

## 101 NLP Exercises (using modern libraries)

*by Shrivarsheni /*

*Natural language processing is the technique by which AI understands human language. NLP tasks such as text classification, summarization, sentiment analysis, translation are widely used. This post aims to serve as a reference for basic and advanced NLP tasks.*

---

### Upcoming Posts

---

[spaCy Tutorial - Complete writeup \(NEW\)](#)

[101 NLP Exercises \(using modern libraries\) \(NEW\)](#)

[How to train spaCy to autodetect new entities \(NER\) \(NEW\)](#)

[Support Vector Machines Algorithm from Scratch](#)

[Creating Plots in Julia](#)

[Julia DataFrames \(NEW\)](#)

[101 Julia Practice Exercises](#)

[Python SQLite - Must Read Guide](#)

[Linear Regression with Julia \(NEW\)](#)

[Waterfall Plot in Python \(NEW\)](#)

[Python JSON - Guide](#)



101 NLP Exercises using modern libraries. Photo by Ana Justin Luebke.

## 1. Import nltk and download the 'stopwords' and 'punkt' packages

[Logistic Regression in Julia](#)

[Probability Theory - Beginners Guide](#)

[Graph Theory](#)

[Gentle Introduction to Markov Chain \(NEW\)](#)

[Logistic Regression from Scratch](#)

[Python Collections - Guide and Examples \(NEW\)](#)

[report this ad](#)

**Recent Posts**

Difficulty Level : L1

Q. Import `nltk` and necessary packages

▼ Show Solution

```
# Downloading packages and importing

import nltk
nltk.download('punkt')
nltk.download('stop')
nltk.download('stopwords')

#> [nltk_data] Downloading package punkt to /root/nltk_data...
#> [nltk_data]   Unzipping tokenizers/punkt.zip.
#> [nltk_data] Error loading stop: Package 'stop' not found in in
#> [nltk_data] Downloading package stopwords to /root/nltk_data..
#> [nltk_data]   Unzipping corpora/stopwords.zip.
#> True
```

Difficulty Level : L1

Q. Import `spacy` library and load 'en\_core\_web\_sm' model for

[cProfile – How to profile your python code](#)

[Subplots Python \(Matplotlib\)](#)

[101 NLP Exercises \(using modern libraries\)](#)

[How to Train spaCy to Autodetect New Entities \(NER\) \[Complete Guide\]](#)

[For-Loop in Julia](#)

[DataFrames in Julia](#)

[Matplotlib Line Plot](#)

[K-Means Clustering Algorithm from Scratch](#)

[While-loop in Julia](#)

[Function in Julia](#)

[Python Scatter Plot](#)

[Julia – Programming Language](#)

[Requests in Python \(Guide\)](#)

[Matplotlib Pyplot](#)

[Python Boxplot](#)

[Bar Plot in Python](#)

[data.table in R – The Complete Beginners Guide](#)

## 2. Import spacy and load the language model

english language. Load 'xx\_ent\_wiki\_sm' for multi language support.

✓ Show Solution

```
# Import and load model

import spacy
nlp=spacy.load("en_core_web_sm")
nlp
# More models here: https://spacy.io/models
```

```
#> <spacy.lang.en.English at 0x7facaf6cd0f0>
```

## 3. How to tokenize a given text?

Difficulty Level : L1

[Augmented Dickey Fuller Test \(ADF Test\) – Must Read Guide](#)

[KPSS Test for Stationarity](#)

[101 R data.table Exercises](#)

### Top Posts & Pages

---

[ARIMA Model - Complete Guide to Time Series Forecasting in Python](#)

[Time Series Analysis in Python - A Comprehensive Guide with Examples](#)

[Parallel Processing in Python - A Practical Guide with Examples](#)

[Cosine Similarity - Understanding the math and how it works \(with python codes\)](#)

[Machine Learning Better Explained!](#)

[Topic Modeling with Gensim \(Python\)](#)

[Top 50 matplotlib Visualizations - The Master Plots \(with full python code\)](#)

[Matplotlib Histogram - How to Visualize Distributions in Python](#)

Q. Print the tokens of the given text document

Input :

```
text="Last week, the University of Cambridge shared its own research"
```

Desired Output :



```
Last  
week  
,  
the  
University
```

[101 Pandas Exercises for Data Analysis](#)

[Python Logging - Simplest Guide with Full Code and Examples](#)

## Tags

[Classification](#) [data.table](#) [Data](#)

[Manipulation](#) [Debugging](#) [Doc2Vec](#) [Evaluation](#)

[Metrics](#) [FastText](#) [Feature Selection](#) [Gensim](#)

[HuggingFace](#) [Julia](#) [Julia Packages](#) [LDA](#)

[Lemmatization](#) [Linear Regression](#) [Logistic Loop](#)

[LSI](#) [Machine Learning](#) [Matplotlib](#)

[NLP](#) [NLTK](#) [Numpy](#) [P-Value](#) [Pandas](#) [Phraser](#)

[plots](#) [Practice](#) [Exercise](#) [Python](#) [R](#) [Regex](#)

[Regression](#) [Residual Analysis](#) [Scikit Learn](#)

[Significance Tests](#) [Soft Cosine Similarity](#)

[spaCy](#) [Stationarity](#) [TextBlob](#) [TextSummarization](#)

[TFIDF](#) [Time Series](#) [Topic Modeling](#)

[Visualization](#) [Word2Vec](#)

```
of  
Cambridge  
shared  
...(truncated)...
```

✓ Show Solution

```
# Tokeniation with nltk  
tokens=nltk.word_tokenize(text)  
for token in tokens:  
    print(token)
```

```
# Tokenization with spaCy  
nlp=spacy.load("en_core_web_sm")  
doc=nlp(text)  
for token in doc:  
    print(token.text)
```

## 4. How to get the sentences of a text document ?

Difficulty Level : L1

Q. Print the sentences of the given text document

Input :

```
text="""The outbreak of coronavirus disease 2019 (COVID-19) has cre
```

Desired Output :

```
The outbreak of coronavirus disease 2019 (COVID-19) has created a g
Not only the rate of contagion and patterns of transmission threate
Within this context of physical threat, social and physical distanc
...(truncated)...
```

▼ Show Solution

```
# Tokenizing the text into sentences with spaCy
doc=nlp(text)
for sentence in doc.sents:
    print(sentence)
    print(' ')
```

```
#> The outbreak of coronavirus disease 2019 (COVID-19) has create  
#> Not only the rate of contagion and patterns of transmission th  
#> Within this context of physical threat, social and physical di  
#> the role of the different mass media channels in our lives on  
#> Mass media have long been recognized as powerful forces shapin  
#> This recognition is accompanied by a growing volume of researc  
#> Are media (broadcast and digital) still able to convey a sense
```

```
# Extracting sentences with nltk  
nltk.sent_tokenize(text)
```

```
#> ['The outbreak of coronavirus disease 2019 (COVID-19) has crea  
#> 'Not only the rate of contagion and patterns of transmission  
#> 'Within this context of physical threat, social and physical d  
#> 'Mass media have long been recognized as powerful forces shapi  
#> 'This recognition is accompanied by a growing volume of resear  
#> 'radio, movies, television, the internet, mobiles) and the zei
```



```
#> 'cold war, 9/11, climate change) in an attempt to map mass me  
#> 'Are media (broadcast and digital) still able to convey a sen
```

## 5. How to tokenize a text using the `transformers` package ?

Difficulty Level : L1

Q. Tokenize the given text in encoded form using the `tokenizer` of Huggingface's `transformer` package.

Input :

```
text="I love spring season. I go hiking with my friends"
```

Desired Output :

```
[101, 1045, 2293, 3500, 2161, 1012, 1045, 2175, 13039, 2007, 2026,
```

```
[CLS] i love spring season. i go hiking with my friends [SEP]
```

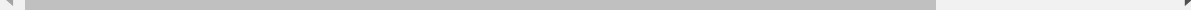
▼ Show Solution

```
# Import tokenizer from transformers
!pip install transformers
from transformers import AutoTokenizer

# Initialize the tokenizer
tokenizer=AutoTokenizer.from_pretrained('bert-base-uncased')

# Encoding with the tokenizer
inputs=tokenizer.encode(text)
print(inputs)
tokenizer.decode(inputs)
```

```
#> [101, 1045, 2293, 3500, 2161, 1012, 1045, 2175, 13039, 2007, 2]
#> [CLS] i love spring season. i go hiking with my friends [SEP]
```



Difficulty Level : L2

## 6. How to tokenize text with stopwords as delimiters?

Q. Tokenize the given text with stop words ("is","the","was") as delimiters. Tokenizing this way identifies meaningful phrases. Sometimes, useful for [topic modeling](#)

Input :

```
text = "Walter was feeling anxious. He was diagnosed today. He prob
```

Expected Output :

```
['Walter',  
 'feeling anxious',  
 'He',  
 'diagnosed today',  
 'He probably',  
 'best person I know']
```

▼ Show Solution

```
# Solution
text = "Walter was feeling anxious. He was diagnosed today. He pr

stop_words_and_delims = ['was', 'is', 'the', '.', ',', '-', '!',
for r in stop_words_and_delims:
    text = text.replace(r, 'DELIM')

words = [t.strip() for t in text.split('DELIM')]
words_filtered = list(filter(lambda a: a not in ['', ], words))
words_filtered

#> ['Walter',
#>  'feeling anxious',
#>  'He',
#>  'diagnosed today',
#>  'He probably',
#>  'best person I know']
```

## 7. How to remove stop words in a text ?

Difficulty Level : L1

Q. Remove all the stopwords ( 'a' , 'the', 'was'...) from the text

Input :

```
text=""the outbreak of coronavirus disease 2019 (COVID-19) has cre
```

