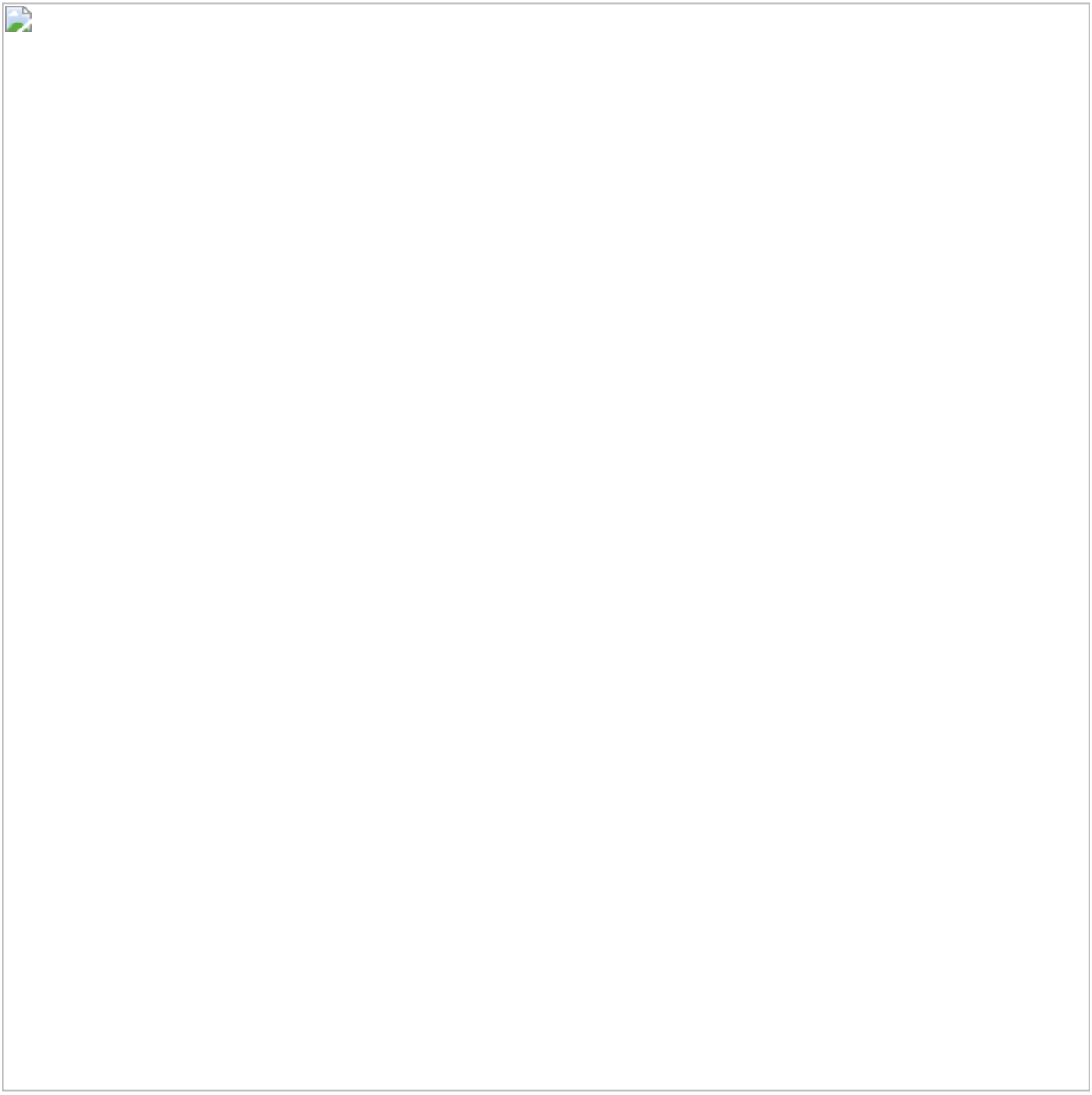
# Every Beginner NLP Engineer must know these Techniques



Self made image

## Tokenization:

Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements, known as tokens.

Here is an example of tokenization in Python using the NLTK library:

```
import nltk
from nltk.tokenize import word_tokenize
```

```
text = tokens = word_tokenize(text)(tokens)

nltk.download()tokenizer = nltk.data.load()
```

## Lemmatization:

Lemmatization is the process of reducing a word to its base or root form, called a lemma. Stemming is a similar process, but it often results in words that are not actual words.

Here is an example of lemmatization in Python using the NLTK library:

```
import nltk
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

(lemmatizer.lemmatize())(lemmatizer.lemmatize())
```

## Steaming:

In Natural Language Processing (NLP), “steaming” refers to the process of reducing a word to its base or root form. This is often done to group together different forms of a word so they can be analyzed together as a single item.

