ]:
```
plt.figure(figsize=(10,5))

sn.heatmap(
```

```
    cm,
    annot=True,
    fmt='d',
    linewidth=0.5
)

plt.xlabel('Prediction')
plt.ylabel('Truth')
```

Out[ ]: Text(95.72222222222221, 0.5, 'Truth')



Case insensitive pattern match using `flags=re.IGNORECASE`

In [ ]:
```
pattern = 'FY\d{4} Q[1-4]'
matches = re.findall(pattern, text, flags=re.IGNORECASE)
matches
```

Out[ ]: ['FY2021 Q1', 'fy2020 Q4']

In [ ]:
```
pattern = 'FY(\d{4} Q[1-4])'
matches = re.findall(pattern, text, flags=re.IGNORECASE)
matches
```

Out[ ]: ['2021 Q1', '2020 Q4']

In [ ]:
```
pattern = '\$([\d\.]+)'
matches = re.findall(pattern, text, flags=re.IGNORECASE)
matches
```

Out[ ]: ['4.85', '3']

In [ ]:
```
pattern = 'FY(\d{4} Q[1-4])|\$([\d\.]+)'
matches = re.findall(pattern, text, flags=re.IGNORECASE)
matches
```

Out[ ]: [('2021 Q1', ''), ('', '4.85'), ('2020 Q4', ''), ('', '3')]

In [ ]:
```
pattern = 'FY(\d{4} Q[1-4])[^\$]+\$([\d\.]+)'
matches = re.findall(pattern, text, flags=re.IGNORECASE)
matches
```

Out[ ]:
```
[('2021 Q1', '4.85'), ('2020 Q4', '3')]
```

In [ ]:
```
def get_pattern_match(pattern, text):
    matches = re.findall(pattern, text, flags=re.IGNORECASE)
    if matches:
        return matches
```

In [ ]:
```
chat1='codebasics: Hello, I am having an issue with my order # 412889912'
chat2='codebasics: I have a problem with my order number 412889912'
chat3='codebasics: My order 412889912 is having an issue, I was charged 300$ when c
```

In [ ]:
```
pattern = 'order[^\d]*(\d*)'
get_pattern_match(pattern, chat1)
```

Out[ ]:
```
['412889912']
```

In [ ]:
```
chat1 = 'codebasics: you ask lot of questions 😠  1235678912, abc@xyz.com'
chat2 = 'codebasics: here it is: (123)-567-8912, abcX@xyz.com'
chat3 = 'codebasics: yes, phone: 1235678912 email: abc_82@xyz.com'
```

In [ ]:
```
pattern = '(\d{10})|(\(\d{3}\)-\d{3}-\d{4})|([a-zA-Z0-9_]*@[a-z]*\.[a-zA-Z0-9]*)'
```

In [ ]:
```
get_pattern_match(pattern, chat1)
```

Out[ ]:
```
[('1235678912', '', ''), ('', '', 'abc@xyz.com')]
```

In [ ]:
```
get_pattern_match(pattern, chat2)
```

Out[ ]:
```
[('', '(123)-567-8912', ''), ('', '', 'abcX@xyz.com')]
```

In [ ]:
```
get_pattern_match(pattern, chat3)
```

Out[ ]:
```
[('1235678912', '', ''), ('', '', 'abc_82@xyz.com')]
```

In [ ]:
```
text = '''
Born    Mukesh Dhirubhai Ambani
19 April 1957 (age 64)
Aden, Colony of Aden
(present-day Yemen)[1][2]
Nationality     Indian
Alma mater
St. Xavier's College, Mumbai
Institute of Chemical Technology (B.E.)
Stanford University (drop-out)
Occupation      Chairman and MD, Reliance Industries
Spouse(s)       Nita Ambani (m. 1985)[3]
Children        3
Parent(s)
Dhirubhai Ambani (father)
Kokilaben Ambani (mother)
Relatives       Anil Ambani (brother)
Tina Ambani (sister-in-law)
'''
```

In [ ]:
```
def get_pattern_match(pattern, text):
    matches = re.findall(pattern, text, flags=re.IGNORECASE)
    if matches:
        return matches[0]

def extract_personal_information(text):
    age = get_pattern_match('age (\d+)', text)
    full_name = get_pattern_match('Born(.*)\n', text)
    birth_date = get_pattern_match('Born.*\n(.*)\(age', text)
    birth_place = get_pattern_match('\(age.*\n(.*)', text)
    return {
        'age': int(age),
        'name': full_name.strip(),
        'birth_date': birth_date.strip(),
        'birth_place': birth_place.strip(),
    }
```

In [ ]:
```
extract_personal_information(text)
```

Out[ ]:
```
{'age': 64,
 'name': 'Mukesh Dhirubhai Ambani',
 'birth_date': '19 April 1957',
 'birth_place': 'Aden, Colony of Aden'}
```

Future Plan

> - Integrate with Industry Domain Knowledge

-- Memo End --