Desired Output :

```
'outbreak coronavirus disease 2019 ( COVID-19 ) created global hea
```

✓ Show Solution

```
# Method 1
# Removing stopwords in nltk

from nltk.corpus import stopwords
```

```
my_stopwords=set(stopwords.words('english'))
new_tokens=[]

# Tokenization using word_tokenize()
all_tokens=nltk.word_tokenize(text)

for token in all_tokens:
    if token not in my_stopwords:
        new_tokens.append(token)

" ".join(new_tokens)
```

```
#> 'outbreak coronavirus disease 2019 ( COVID-19 ) created global
```

```
# Method 2
# Removing stopwords in spaCy

doc=nlp(text)
new_tokens=[]

# Using is_stop attribute of each token to check if it's a stopwo
for token in doc:
    if token.is_stop==False:
```

```
new_tokens.append(token.text)
```

```
" ".join(new_tokens)
```

```
#> 'outbreak coronavirus disease 2019 ( COVID-19 ) created global'
```

## 8. How to add custom stop words in spaCy ?

Difficulty Level : L1

Q. Add the custom stopwords "NIL" and "JUNK" in spaCy and remove the stopwords in below text

Input :

```
text=" Jonas was a JUNK great guy NIL Adam was evil NIL Martha JUNI
```

Expected Output :

```
'Jonas great guy Adam evil Martha fool'
```

▼ Show Solution

```
# list of custom stop words
customize_stop_words = ['NIL', 'JUNK']

# Adding these stop words
for w in customize_stop_words:
    nlp.vocab[w].is_stop = True
doc = nlp(text)
tokens = [token.text for token in doc if not token.is_stop]

" ".join(tokens)
```

```
#> ' Jonas great guy Adam evil Martha fool'
```

## 9. How to remove punctuations ?



Difficulty Level : L1

Q. Remove all the punctuations in the given text

Input :

```
text="The match has concluded !!! India has won the match . Will we
```

Desired Output :

```
'The match has concluded India has won the match Will we fin the f.
```

▼ Show Solution

```
# Removing punctuations in spaCy

doc=nlp(text)
new_tokens=[]
# Check if a token is a punctuation through is_punct attribute
for token in doc:
    if token.is_punct==False:
        new_tokens.append(token.text)
```

```
" ".join(new_tokens)
```

```
#> 'The match has concluded India has won the match Will we fin t
```

```
# Method 2  
# Removing punctuation in nltk with RegexpTokenizer  
  
tokenizer=nltk.RegexpTokenizer(r"\w+")  
  
tokens=tokenizer.tokenize(text)  
" ".join(tokens)
```

```
#> 'The match has concluded India has won the match Will we fin t
```

## 10. How to perform stemming

Difficulty Level : L2

Q. Perform stemming/ convert each token to it's root form in the given text

Input :

```
text= "Dancing is an art. Students should be taught dance as a sub
```

Desired Output:

```
text= 'danc is an art . student should be taught danc as a subject
```

▼ Show Solution

```
# Stemming with nltk's PorterStemmer

from nltk.stem import PorterStemmer
stemmer=PorterStemmer()
stemmed_tokens=[]
for token in nltk.word_tokenize(text):
    stemmed_tokens.append(stemmer.stem(token))

" ".join(stemmed_tokens)
```

```
#> 'danc is an art . student should be taught danc as a subject j
```

## 11. How to lemmatize a given text ?

Difficulty Level : L2

Q. Perform lemmatization on the given text

Hint: [Lemmatization Approaches](#)

Input:

```
text= "Dancing is an art. Students should be taught dance as a subj
```

Desired Output:

```
text= 'dancing be an art . student should be teach dance as a subj
```

✓ Show Solution

```
# Lemmatization using spacy's lemma_ attribute of token
nlp=spacy.load("en_core_web_sm")
doc=nlp(text)

lemmatized=[token.lemma_ for token in doc]
" ".join(lemmatized)
```

```
#> 'dancing be an art . student should be teach dance as a subject'
```

## 12. How to extract usernames from emails ?

Difficulty Level : L2

Q. Extract the usernames from the email addresses present in the text

Input :

```
text= "The new registrations are potter709@gmail.com , elixir101@gr
```

Desired Output :

```
['potter709', 'elixir101', 'granger111', 'severus77']
```

▼ Show Solution

```
# Using regular expression to extract usernames
import re

# \S matches any non-whitespace character
# @ for as in the Email
# + for Repeats a character one or more times
usernames= re.findall('(\S+)@', text)
print(usernames)
```

```
#> ['potter709', 'elixir101', 'granger111', 'severus77']
```

Difficulty Level : L2

## 13. How to find the most common words in the text excluding stopwords

Q. Extract the top 10 most common words in the given text excluding stopwords.

Input :

```
text="""Junkfood - Food that do no good to our body. And there's no
that they are ready to eat or easy to cook foods. People, of all ages
Junkfood is the most dangerous food ever but it is pleasure in eat
Junkfood is very harmful that is slowly eating away the health of t
The problem is more serious than you think. Various studies show th
Junkfood is the easiest way to gain unhealthy weight. The amount of
This food only looks and tastes good, other than that, it has no po
```

Desired Output:

```
text= {Junkfood: 10,
      food: 8,
```

```
good: 5,  
harmful : 3  
body: 1,  
need: 1,  
  
...(truncated)
```

▼ Show Solution

```
# Creating spacy doc of the text  
nlp=spacy.load("en_core_web_sm")  
doc=nlp(text)  
  
# Removal of stop words and punctuations  
words=[token for token in doc if token.is_stop==False and token.i  
  
freq_dict={}  
  
# Calculating frequency count  
for word in words:  
    if word not in freq_dict:  
        freq_dict[word]=1  
    else:  
        freq_dict[word]+=1  
  
freq_dict
```



```
{Junkfood: 10,  
  food: 8,  
  good: 5,  
  harmful : 3  
  body: 1,  
  need: 1,  
  
  ...(truncated)
```

## 14. How to do spell correction in a given text ?

Difficulty Level : L2

Q. Correct the spelling errors in the following text

Input :

```
text="He is a gret person. He beleives in bod"
```

Desired Output:

```
text="He is a great person. He believes in god"
```

▼ Show Solution

```
# Import textblob
from textblob import TextBlob

# Using textblob's correct() function
text=TextBlob(text)
print(text.correct())
#> He is a great person. He believes in god
```

## 15. How to tokenize tweets ?

Difficulty Level : L2

Q. Clean the following tweet and tokenize them

Input :

```
text=" Having lots of fun #goa #vaction #summervacation. Fancy dini
```

Desired Output :

```
['Having',  
'lots',  
'of',  
'fun',  
'goa',  
'vaction',  
'summervacation',  
'Fancy',  
'dinner',  
'Beachbay',  
'restro']  
````
```

✓ Show Solution

```
import re  
# Cleaning the tweets  
text=re.sub(r'^\w+', ' ', text)  
  
# Using nltk's TweetTokenizer
```

```
from nltk.tokenize import TweetTokenizer
tokenizer=TweetTokenizer()
tokenizer.tokenize(text)
```

```
#> ['Having',
#>  'lots',
#>  'of',
#>  'fun',
#>  'goa',
#>  'vaction',
#>  'summervacation',
#>  'Fancy',
#>  'dinner',
#>  'Beachbay',
#>  'restro']
```

## 16. How to extract all the nouns in a text?

Difficulty Level : L2

Q. Extract and print all the nouns present in the below text

Input:

```
text="James works at Microsoft. She lives in manchester and likes
```

Desired Output :

```
James
Microsoft
manchester
flute
```

▼ Show Solution

```
# Coverting the text into a spacy Doc
nlp=spacy.load("en_core_web_sm")
doc=nlp(text)

# Using spacy's pos_ attribute to check for part of speech tags
for token in doc:
    if token.pos_=='NOUN' or token.pos_=='PROPN':
```

```
print(token.text)
```

```
#> James
```

```
#> Microsoft
```

```
#> manchester
```

```
#> flute
```

## 17. How to extract all the pronouns in a text?

Difficulty Level : L2

Q. Extract and print all the pronouns in the text

Input :

Desired Output :

```
He
```

He  
She

```
text="John is happy finally. He had landed his dream job finally. I
```

▼ Show Solution

```
# Using spacy's pos_ attribute to check for part of speech tags
nlp=spacy.load("en_core_web_sm")
doc=nlp(text)

for token in doc:
    if token.pos_=='PRON':
        print(token.text)

#> He
#> He
#> She
```

Find the similarity between any two words.

## 18. How to find similarity between two words?

Input :

```
word1="amazing"  
word2="terrible"  
word3="excellent"
```

Desired Output:

```
#> similarity between amazing and terrible is 0.46189071343764604  
#> similarity between amazing and excellent is 0.6388207086737778
```

▼ Show Solution

```
# Convert words into spacy tokens  
import spacy  
!python -m spacy download en_core_web_lg  
nlp=spacy.load('en_core_web_lg')  
token1=nlp(word1)  
token2=nlp(word2)
```



```
token3=nlp(word3)

# Use similarity() function of tokens
print('similarity between', word1,'and' ,word2, 'is' ,token1.simi
print('similarity between', word1,'and' ,word3, 'is' ,token1.simi

#> similarity between amazing and terrible is 0.46189071343764604
#> similarity between amazing and excellent is 0.6388207086737778
```

## 19. How to find similarity between two documents?

Difficulty Level : L2

Q. Find the similarity between any two text documents

Input :

```
text1="John lives in Canada"
text2="James lives in America, though he's not from there"
```

Desired Output :

```
0.792817083631068
```

▼ Show Solution

```
# Finding similarity using spaCy library

doc1=nlp(text1)
doc2=nlp(text2)
doc1.similarity(doc2)

