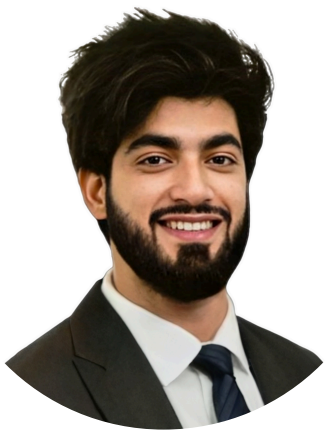

Tajamul Khan

Cheat Sheet

@Tajamulkhann

NLP Basics

Tokenization – Splitting text into words/sentences
Stopword Removal – Removing common words (e.g., "the", "is")
Stemming & Lemmatization – Reducing words to root form
TF-IDF – Measuring word importance in a document
Word Embeddings – Representing words as dense vectors

@Tajamulkhann

Text Preprocessing

Lowercasing – Convert text to lowercase
Removing Special Characters – Remove punctuation & numbers
Tokenization – Splitting text into words/sentences
Removing Stopwords – Removing commonly used words
Lemmatization – Converting words to their base form

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

text = "Natural Language Processing is amazing!"
tokens = word_tokenize(text.lower())  # Lowercasing & Tokenization
tokens = [word for word in tokens if word.isalnum()]  # Remove special
characters
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word not in stop_words]  # Remove
stopwords
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]
# Lemmatization
```

@Tajamulkhann

Feature Extraction

Bag-of-Words (BoW) – Convert text into numerical features
TF-IDF (Term Frequency-Inverse Document Frequency) – Weight words based on importance
Word2Vec & GloVe – Convert words into dense vector representations

```
from sklearn.feature_extraction.text import
CountVectorizer, TfidfVectorizer

corpus = ["NLP is amazing", "Text processing is
fun"]
vectorizer = CountVectorizer()
X_bow = vectorizer.fit_transform(corpus)
# Bag-of-Words

tfidf = TfidfVectorizer()
X_tfidf = tfidf.fit_transform(corpus)  # TF-IDF
```

@Tajamulkhann

Sentiment Analysis

Rule-Based Sentiment Analysis – Using lexicons
Machine Learning - Based Sentiment Analysis

```
from textblob import TextBlob

text = "I love learning NLP!"
sentiment =
TextBlob(text).sentiment.polari
ty  # -1 (negative) to +1
(positive)
```

@Tajamulkhann

Named Entity Recognition (NER)

Identifies entities like names, places, organizations, dates

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

text = "Elon Musk founded SpaceX in
2002."
doc = nlp(text)
for ent in doc.ents:
    print(ent.text, ent.label_)
```

@Tajamulkhann

Text Classification

Classify text into categories (e.g., spam detection, sentiment analysis)

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

text_clf = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('clf', MultinomialNB())  # Naive Bayes for
text classification
])

text_clf.fit(X_train, y_train)
y_pred = text_clf.predict(X_test)
```

@Tajamulkhann

Topic Modeling (LDA)

Extracts topics from a large collection of documents

```
from sklearn.decomposition import
LatentDirichletAllocation

lda =
LatentDirichletAllocation(n_components=5,
random_state=42)
topics = lda.fit_transform(X_tfidf)  #
Applying LDA on TF-IDF features
```

@Tajamulkhann

Word Embeddings

Word2Vec – Learns vector representation of words
GloVe – Captures meaning based on word co-occurrence

```
from gensim.models import Word2Vec

sentences = [["NLP", "is", "amazing"],
["Text", "processing", "is", "fun"]]
model = Word2Vec(sentences,
vector_size=100, window=5, min_count=1,
workers=4)
word_vector = model.wv['NLP']  # Get
word vector for 'NLP'
```

@Tajamulkhann

Text Generation with GPT

Generate human-like text using GPT models

```
from transformers import pipeline

generator = pipeline("text-generation", model="gpt2")
generated_text = generator("The future of AI is", max_length=50)
print(generated_text)
```

@Tajamulkhann

Speech-to-Text & Text-to-Speech

```
import speech_recognition as sr
import pyttsx3

# Speech to Text
r = sr.Recognizer()
with sr.Microphone() as source:
    print("Speak now...")
    audio = r.listen(source)
    text = r.recognize_google(audio)
    print("You said:", text)

# Text to Speech
engine = pyttsx3.init()
engine.say("Hello, how are you?")
engine.runAndWait()
```

@Tajamulkhann

Found Helpful ?

Repost

Follow for more!