Here is an example of stemming in python using NLTK library

```
import nltk
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()(stemmer.stem())(stemmer.stem())
```

## Part-of-Speech Tagging:

Part-of-speech (POS) tagging is the process of marking each word in a text with its corresponding POS tag. Here is an example of POS tagging in Python using the NLTK library:

```
 nltkfrom nltk  pos_tagfrom nltk.tokenize  word_tokenizetext = tokens =
word_tokenize(text)pos_tags = pos_tag(tokens)(pos_tags)# Output: [(, ), (, ), (,
), (, ), (, ), (, ), (, ), (, )]
```

## Named Entity Recognition:

Named Entity Recognition (NER) is the process of identifying and classifying named entities in a text into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc. Here is an example of NER in python using NLTK

```
 nltk nltk  ne_chunk nltk.tokenize  word_tokenizetext = tokens =
word_tokenize(text)tagged_tokens = nltk.pos_tag(tokens)ner_tree =
ne_chunk(tagged_tokens)(ner_tree)
```

## Sentiment Analysis:

Sentiment Analysis is the process of determining the emotional tone behind a piece of text, whether it is positive, negative, or neutral. Here is an example of Sentiment Analysis in Python using the NLTK library:

```
 nltkfrom nltk.sentiment  SentimentIntensityAnalyzertext = sia = SentimentIntensityAnalyzer()score = sia.polarity_scores(text)(score)# Output: {: , : , : , : }
```

## Text Classification:

Text Classification is the process of assigning predefined categories or tags to a piece of text. Here is an example of Text Classification in Python using the scikit-learn library:

```
 pandas  pd sklearn.feature_extraction.text  CountVectorizer sklearn.naive_bayes MultinomialNBdata = {: [, ], : [, ]}df = pd.DataFrame(data)vectorizer = CountVectorizer()X = vectorizer.fit_transform(df[])clf = MultinomialNB()clf.fit(X, df[])text = X_test = vectorizer.transform([text])pred = clf.predict(X_test)(pred)
```

## Language Translation:

Language Translation is the process of converting text from one language to another.

Here is an example of Language Translation in Python using the googletrans library:

```
 googletrans  Translatortranslator = Translator()text = translated_text = translator.translate(text, dest=).text(translated_text)
```

## Text Summarization:

Text summarization is the process of condensing a piece of text to its main points.

Here is an example of Text Summarization in Python using the gensim library:

```
 gensim.summarization  summarizetext = summary = summarize(text)(summary)
```

## Word Embeddings (e.g. Word2Vec, GloVe):

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation.

Here is an example of training a Word2Vec model in Python using the gensim library:

```
 gensim.models  Word2Vecsentences = [[, , , , ],            [, , , , ], [, , , , ]]model = Word2Vec(sentences, size=, window=, min_count=, workers=)word_vector = model.wv[](word_vector)
```

Here is an example of loading pre-trained GloVe model in Python using the gensim library:

```
 gensim.models  KeyedVectorsmodel = KeyedVectors.load_word2vec_format(, binary=)word_vector = model[](word_vector)
```

## Dependency Parsing:

Dependency parsing is the process of analyzing the grammatical structure of a sentence, based on the dependencies between the words in the sentence.

Here is an example of Dependency Parsing in Python using the spaCy library:

```
import spacynlp = spacy.load()sentence = doc = nlp(sentence)for token in doc: print(token.text, token.dep_)
```

> Note: while the above example uses spaCy library, there are other libraries such as NLTK, Stanford Parser, etc that can be used for Dependency parsing.

## Topic modeling

Topic modeling is a method used in natural language processing (NLP) to identify patterns and topics in a text corpus. One popular technique for topic modeling is Latent Dirichlet Allocation (LDA), which uses a statistical model to discover latent topics in a set of documents.

Here is an example of how to perform topic modeling using LDA and the gensim library in Python:

```
 gensim.corpora  Dictionary gensim.models  LdaModeltexts = [[, , , ],         [, , , ],         [, , ]]dictionary = Dictionary(texts)corpus = [dictionary.doc2bow(text)  text  texts]lda = LdaModel(corpus, num_topics=, id2word=dictionary) topic_id, topic  lda.print_topics():    (, topic_id+) (topic)
```

This example uses a simple text corpus containing three documents and trains an LDA model with 2 topics. The output will show the two topics learned by the model and the words that are associated with each topic.

## Term frequency

Term frequency(tf) is a measure of how often a term appears in a document. It is commonly used in information retrieval and text mining. The tf-idf (term frequency-inverse document frequency) is a weighting scheme that assigns a weight to each term in a document based on its tf and idf.

Here is an example of how to calculate the term frequency of a document using python:

```
from collections import Counterdocument = tokens = document.split()tf = Counter(tokens)print(tf)
```

This example will show the frequency of each word in the document in the form of a dictionary.

## Mastering these techniques will give you a solid foundation for more advanced NLP tasks, such as language translation, text summarization, and question answering.

Follow given blog link to master in advance NLP techniques https://ankushmulkar.medium.com/top-most-ten-nlp-techniques-used-in-the-industry-34570a29f2f

To know more about Advance NLP, follow below link.

AnkushMulkar/Natural-Language-processing (github.com)

Ankush Mulkar Github portfolio