#> 0.792817083631068
```

## 20. How to find the cosine similarity of two documents?

Difficulty Level : L3

Q. Find the cosine similarity between two given documents

Input

```
text1='Taj Mahal is a tourist place in India'  
text2='Great Wall of China is a tourist place in china'
```

Desired Output :

```
[[1.          0.45584231]  
 [0.45584231 1.          ]]
```

✓ Show Solution

```
# Using Vectorizer of sklearn to get vector representation  
documents=[text1,text2]  
from sklearn.feature_extraction.text import CountVectorizer  
import pandas as pd  
  
vectorizer=CountVectorizer()  
matrix=vectorizer.fit_transform(documents)  
  
# Obtaining the document-word matrix
```

```
doc_term_matrix=matrix.todense()
doc_term_matrix

# Computing cosine similarity
df=pd.DataFrame(doc_term_matrix)

from sklearn.metrics.pairwise import cosine_similarity
print(cosine_similarity(df,df))

#> [[1.          0.45584231]
#> [0.45584231 1.          ]]
```

To understand more about the maths behind this or for similar problems, check this post

<https://www.machinelearningplus.com/nlp/cosine-similarity/>

## 21. How to find soft cosine similarity of documents ?

Difficulty Level : L3

Q. Compute the soft cosine similarity of the given documents

Hint: [Soft Cosine Similarity](#)

Input :

```
doc_soup = "Soup is a primarily liquid food, generally served warm  
doc_noodles = "Noodles are a staple food in many cultures. They are  
doc_dosa = "Dosa is a type of pancake from the Indian subcontinent,  
doc_trump = "Mr. Trump became president after winning the political  
doc_election = "President Trump says Putin had no political interfe  
doc_putin = "Post elections, Vladimir Putin became President of Rus
```

Desired Output :

```
0.5842470477718544
```

▼ Show Solution

```
# Prepare a dictionary and a corpus.
```

```
dictionary = corpora.Dictionary([simple_preprocess(doc) for doc i

# Prepare the similarity matrix
similarity_matrix = fasttext_model300.similarity_matrix(dictionary

# Convert the sentences into bag-of-words vectors.
sent_1 = dictionary.doc2bow(simple_preprocess(doc_trump))
sent_2 = dictionary.doc2bow(simple_preprocess(doc_election))
sent_3 = dictionary.doc2bow(simple_preprocess(doc_putin))
sent_4 = dictionary.doc2bow(simple_preprocess(doc_soup))
sent_5 = dictionary.doc2bow(simple_preprocess(doc_noodles))
sent_6 = dictionary.doc2bow(simple_preprocess(doc_dosa))

sentences = [sent_1, sent_2, sent_3, sent_4, sent_5, sent_6]

# Compute soft cosine similarity
print(softcossim(sent_1, sent_2, similarity_matrix))

#> 0.5842470477718544
```

Difficulty Level : L2

Q. Find all similar words to “amazing” using Google news Word2Vec.

## 22. How to find similar words using pre-trained Word2Vec?

Desired Output:

```
#> [('incredible', 0.90),  
#> ('awesome', 0.82),  
#> ('unbelievable', 0.82),  
#> ('fantastic', 0.77),  
#> ('phenomenal', 0.76),  
#> ('astounding', 0.73),  
#> ('wonderful', 0.72),  
#> ('unbelievable', 0.71),  
#> ('remarkable', 0.70),  
#> ('marvelous', 0.70)]
```

✓ Show Solution

```
# Import gensim api  
import gensim.downloader as api  
  
# Load the pretrained google news word2vec model  
word2vec_model300 = api.load('word2vec-google-news-300')  
  
# Using most_similar() function
```

```
word2vec_model300.most_similar('amazing')

#> [('incredible', 0.9054000973701477),
#> ('awesome', 0.8282865285873413),
#> ('unbelievable', 0.8201264142990112),
#> ('fantastic', 0.778986930847168),
#> ('phenomenal', 0.7642048001289368),
#> ('astounding', 0.7347068786621094),
#> ('wonderful', 0.7263179421424866),
#> ('unbelievable', 0.7165080904960632),
#> ('remarkable', 0.7095627188682556),
#> ('marvelous', 0.7015583515167236)]
```

## 23. How to compute Word mover distance?

Difficulty Level : L3

Q. Compute the word mover distance between given two texts

Input :



```
sentence_orange = 'Oranges are my favorite fruit'  
sent="apples are not my favorite"
```

Desired Output :

5.378

✓ Show Solution

```
# Impting gensim 's Word2Vec model  
import gensim  
  
from gensim.models import Word2Vec  
model=Word2Vec()  
  
sentence_orange = 'Oranges are my favorite fruit'  
sent="apples are not my favorite"  
  
# Computing the word mover distance  
distance = model.wmdistance(sent, sentence_orange)  
  
#> 5.378
```

## 24. How to replace all the pronouns in a text with their respective object names

Difficulty Level : L2

Q. Replace the pronouns in below text by the respective object names

Input :

```
text=" My sister has a dog and she loves him"
```

Desired Output :

```
[My sister,she]  
[a dog ,him ]
```

✓ Show Solution

```
# Import neural coref library
!pip install neuralcoref
import spacy
import neuralcoref

# Add it to the pipeline
nlp = spacy.load('en')
neuralcoref.add_to_pipe(nlp)

# Printing the coreferences
doc1 = nlp('My sister has a dog. She loves him.')
print(doc1._.coref_clusters)
```

spaCy also provides the feature of visualizing the coreferences. Check out this <https://spacy.io/universe/project/neuralcoref-vizualizer/>.

## 25. How to extract topic keywords using LSA?

Difficulty Level : L3

Q. Extract the topic keywords from the given texts using LSA(Latent Semantic Analysis )

Input :

```
texts= ["""It's all about travel. I travel a lot.  those who do no
      "" " You can learn a lot about yourself through travelling.
      "" "Some of my most cherished memories are from the times wh
      "" "If you travel, you may learn a lot of useful things. The
      "" "After arriving home from a long journey, a lot of travel
      "" "The benefits of travel are not just a one-time thing: tr
      "" "Sure, you probably feel comfortable where you are, but t
      "" " So, travel makes you cherish life. Let's travel more .
      ]
```

Desired Output :

```
#> Topic 0:
#> learn new life travelling country feel
#> Topic 1:
```

```
#> life cherish diaries let share experience
#> Topic 2:
#> feel know time people just regions
#> Topic 3:
#> time especially cherish diaries let share
..(truncated)..
```

▼ Show Solution

```
# Importing the Tf-idf vectorizer from sklearn
from sklearn.feature_extraction.text import TfidfVectorizer

# Defining the vectorizer
vectorizer = TfidfVectorizer(stop_words='english', max_features=

# Transforming the tokens into the matrix form through .fit_trans
matrix= vectorizer.fit_transform(texts)

# SVD represent documents and terms in vectors
from sklearn.decomposition import TruncatedSVD
SVD_model = TruncatedSVD(n_components=10, algorithm='randomized',
SVD_model.fit(matrix)

# Getting the terms
terms = vectorizer.get_feature_names()

# Iterating through each topic
```

```
for i, comp in enumerate(SVD_model.components_):
    terms_comp = zip(terms, comp)
    # sorting the 7 most important terms
    sorted_terms = sorted(terms_comp, key= lambda x:x[1], reverse
    print("Topic "+str(i)+" : ")
    # printing the terms of a topic
    for t in sorted_terms:
        print(t[0],end=' ')
    print(' ')
```

```
#> Topic 0:
#> learn new life travelling country feel
#> Topic 1:
#> life cherish diaries let share experience
#> Topic 2:
#> feel know time people just regions
#> Topic 3:
#> time especially cherish diaries let share
#> Topic 4:
#> cherish diaries let share makes feel
#> Topic 5:
#> culture augustine course cultural cultures eyes
#> Topic 6:
#> want experiences life things advantage bad
#> Topic 7:
#> observe feel experiences want skills test
```

## 26. How to extract topic Keywords using LDA ?

Difficulty Level : L3

Q. Extract the the topics from the given texts with the help of LDA(Latent dirichlet algorithm)

Input :

```
texts= [""It's all about travel. I travel a lot.  those who do not  
"" You can learn a lot about yourself through travelling.  
""Some of my most cherished memories are from the times wh  
""If you travel, you may learn a lot of useful things. The  
""After arriving home from a long journey, a lot of travel  
""The benefits of travel are not just a one-time thing: tr  
""Sure, you probably feel comfortable where you are, but t  
"" So, travel makes you cherish life. Let's travel more .  
]
```

Desired Output :

```
[(0, '0.068*"travel" + 0.044*"learn" + 0.027*"country" + 0.027*"If'
```

▼ Show Solution

```
# Import gensim, nltk
import gensim
from gensim import models, corpora
import nltk
from nltk.corpus import stopwords

# Before topic extraction, we remove punctuations and stopwords.
my_stopwords=set(stopwords.words('english'))
punctuations=['.', '!', ',', '"', 'You', 'I']

# We prepare a list containing lists of tokens of each text
all_tokens=[]
for text in texts:
    tokens=[]
    raw=nltk.wordpunct_tokenize(text)
    for token in raw:
        if token not in my_stopwords:
            if token not in punctuations:
                tokens.append(token)
```



```
all_tokens.append(tokens)

# Creating a gensim dictionary and the matrix
dictionary = corpora.Dictionary(all_tokens)
doc_term_matrix = [dictionary.doc2bow(doc) for doc in all_tokens]

# Building the model and training it with the matrix
from gensim.models.ldamodel import LdaModel
model = LdaModel(doc_term_matrix, num_topics=5, id2word = diction

print(model.print_topics(num_topics=6,num_words=5))
```

To understand more about how LDA works , check out our <https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/> .

## 27. How to extract topic keywords using NMF?

Difficulty Level : L3

Q. Extract the the topics from the given texts with the help of NMF(Non-negative Matrix Factorization method)

Input :

```
texts= ["""It's all about travel. I travel a lot.  those who do no
        "" " You can learn a lot about yourself through travelling.
        "" "Some of my most cherished memories are from the times wh
        "" "If you travel, you may learn a lot of useful things. The
        "" "After arriving home from a long journey, a lot of travel
        "" "The benefits of travel are not just a one-time thing: tr
        "" "Sure, you probably feel comfortable where you are, but t
        "" " So, travel makes you cherish life. Let's travel more .
        ]
```

Desired Output:

Topic 0:

```
[('new', 0.6329770846997606), ('learn', 0.49810389825931783), ('sp
```

Topic 1:

```
[('life', 0.34063551920788737), ('home', 0.31402014643240667), ('e
Topic 2:
[('feel', 0.3462484013922396), ('know', 0.28400088182008115), ('pe
...(truncated)
```

✓ Show Solution

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Defining the vectorizer
vectorizer = TfidfVectorizer(stop_words='english', max_features=

# Transforming the tokens into the matrix form through .fit_trans
nmf_matrix= vectorizer.fit_transform(texts)
from sklearn.decomposition import NMF
nmf_model = NMF(n_components=6)
nmf_model.fit(nmf_matrix)

# Function to print topics
def print_topics_nmf(model, vectorizer, top_n=6):
    for idx, topic in enumerate(model.components_):
        print("Topic %d:" % (idx))
        print([(vectorizer.get_feature_names()[i], topic[i])
                for i in topic.argsort()[:-top_n - 1:-1]])

print_topics_nmf(nmf_model, vectorizer)
```

```
#> Topic 0:  
#> [('new', 0.6329770846997606), ('learn', 0.49810389825931783),  
#> Topic 1:  
#> [('life', 0.34063551920788737), ('home', 0.31402014643240667),  
#> Topic 2:  
#> [('feel', 0.3462484013922396), ('know', 0.28400088182008115),  
#> Topic 3:  
#> [('time', 0.44163173193053806), ('especially', 0.2944211546203  
#> Topic 4:  
#> [('cherish', 0.4703713910017504), ('diaries', 0.47037139100175  
#> Topic 5:  
#> [('learn', 0.2790596001102511), ('culture', 0.2285890660745815
```

## 28. How to classify a text as positive/negative sentiment

Difficulty Level : L2

Q. Detect if a text is positive or negative sentiment

Input :

```
text="It was a very pleasant day"
```

Desired Output:

```
Sentiment(polarity=0.9533333333333333, subjectivity=1.0)
Positive
```

▼ Show Solution

```
# Sentiment analysis with TextBlob
from textblob import TextBlob
blob=TextBlob(text)

# Using the sentiment attribute
print(blob.sentiment)
if(blob.sentiment.polarity > 0):
    print("Positive")

#> Sentiment(polarity=0.9533333333333333, subjectivity=1.0)
#> Positive
```

---

Note that the magnitude of **polarity** represents the extent/intensity . If it the polarity is greater than 0 , it represents positive sentiment and vice-versa.

## 29. How to use the Word2Vec model for representing words?

Difficulty Level : L2

Q. Extract the word vector representation of the word using word2vec model

Input :

```
texts= [" Photography is an excellent hobby to pursue ",  
        " Photographers usually develop patience, calmnesss"  
        " You can try Photography with any good mobile too"]
```

Desired Output:

```
array([ 2.94046826e-03, -1.31368915e-05, -3.43682081e-03, -3.73881
       2.49790819e-03, -1.23431312e-03, -9.60227044e-04,  2.313456
      -4.97973803e-03,  2.09524506e-03,  2.00997619e-03, -4.104598
       8.42132606e-04, -2.70003616e-03,  3.12150107e-03,  1.236076
       2.16376456e-03,  5.02903073e-04, -3.72780557e-03,  4.352665
      -1.80016900e-03,  3.42973252e-03, -2.12087762e-03,  1.145313
       3.03449039e-03, -8.75897415e-04, -3.50620854e-03,  5.103226
      ...(truncated)
```

Positive

▼ Show Solution

```
# We prepare a list containing lists of tokens of each text
tokens=[]
for text in texts:
    tokens=[]
    raw=nlTK.wordpunct_tokenize(text)
    for token in raw:
        tokens.append(token)
    all_tokens.append(tokens)

# Import and fit the model with data
import gensim
from gensim.models import Word2Vec
model=Word2Vec(all_tokens)
```

```
# Getting the vector representation of a word
model['Photography']
```

```
array([ 2.94046826e-03, -1.31368915e-05, -3.43682081e-03, -3.7388
        2.49790819e-03, -1.23431312e-03, -9.60227044e-04,  2.3134
       -4.97973803e-03,  2.09524506e-03,  2.00997619e-03, -4.1045
        8.42132606e-04, -2.70003616e-03,  3.12150107e-03,  1.2360
        2.16376456e-03,  5.02903073e-04, -3.72780557e-03,  4.3526
       -1.80016900e-03,  3.42973252e-03, -2.12087762e-03,  1.1453
        3.03449039e-03, -8.75897415e-04, -3.50620854e-03,  5.1032
        2.36228597e-03,  3.20315338e-03, -1.77754264e-03,  3.4404
       -4.72177169e-04,  3.79201653e-03,  3.50930146e-03,  9.2463
       -3.63159878e-03,  4.49452689e-03, -1.94674812e-03,  2.6679
        3.57741816e-03,  4.08058614e-03, -4.22306563e-04,  3.2155
        1.93726353e-03, -4.70201066e-03, -6.77402073e-04,  3.5747
        2.40847061e-04, -3.06745851e-03, -3.21992044e-03, -2.7757
        1.84161821e-03, -2.28599668e-03,  1.12327258e-03,  4.9077
       -3.74632655e-03,  4.14755428e-03, -1.51176169e-03, -2.4668
       -2.91575165e-03,  1.66514842e-03, -2.64900009e-04,  4.1762
       -1.15438248e-03,  3.30674206e-03,  3.89241078e-03,  1.0731
       -3.56393168e-03,  4.21310542e-03, -3.83528182e-03,  4.8784
        3.38425953e-03,  5.87464485e-04,  1.10692088e-03,  1.8232
        3.44771869e-03,  2.54350528e-03, -3.22796614e-03,  4.8392
       -4.45320550e-03,  4.85936319e-03, -3.69266351e-03, -1.2624
        4.05845884e-03,  2.44187587e-03,  1.55774585e-03, -1.9790
       -2.21285340e-03,  1.51218695e-03, -1.10817770e-03, -1.9192
```



```
3.81433661e-03, -9.82026220e-04, -8.55478633e-04, 1.7392  
-9.87094129e-04, 1.61158561e-03, 1.61566911e-03, -6.7710  
dtype=float32)
```

## 30. How to visualize the word embedding obtained from word2Vec model ?

Difficulty Level : L4

Q. Implement Word embedding on the given texts and visualize it

```
texts= [" Photography is an excellent hobby to pursue ",  
        " Photographers usually develop patience, calmnesss"  
        " You can try Photography with any good mobile too"]
```

✓ Show Solution

```
# We prepare a list containing lists of tokens of each text
all_tokens=[]
for text in texts:
    tokens=[]
    raw=nlTK.wordpunct_tokenize(text)
    for token in raw:
        tokens.append(token)
    all_tokens.append(tokens)

# Import and fit the model with data
import gensim
from gensim.models import Word2Vec
model=Word2Vec(all_tokens)

# Visualizing the word embedding
from sklearn.decomposition import PCA
from matplotlib import pyplot

X = model[model.wv.vocab]
pca = PCA(n_components=2)
result = pca.fit_transform(X)
# create a scatter plot of the projection
pyplot.scatter(result[:, 0], result[:, 1])
words = list(model.wv.vocab)
for i, word in enumerate(words):
```

```
pyplot.annotate(word, xy=(result[i, 0], result[i, 1]))  
pyplot.show()
```

Word2Vec Representation

## 31. How to represent the document using Doc2Vec model?

Difficulty Level : L2

Q. Represent a text document in the form a vector

Input :

```
texts= [" Photography is an excellent hobby to pursue ",
        " Photographers usually develop patience, calmnesss"
        " You can try Photography with any good mobile too"]
```

Desired Output:

```
array([ 2.6586275e-03,  3.2867077e-03, -2.0473711e-03,  6.0251489e-
-1.5340233e-03,  1.5060971e-03,  1.0988972e-03,  1.0712545e-
-4.3745534e-03, -4.0448168e-03, -1.8953394e-04, -2.0953947e-
-3.3285557e-03,  1.0409033e-03, -8.5728493e-04,  4.5999791e-
...(truncated)..
```

✓ Show Solution

```
# Importing the model
from gensim.models import Doc2Vec

# Preparing data in the format and fitting to the model
def tagged_document(list_of_list_of_words):
    for i, list_of_words in enumerate(list_of_list_of_words):
        yield gensim.models.doc2vec.TaggedDocument(list_of_words, [
```

```
my_data = list(tagged_document(all_tokens))
model=Doc2Vec(my_data)

model.infer_vector(['photography','is','an',' excellent ','hobby
```

```
array([ 2.6586275e-03,  3.2867077e-03, -2.0473711e-03,  6.0251489e-03,
       -1.5340233e-03,  1.5060971e-03,  1.0988972e-03,  1.0712545e-03,
       -4.3745534e-03, -4.0448168e-03, -1.8953394e-04, -2.0953947e-03,
       -3.3285557e-03,  1.0409033e-03, -8.5728493e-04,  4.5999791e-03,
        1.8428586e-03,  2.9749258e-03,  4.8927322e-04, -4.1088923e-03,
       -1.2474873e-03,  4.5802444e-03,  2.4389643e-03, -4.2193010e-03,
       -2.4726104e-03,  2.4501325e-03,  3.3282219e-03, -3.0891516e-03,
        3.2441942e-03, -1.2857418e-03, -8.4910257e-04, -1.0371304e-03,
        4.3518590e-03,  1.3085983e-03,  4.8915138e-03,  1.9108410e-03,
       -2.3149159e-03, -2.8708300e-03,  3.5418086e-03,  4.3390174e-03,
        2.7052627e-03,  4.1727605e-03, -3.7339646e-03,  4.4227624e-03,
        3.5092062e-03,  1.0140887e-03, -1.2085630e-03, -1.5898966e-03,
       -1.0424303e-03,  2.5275371e-03, -4.4435970e-03,  2.9752296e-03,
        4.6713585e-03,  4.1678254e-03, -1.3408947e-03, -4.1671298e-03,
       -5.3989125e-04,  2.3537579e-03,  4.9786703e-03, -2.0938511e-03,
       -4.0806020e-03, -3.6052074e-03,  1.2903051e-03, -4.2635379e-03,
       -3.6390694e-03, -3.3433773e-03,  3.6569773e-03, -1.8581208e-03,
        1.3781790e-04, -1.6561428e-03, -4.5162151e-03,  2.0534093e-03,
       -2.7264019e-03, -1.7743753e-03, -2.7915081e-03, -1.1389129e-03,
        4.9526147e-03,  3.7630240e-03, -1.9377380e-03,  1.6532684e-03,
        4.9404724e-04,  3.4463860e-03,  2.6799906e-03,  1.6751935e-03])
```

```
-6.6813978e-04, 3.6566416e-03, 2.5076446e-05, 1.9042364  
-1.0040828e-03, -8.4077887e-04, 3.3536348e-03, -1.2608888  
-4.6293526e-03, 2.6570701e-03, -3.4919968e-03, 8.2246581  
6.5824442e-04, 1.2701214e-04, 3.8290059e-04, -3.5389795  
dtype=float32)
```

To understand more about how to `gensim` library's features ,  
check out our detailed post  
[/https://www.machinelearningplus.com/nlp/gensim-tutorial/](https://www.machinelearningplus.com/nlp/gensim-tutorial/)

## 32. How to extract the TF-IDF Matrix ?

Difficulty Level : L3

Q. Extract the TF-IDF (Term Frequency -Inverse Document Frequency) Matrix for the given list of text documents

Input :

```
text_documents=['Painting is a hobby for many , passion for some',
               'My hobby is coin collection'
               'I do some Painting every now and then']
```

Desired Output:

```
(0, 13) 0.2511643891128359
(0, 12) 0.35300278529739293
(0, 8) 0.35300278529739293
(0, 5) 0.7060055705947859
(0, 6) 0.2511643891128359
(0, 7) 0.2511643891128359
...(truncated)..
```

✓ Show Solution

```
# Method 1-Using gensim

from gensim import corpora
from gensim.utils import simple_preprocess
doc_tokenized = [simple_preprocess(text) for text in text_documents]
dictionary = corpora.Dictionary()
```

```
# Creating the Bag of Words from the docs
BoW_corpus = [dictionary.doc2bow(doc, allow_update=True) for doc
for doc in BoW_corpus:
    print([[dictionary[id], freq] for id, freq in doc])
import numpy as np
tfidf = models.TfidfModel(BoW_corpus)

#> [['for', 2], ['hobby', 1], ['is', 1], ['many', 1], ['painting'
#> [['hobby', 1], ['is', 1], ['painting', 1], ['some', 1], ['and'
```

```
# Method 2- Using sklearn's TfidfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

# Fit the vectorizer to our text documents
vectorizer = TfidfVectorizer()
matrix = vectorizer.fit_transform(text_documents)
print(matrix)

#> (0, 13)      0.2511643891128359
#> (0, 12)      0.35300278529739293
#> (0, 8)       0.35300278529739293
#> (0, 5)       0.7060055705947859
#> (0, 6)       0.2511643891128359
#> (0, 7)       0.2511643891128359
#> (0, 11)      0.2511643891128359
#> (1, 14)      0.31583360086881224
```



```
#> (1, 0)      0.31583360086881224
#> (1, 10)     0.31583360086881224
#> (1, 4)      0.31583360086881224
#> (1, 3)      0.31583360086881224
#> (1, 2)      0.31583360086881224
#> (1, 1)      0.31583360086881224
#> (1, 9)      0.31583360086881224
#> (1, 13)     0.22471820826199104
#> (1, 6)      0.22471820826199104
#> (1, 7)      0.22471820826199104
#> (1, 11)     0.22471820826199104
```

## 33. How to create bigrams using Gensim's Phraser ?

Difficulty Level : L3

Q. Create bigrams from the given texts using Gensim library's Phrases

Input :

```
documents = ["the mayor of new york was there", "new york mayor wa
```

Desired Output:

```
['the', 'mayor', 'of', 'new york', 'was', 'there']  
['new york', 'mayor', 'was', 'present']
```

▼ Show Solution

```
# Import Phraser from gensim  
from gensim.models import Phrases  
from gensim.models.phrases import Phraser  
  
sentence_stream = [doc.split(" ") for doc in documents]  
  
# Creating bigram phraser  
bigram = Phrases(sentence_stream, min_count=1, threshold=2, delim  
bigram_phraser = Phraser(bigram)  
  
for sent in sentence_stream:  
    tokens_ = bigram_phraser[sent]  
    print(tokens_)
```

## 34. How to create bigrams, trigrams using ngrams ?

Difficulty Level : L3

Q. Extract all bigrams , trigrams using `ngrams` of `nltk` library

Input :

```
Sentences="Machine learning is a neccessary field in today's world"
```

Desired Output:

```
Bigrams are [('machine', 'learning'), ('learning', 'is'), ('is', 'a'), ('a', 'neccessary'), ('neccessary', 'field'), ('field', 'in'), ('in', 'today'), ('today', 's'), ('s', 'world')]
Trigrams are [('machine', 'learning', 'is'), ('learning', 'is', 'a'), ('is', 'a', 'neccessary'), ('a', 'neccessary', 'field'), ('neccessary', 'field', 'in'), ('field', 'in', 'today'), ('in', 'today', 's'), ('today', 's', 'world')]
```

✓ Show Solution

```
# Creating bigrams and trigrams
from nltk import ngrams
bigram=list(ngrams(Sentences.lower().split(),2))
trigram=list(ngrams(Sentences.lower().split(),3))

print(" Bigrams are",bigram)
print(" Trigrams are", trigram)
```

## 35. How to detect the language of entered text ?

Difficulty Level : L1

Q. Find out the language of the given text

Input :

```
text="El agente imprime su pase de abordaje. Los oficiales de segu
```

Desired Output:

```
{'language': 'es', 'score': 0.9999963653206719}  
El agente imprime su pase de abordaje. {'language': 'es', 'score':
```

▼ Show Solution

```
# Install spacy's languagedetect library  
import spacy  
!pip install spacy_langdetect  
from spacy_langdetect import LanguageDetector  
nlp = spacy.load('en')  
  
# Add the language detector to the processing pipeline  
nlp.add_pipe(LanguageDetector(), name='language_detector', last=True)  
  
doc = nlp(text)  
# document level language detection. Think of it like average language  
print(doc._.language)  
# sentence level language detection  
for sent in doc.sents:  
    print(sent, sent._.language)
```

```
#> {'language': 'es', 'score': 0.9999963653206719}  
#> El agente imprime su pase de abordaje. {'language': 'es', 'sco  
#> Los oficiales de seguridad del aeropuerto pasan junto a él con  
#> El perro está olfateando alrededor del equipaje de las persona
```

## 36. How to merge two tokens as one ?

Difficulty Level : L3

Q. Merge the first name and last name as single token in the given sentence

Input:

```
text="Robert Langdon is a famous character in various books and mo
```

Desired Output:

Robert Langdon  
is  
a  
famous  
character  
in  
various  
books  
and  
movies

▼ Show Solution

```
# Using retokenize() method of Doc object to merge two tokens

doc = nlp(text)
with doc.retokenize() as retokenizer:
    retokenizer.merge(doc[0:14])

for token in doc:
    print(token.text)

#> Robert Langdon
#> is
#> a
#> famous
```

```
#> character  
#> in  
#> various  
#> books  
#> and  
#> movies
```

## 37. How to extract Noun phrases from a text ?

Difficulty Level : L2

Q. Extract and print the noun phrases in given text document

Input:

```
text="There is a empty house on the Elm Street"
```

Expected Output :



```
[a empty house, the Elm Street]
```

✓ Show Solution

```
# Create a spacy doc of the text
doc = nlp(text)

# Use `noun_chunks` attribute to extract the Noun phrases
chunks = list(doc.noun_chunks)
chunks

#> [a empty house, the Elm Street]
```

## 38. How to extract Verb phrases from the text ?

Difficulty Level : L3

Q. Extract the Verb Phrases from the given text

Input :

```
text=("I may bake a cake for my birthday. The talk will introduce I
```

Desired Output:

```
may bake
will introduce
```

▼ Show Solution

```
# Import textacy library
!pip install textacy
import textacy

# Regex pattern to identify verb phrase
pattern = r'(<VERB>?<ADV>*<VERB>+)'
doc = textacy.make_spacy_doc(text, lang='en_core_web_sm')

# Finding matches
verb_phrases = textacy.extract.pos_regex_matches(doc, pattern)

# Print all Verb Phrase
for chunk in verb_phrases:
    print(chunk.text)
```

```
#> may bake  
#> will introduce
```

## 39. How to extract first name and last names present in the document ?

Difficulty Level : L3

Q. Extract any two consecutive Proper Nouns that occur in the text document

Input :

```
text="Sherlock Holmes and Clint Thomas were good friends. I am a f
```



Desired Output:

Sherlock Holmes  
Clint Thomas  
John Mark

▼ Show Solution

```
# Import and initialize spacy's matcher
from spacy.matcher import Matcher
matcher = Matcher(nlp.vocab)
doc=nlp(text)

# Function that adds patterns to the matcher and finds the respec
def extract_matches(doc):
    pattern = [{'POS': 'PROPN'}, {'POS': 'PROPN'}]
    matcher.add('FULL_NAME', None, pattern)
    matches = matcher(doc)

    for match_id, start, end in matches:
        span = doc[start:end]
        print(span.text)

extract_matches(doc)

#> Sherlock Holmes
#> Clint Thomas
#> John Mark
```



## 40. How to identify named entities in the given text

Difficulty Level : L2

Q. Identify and print all the named entities with their labels in the below text

Input

```
text=" Walter works at Google. He lives in London."
```

Desired Output:

```
Walter PERSON  
Google ORG  
London GPE
```

✓ Show Solution

```
# Load spacy modelimport spacy
nlp=spacy.load("en_core_web_sm")doc=nlp(text)
# Using the ents attribute of doc, identify labels
for entity in doc.ents:
    print(entity.text,entity.label_)

#> Walter PERSON
#> Google ORG
#> London GPE
```

## 41. How to identify all the names of Organizations present in the text with NER ?

Difficulty Level : L2

Q. Identify and extract a list of all organizations/Companies mentioned in the given news article

Input :

```
text =" Google has released it's new model which has got attention
```

Expected Solution

```
['Google', 'Amazon', 'Apple', 'Flipkart']
```

✓ Show Solution

```
doc=nlp(text)
list_of_org=[]
for entity in doc.ents:
    if entity.label_=="ORG":
        list_of_org.append(entity.text)

print(list_of_org)

#> ['Google', 'Amazon', 'Apple', 'Flipkart']
```

Difficulty Level : L3

## 42. How to replace all names of people in the text with 'UNKNOWN'

Q. Identify and replace all the person names in the news article with UNKNOWN to keep privacy  
Input :

```
news=" Walter was arrested yesterday at Brooklyn for murder. The su
```

Desired Output :

```
' UNKNOWN was arrested yesterday at Brooklyn for murder . The sus
```

▼ Show Solution

```
doc=nlp(news)

# Identifying the entities of category 'PERSON'
entities = [entity.text for entity in doc.ents if entity.label_
updated_text=[]
```



```
for token in doc:
    if token.text in entities:
        updated_text.append("UNKNOWN")
    else :
        updated_text.append(token.text)

" ".join(updated_text)

#> ' UNKNOWN was arrested yesterday at Brooklyn for murder . The
```

## 43. How to visualize the named entities using spaCy

Difficulty Level : L2

Q. Display the named entities present in the given document along with their categories using spacy

Input :

```
text=" Walter was arrested yesterday at Brooklyn for murder. The s
```

✓ Show Solution

```
# Use spacy's displacy with the parameter style="ent"

from spacy import displacy
doc=nlp(text)
displacy.render(doc, style='ent', jupyter=True)
```

## 44. How to implement dependency parsing ?

Difficulty Level : L2

Q. Find the dependencies of all the words in the given text

Input :

```
text="Mark plays volleyball every evening."
```

Desired Output :

```
Mark nsubj
plays ROOT
volleyball dobj
every det
evening npadvmod
. punct
```

✓ Show Solution

```
# Using dep_ attribute of tokens in spaCy to access the dependencies
doc=nlp(text)

for token in doc:
    print(token.text,token.dep_)

#> Mark nsubj
#> plays ROOT
```

```
#> volleyball dobj  
#> every det  
#> evening npadvmod  
#> . punct
```

## 45. How to find the ROOT word of any word in a sentence?

Difficulty Level : L3

Q. Find and print the root word / headword of any word in the given sentence

Input :

```
text="Mark plays volleyball. Sam is not into sports, he paints a 10
```

Desired Output :

```
Mark plays
plays plays
volleyball plays
. plays
Sam is
is paints
not is
into is
sports into
, paints
he paints
paints paints
a lot
lot paints
```

▼ Show Solution

```
# use the head attribute of tokens to find it's rootword
doc=nlp(text)
for token in doc:
    print(token.text,token.head)

#> Mark plays
#> plays plays
#> volleyball plays
#> . plays
#> Sam is
```

```
#> is paints  
#> not is  
#> into is  
#> sports into  
#> , paints  
#> he paints  
#> paints paints  
#> a lot  
#> lot paints
```

## 46. How to visualize the dependency tree in spaCy

Difficulty Level : L2

Q. Visualize the dependencies of various tokens of the given text using spaCy

Input :

```
text="Mark plays volleyball. Sam is not into sports, he paints a lot"
```

▼ Show Solution

```
# Use spacy's displacy with the parameter style="dep"
doc=nlp(text)

from spacy import displacy
displacy.render(doc, style='dep', jupyter=True)
```

## 47. How to detect all the Laptop names present in the text ?

Difficulty Level : L4

Q. Detect all the Laptop names present in the given document .

Input :

```
text="For my offical use, I prefer lenova. For gaming purposes, I .
```

Expected Output

```
lenova laptop
asus laptop
```

✓ Show Solution

```
# Import EntityRuler of spacy model
import spacy
nlp=spacy.load("en_core_web_sm")
from spacy.pipeline import EntityRuler

# Functions to create patterns of laptop name to match
def create_versioned(name):
    return [
        [{'LOWER': name}],
        [{'LOWER': {'REGEX': f'({name}\d+\.\d*\.\d*)'}}],
    ]
```



```

[{'LOWER': name}, {'TEXT': {'REGEX': '(\d+\.\d*\.\d*)'}}]

def create_patterns():
    versioned_languages = ['dell', 'HP', 'asus', 'msi', 'Apple', 'HC']
    flatten = lambda l: [item for sublist in l for item in sublist]
    versioned_patterns = flatten([create_versioned(lang) for lang

lang_patterns = [
    [{'LOWER': 'dell'}, {'LIKE_NUM': True}],
    [{'LOWER': 'HP'}],
    [{'LOWER': 'asus'}, {'LOWER': '#'}],
    [{'LOWER': 'msi'}, {'LOWER': 'sharp'}],
    [{'LOWER': 'Apple'}],
    [{'LOWER': 'HCL'}, {'LOWER': '#'}],
    [{'LOWER': 'sony'}],
    [{'LOWER': 'samsung'}],
    [{'LOWER': 'toshiba'}],
    [{'LOWER': 'dell'}, {'LOWER': 'inspiron'}],
    [{'LOWER': 'acer'}, {'IS_PUNCT': True, 'OP': '?'}, {'LOWER'
    [{'LOWER': 'golang'}],
    [{'LOWER': 'lenova'}],
    [{'LOWER': 'HP'}, {'LOWER': 'gaming'}],
    [{'LOWER': 'Fujitsu'}],
    [{'LOWER': 'micromax'}],
]

return versioned_patterns + lang_patterns

```

```
# Add the Entity Ruler to the pipeline
ruler=EntityRuler(nlp)
ruler.add_patterns([{'label':'laptop','pattern':p} for p in creat
nlp.add_pipe(ruler)

# Identify the car names now
doc=nlp("For my offical use, I prefer lenova. For gaming purposes
for ent in doc.ents:
    print(ent.text,ent.label_)

#> lenova laptop
#> asus laptop
```

## 48. How to summarize text using gensim ?

Difficulty Level : L3

Q. Extract the summary of the given text based using **gensim** package based on the TextRank Algorithm.

Input :

original\_text="""Studies show that exercise can treat mild to moderate depression. Exercise is a powerful depression fighter for several reasons. Most people who exercise regularly tend to do so because it gives them a sense of accomplishment. Regular exercise can have a profoundly positive impact on depression. Have you ever noticed how your body feels when you're under stress? Your muscles tense up, and your heart rate increases. Exercising is an effective way to break this cycle. As well as releasing endorphins, exercise helps to regulate the body's stress response. Instead of allowing your mind to wander, pay close attention to the physical sensations of movement. Outdoor activities like hiking, sailing, mountain biking, rock climbing, and swimming can be particularly beneficial. The key is to find an activity you enjoy and stick to it. Consistency is key. Even a short walk can make a difference. Don't be discouraged if you don't feel the benefits immediately. It takes time for the body to adjust. Start small and build up gradually. The more you exercise, the more you'll notice the positive effects. Exercise isn't just about physical health; it's about mental health too. So, lace up those sneakers and get moving. Your mind will thank you. """

Desired Output :

As one example, a recent study done by the Harvard T.H. Chan School of Public Health found that, no matter your age or fitness level, you can learn to use exercise to manage stress. The worry and discomfort of all these physical symptoms can in turn be reduced by regular physical activity. As well as releasing endorphins in the brain, physical activity helps to reduce the levels of stress hormones in the body.

▼ Show Solution

```
# Importing the summarize function from gensim module
import gensim
```

```
from gensim.summarization.summarizer import summarize

# Pass the document along with desired word count to get the summ
my_summary=summarize(original_text,word_count=100)
print(my_summary)

#> As one example, a recent study done by the Harvard T.H. Chan S
#> No matter your age or fitness level, you can learn to use exer
#> The worry and discomfort of all these physical symptoms can in
#> As well as releasing endorphins in the brain, physical activit
```

## 49. How to summarize text based on the LexRank algorithm ?

Difficulty Level : L3

Q. Extract the summary of the given text based on the TextRank Algorithm.

Input :

```
original_text="""Studies show that exercise can treat mild to moderate depression. Exercise is a powerful depression fighter for several reasons. Most people who exercise regularly tend to do so because it gives them a sense of accomplishment. Regular exercise can have a profoundly positive impact on depression. Have you ever noticed how your body feels when you're under stress? Your muscles tense up and your heart races. Exercising is an effective way to break this cycle. As well as releasing stress, exercise also helps to improve your mood. Instead of allowing your mind to wander, pay close attention to the present moment. Outdoor activities like hiking, sailing, mountain biking, rock climbing, and gardening are all great ways to get some exercise and enjoy the outdoors. So, if you're looking for a way to improve your mental health, try exercising regularly. You'll be surprised at how much better you feel.


```

Desired Output :

```
Since the body and mind are so closely linked, when your body feels stressed, your mind is also stressed. Exercise is a powerful depression fighter for several reasons. Most people who exercise regularly tend to do so because it gives them a sense of accomplishment. Regular exercise can have a profoundly positive impact on depression. Have you ever noticed how your body feels when you're under stress? Your muscles tense up and your heart races. Exercising is an effective way to break this cycle. As well as releasing stress, exercise also helps to improve your mood. Instead of allowing your mind to wander, pay close attention to the present moment. Outdoor activities like hiking, sailing, mountain biking, rock climbing, and gardening are all great ways to get some exercise and enjoy the outdoors. So, if you're looking for a way to improve your mental health, try exercising regularly. You'll be surprised at how much better you feel.


```

▼ Show Solution

```
import sumy
from sumy.summarizers.lex_rank import LexRankSummarizer

#Plain text parsers since we are parsing through text
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer

parser=PlaintextParser.from_string(original_text,Tokenizer("english"))
summarizer=LexRankSummarizer(parser)
summary=summarizer.summarize(original_text,10)
print(summary)


```

```
summarizer=LexRankSummarizer()  
my_summary=summarizer(parser.document,2)  
print(my_summary)  
  
#> (<Sentence: Since the body and mind are so closely linked, whe
```

## 50. How to summarize text using Luhn algorithm?

Q. Extract the summary of the given text based on the Luhn Algorithm.

Difficulty Level : L3

Input :

```
original_text="""Studies show that exercise can treat mild to moderate  
Exercise is a powerful depression fighter for several reasons. Most  
Exercise is not just about aerobic capacity and muscle size. Sure,
```

People who exercise regularly tend to do so because it gives them a  
Regular exercise can have a profoundly positive impact on depression  
Ever noticed how your body feels when you're under stress? Your muscles  
Exercising is an effective way to break this cycle. As well as releasing  
Instead of allowing your mind to wander, pay close attention to the  
Outdoor activities like hiking, sailing, mountain biking, rock climbing

Desired Output :

Finally, exercise can also serve as a distraction, allowing you to

▼ Show Solution

```
import sumy
from sumy.summarizers.luhn import LuhnSummarizer

#Plain text parsers since we are parsing through text
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer

parser=PlaintextParser.from_string(original_text,Tokenizer("english"))

summarizer=LuhnSummarizer()
```

```
my_summary=summarizer(parser.document,2)
print(my_summary)
```

```
#> (<Sentence: Finally, exercise can also serve as a distraction,
```

## 51. How to summarize text based on LSA algorithm ?

Difficulty Level : L3

Q. Extract the summary of the given text based on the LSA Algorithm.

Input :

```
original_text="""Studies show that exercise can treat mild to moderate depression. Exercise is a powerful depression fighter for several reasons. Most people who exercise regularly feel better. Exercise is not just about aerobic capacity and muscle size. Sure,
```



People who exercise regularly tend to do so because it gives them a  
Regular exercise can have a profoundly positive impact on depression  
Ever noticed how your body feels when you're under stress? Your muscles  
Exercising is an effective way to break this cycle. As well as releasing  
Instead of allowing your mind to wander, pay close attention to the  
Outdoor activities like hiking, sailing, mountain biking, rock climbing

Desired Output :

In addition to relieving depression symptoms, research also shows that

▼ Show Solution

```
import sumy
from sumy.summarizers.lsa import LsaSummarizer

#Plain text parsers since we are parsing through text
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer

parser=PlaintextParser.from_string(original_text,Tokenizer("english"))

summarizer=LsaSummarizer()
```

```
my_summary=summarizer(parser.document,2)
print(my_summary)
```

```
#> (<Sentence: In addition to relieving depression symptoms, rese
```

## 52. How to convert documents into json format ?

Difficulty Level : L3

Q. Covert the given text documents into json format for spacy usage

Input:

```
text1="Netflix has released a new series"
text2="It was shot in London"
```

```
text3="It is called Dark and the main character is Jonas"
text4="Adam is the evil character"
```

Desired Output :

```
{'id': 0,
 'paragraphs': [{'cats': [],
  'raw': 'Netflix has released a new series',
  'sentences': [{'brackets': [],
   'tokens': [{'dep': 'nsubj',
    'head': 2,
    'id': 0,
    'ner': 'U-ORG',
    'orth': 'Netflix',
    'tag': 'NNP'}],
   {'dep': 'aux',
    'head': 1,
    'id': 1,
    'ner': 'O',
    'orth': 'has',
    'tag': 'VBZ'}],
   {'dep': 'ROOT',
    'head': 0,
    'id': 2,
    'ner': 'O',
    'orth': 'released',
```

```
'tag': 'VBN'},
{'dep': 'det', 'head': 2, 'id': 3, 'ner': 'O', 'orth': 'a', '
{'dep': 'amod',
  'head': 1,
  'id': 4,
  'ner': 'O',
  'orth': 'new',
  'tag': 'JJ'},
{'dep': 'dobj',
  'head': -3,
  'id': 5,
  'ner': 'O',
  'orth': 'series',
  'tag': 'NN'}}]]]],
...(truncated)
```

▼ Show Solution

```
# Covert into spacy documents
doc1=nlp(text1)
doc2=nlp(text2)
doc3=nlp(text3)
doc4=nlp(text4)

# Import docs_to_json
from spacy.gold import docs_to_json
```

```
# Converting into json format
json_data = docs_to_json([doc1, doc2, doc3, doc4])
json_data
```

```
{'id': 0,
 'paragraphs': [{'cats': [],
 'raw': 'Netflix has released a new series',
 'sentences': [{'brackets': [],
 'tokens': [{'dep': 'nsubj',
 'head': 2,
 'id': 0,
 'ner': 'U-ORG',
 'orth': 'Netflix',
 'tag': 'NNP'}],
 {'dep': 'aux',
 'head': 1,
 'id': 1,
 'ner': 'O',
 'orth': 'has',
 'tag': 'VBZ'}],
 {'dep': 'ROOT',
 'head': 0,
 'id': 2,
 'ner': 'O',
 'orth': 'released',
 'tag': 'VBN'}],
```

```
{'dep': 'det', 'head': 2, 'id': 3, 'ner': 'O', 'orth': 'a',
{'dep': 'amod',
  'head': 1,
  'id': 4,
  'ner': 'O',
  'orth': 'new',
  'tag': 'JJ'}},
{'dep': 'dobj',
  'head': -3,
  'id': 5,
  'ner': 'O',
  'orth': 'series',
  'tag': 'NN'}}]]]],
{'cats': [],
'raw': 'It was shot in London',
'sentences': [{'brackets': [],
'tokens': [{'dep': 'nsubjpass',
  'head': 2,
  'id': 0,
  'ner': 'O',
  'orth': 'It',
  'tag': 'PRP'}},
{'dep': 'auxpass',
  'head': 1,
  'id': 1,
  'ner': 'O',
  'orth': 'was',
  'tag': 'VBD'}},
```

```
{'dep': 'ROOT',  
  'head': 0,  
  'id': 2,  
  'ner': 'O',  
  'orth': 'shot',  
  'tag': 'VBN'},  
{'dep': 'prep',  
  'head': -1,  
  'id': 3,  
  'ner': 'O',  
  'orth': 'in',  
  'tag': 'IN'},  
{'dep': 'pobj',  
  'head': -1,  
  'id': 4,  
  'ner': 'U-GPE',  
  'orth': 'London',  
  'tag': 'NNP'}}]]]],  
...(truncated)
```

## 53. How to build a text classifier with TextBlob ?

Difficulty Level : L3

Q Build a text classifier with available train data using textblob library

Input:

```
# Data to train the classifier
train = [
    ('I love eating sushi','food-review'),
    ('This is an amazing place!', 'Tourist-review'),
    ('Pizza is my all time favorite food','food-review'),
    ('I baked a cake yesterday, it was tasty', 'food-review'),
    ("What an awesome taste this sushi has", 'food-review'),
    ('It is a perfect place for outing', 'Tourist-review'),
    ('This is a nice picnic spot', 'Tourist-review'),
    ("Families come out on tours here", 'Tourist-review'),
    ('It is a beautiful place !', 'Tourist-review'),
    ('The place was warm and nice', 'Tourist-review')
]
test = [
    ('The sushi was good', 'food-review'),
    ('The place was perfect for picnics ', 'Tourist-review'),
    ("Burgers are my favorite food", 'food-review'),
    ("I feel amazing!", 'food-review'),
    ('It is an amazing place', 'Tourist-review'),
```



```
("This isn't a very good place", 'Tourist-review')  
]
```

Desired Output :

```
Accuracy: 0.8333333333333334
```

▼ Show Solution

```
# Importing the classifier  
from textblob.classifiers import NaiveBayesClassifier  
from textblob import TextBlob  
  
# Training  
cl = NaiveBayesClassifier(train)  
  
# Classify some text  
print(cl.classify("My favorite food is spring rolls"))  
print(cl.classify("It was a cold place for picnic"))  
  
# Printing accuracy of classifier  
print("Accuracy: {0}".format(cl.accuracy(test)))  
  
#> food-review  
#> Tourist-review
```

```
#> Tourist-review  
#> Accuracy: 0.8333333333333334
```

## 54. How to train a text classifier using Simple transformers ?

Difficulty Level : L4

Q. Build and train a text classifier for the given data using `simpletransformers` library

Input :

```
train_data = [  
    ["The movie was amazing", 1],  
    ["It was a boring movie", 0],  
    ["I had a great experience",1],  
    ["I was bored during the movie",0],  
    ["The movie was great",1],  
    ["The movie was bad",0],
```

```
["The movie was good",1]  
]
```

▼ Show Solution

```
# Import requirements  
  
!pip install simpletransformers  
from simpletransformers.classification import ClassificationModel  
import pandas as pd  
import logging  
  
logging.basicConfig(level=logging.INFO)  
transformers_logger = logging.getLogger("transformers")  
transformers_logger.setLevel(logging.WARNING)  
  
# Preparing train data  
  
train_df = pd.DataFrame(train_data)  
train_df.columns = ["text", "labels"]  
  
# Optional model configuration  
model_args = ClassificationArgs(num_train_epochs=5)  
  
# Create a ClassificationModel  
model = ClassificationModel("bert", "bert-base-uncased", args=model_args)
```

```
# Train the model
model.train_model(train_df)

# Make predictions with the model
predictions, raw_outputs = model.predict(["The titanic was a good

predictions
#> array([1])
```

## 55. How to perform text classification using spaCy ?

Difficulty Level : L4

Q. Build a text classifier using spacy that can classify IMDB reviews as positive or negative

▼ Show Solution

```
import spacy
nlp=spacy.load("en_core_web_sm")

textcat = nlp.create_pipe("textcat", config={"exclusive_classes":
nlp.add_pipe(textcat, last=True)
textcat = nlp.get_pipe("textcat")

# add label to text classifier
textcat.add_label("POSITIVE")
textcat.add_label("NEGATIVE")

def load_data(limit=0, split=0.8):
    """Load data from the IMDB dataset."""
    # Partition off part of the train data for evaluation
    train_data, _ = thinc.extra.datasets.imdb()
    random.shuffle(train_data)
    train_data = train_data[-limit:]
    texts, labels = zip(*train_data)
    cats = [{"POSITIVE": bool(y), "NEGATIVE": not bool(y)} for y
    split = int(len(train_data) * split)
    return (texts[:split], cats[:split]), (texts[split:], cats[sp

# load the IMDB dataset
print("Loading IMDB data...")
```

```

(train_texts, train_cats), (dev_texts, dev_cats) = load_data()
train_texts = train_texts[:n_texts]
train_cats = train_cats[:n_texts]

train_data = list(zip(train_texts, [{"cats": cats} for cats in tr

# get names of other pipes to disable them during training
pipe_exceptions = ["textcat", "trf_wordpiecer", "trf_tok2vec"]
other_pipes = [pipe for pipe in nlp.pipe_names if pipe not in pip

# Training the text classifier
with nlp.disable_pipes(*other_pipes): # only train textcat
    optimizer = nlp.begin_training()
    if init_tok2vec is not None:
        with init_tok2vec.open("rb") as file_:
            textcat.model.tok2vec.from_bytes(file_.read())
            print("Training the model...")
            print("{:^5}\t{:^5}\t{:^5}\t{:^5}".format("LOSS", "P", "R
            batch_sizes = compounding(4.0, 32.0, 1.001)
            for i in range(n_iter):
                losses = {}
                # batch up the examples using spaCy's minibatch
                random.shuffle(train_data)
                batches = minibatch(train_data, size=batch_sizes)
                for batch in batches:
                    texts, annotations = zip(*batch)
                    nlp.update(texts, annotations, sg=optimizer, dro

```



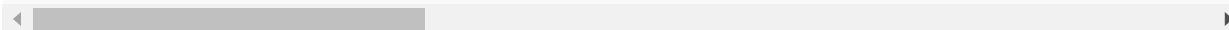
## 56. How to translate the text (using simpletransformers) ?

Difficulty Level : L3

Q. Translate the given list of texts from English to Dutch using simpletransformers package

Input :

```
['Our experienced writers travel the world to bring you informative  
Each part of Germany is different, and there are  
Christmas Markets originated in Germany, and the  
Garmisch-Partenkirchen is a small town in Bavaria  
It's one of the country's top alpine destination  
In spring, take a road trip through Bavaria and
```



Desired Output :

['Unsere erfahrenen Autoren reisen die Welt, um Ihnen informative und inspirierende Funktionen, Destination Rund', 'Jeder Teil Deutschlands ist anders, und es gibt Tausende von denkwürdigen Orten zu besuchen.', 'Weihnachtsmärkte entstanden in Deutschland, und die Tradition stammt aus dem späten Mittelalter.', 'Garmisch-Partenkirchen ist eine kleine Stadt in Bayern, nahe Deutschland.Die Zug', 'Es ist eines der Top-Alpenziele des Landes, sehr beliebt im Winter', 'Im Frühjahr machen Sie eine Roadtrip durch Bayern und genießen den Blick auf die dunkelgrünen Alpen']

▼ Show Solution

```
# Install the package
!pip install simpletransformers

# Import the model
from simpletransformers.seq2seq import Seq2SeqModel

# Setting desired arguments
my_args = {    "train_batch_size": 2,
               "num_train_epochs": 10,
               "save_eval_checkpoints": False,
               "save_model_every_epoch": False,
               "evaluate_during_training": True,
```



```
"evaluate_generated_text": True    }
```

```
# Instantiating the model
```

```
my_model=Seq2SeqModel(encoder_decoder_name="Helsinki-NLP/opus-mt-
```

```
# translating the text
```

```
my_model.predict(['Our experienced writers travel the world to br  
    'Each part of Germany is different, and there a  
    "Christmas Markets originated in Germany, and t  
    "Garmisch-Partenkirchen is a small town in Bava  
    "It's one of the country's top alpine destinati  
    "In spring, take a road trip through Bavaria an
```

```
#> ['Unsere erfahrenen Autoren reisen die Welt, um Ihnen informat  
#> 'Jeder Teil Deutschlands ist anders, und es gibt Tausende von  
#> 'Weihnachtsmärkte entstanden in Deutschland, und die Tradition  
#> 'Garmisch-Partenkirchen ist eine kleine Stadt in Bayern, nahe  
#> 'Es ist eines der Top-Alpenziele des Landes, sehr beliebt im W  
#> 'Im Frühjahr machen Sie eine Roadtrip durch Bayern und genieße
```

## 57. How to create a Question-Answering system from given context

Difficulty Level : L4

Q. Build a Question Answering model that answers questions from the given context using transformers package

Input :

```
context=""" Harry Potter is the best book series according to many  
It is afantasy based novel that provides a thrilling experience to  
  
question="What is Harry Potter ?"
```

Desired Output :

```
{'score': 0.2375375191101107, 'start': 17, 'end': 37, 'answer': 'tl
```

✓ Show Solution

```
#Install and import the pipeline of transformers
!pip install transformers
from transformers import pipeline

# Get the task-specific pipeline
my_model=pipeline(task="question-answering")

context = r""" Harry Potter is the best book series according to
It is a fantasy based novel that provides a thrilling experience t

# Pass the question and context to the model to obtain answer
print(my_model(question="What is Harry Potter ?", context=context
print(my_model(question="Who wrote Harry Potter ?", context=conte

#> {'score': 0.2375375191101107, 'start': 17, 'end': 37, 'answer'
#> {'score': 0.9813234768798256, 'start': 92, 'end': 102, 'answer'
```

## 58. How to do text generation starting from a given piece of text?

Difficulty Level : L4

Q. Generate text based on the the starting provided.

Input :

```
starting="It was a bright"
```

Desired Output :

```
'It was a bright day in New Jersey\'s capitol," the senator told a
```

▼ Show Solution

```
# Import pipeline from transformers package
from transformers import pipeline

# Get the task-specific pipeline
my_model=pipeline(task="text-generation")

# Pass the starting sequence as input to generate text
my_model(starting)

#> [{'generated_text': 'It was a bright day in New Jersey\'s capi
```

## 59. How to classify a text as positive or negative sentiment with transformers?

Difficulty Level : L4

Q. Find out whether a given text is positive or negative sentiment along with score for predictions

Input text:

```
text1="It is a pleasant day, I am going for a walk"  
text2="I have a terrible headache"
```

Desired Output :

```
[{'label': 'POSITIVE', 'score': 0.9998570084571838}]  
[{'label': 'NEGATIVE', 'score': 0.9994378089904785}]
```

✓ Show Solution

```
# Import pipeline from transformers package
from transformers import pipeline

# Get the task specific pipeline
my_model = pipeline("sentiment-analysis")

# Predicting the sentiment with score
print(my_model(text1))
print(my_model(text2))

[{'label': 'POSITIVE', 'score': 0.9998570084571838}]
[{'label': 'NEGATIVE', 'score': 0.9994378089904785}]
```

I hope you found this useful. For more such posts, stay tuned to our page !



[report this ad](#)

---

**Copyright** Machine Learning Plus. **All rights reserved.**

[Home](#) [Contact Us](#) [Privacy Policy](#) [About Selva](#) [Terms and Conditions](#)