NLP : technology used by machine to understand, analyze, manipulate, and interpret Human's language

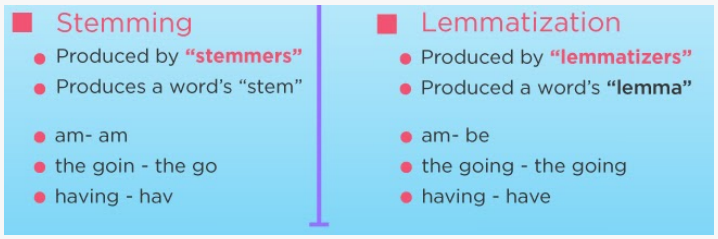
NLG : helps machine to understand & Analyze human language

Text Mining : Analyze and extract patterns from the text data.

NLU : it acts as a translator that converts computerized data into human language.

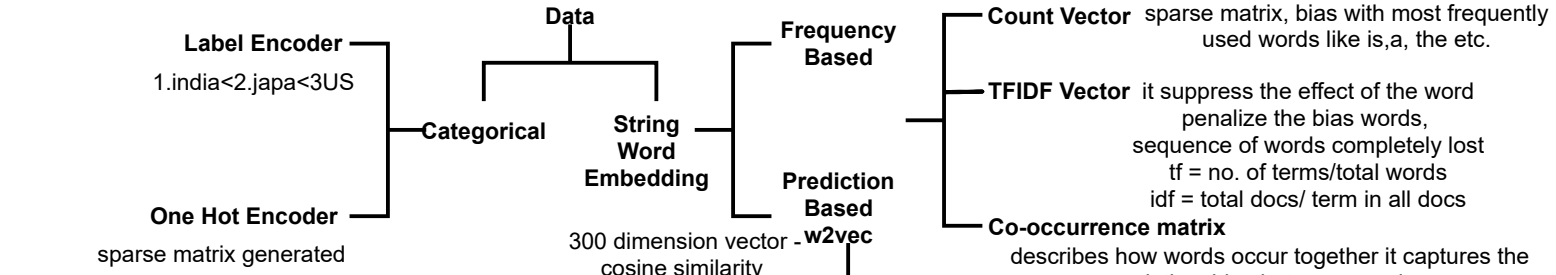
NLP Concepts:

```python
import spacy
nlp = spacy.load('en_core_web_sm')
doc = nlp(u"It cost $5 million and Companies Email is ansarijuned4all@gmail.com and i live in india, right now I am eating mango")
print("Sentence Token:", [sentense for sentense in doc.sents])
# Sentence Token: [It cost $5 million and Companies Email is ansarijuned4all@gmail.com, and i live in india, , right now I am eating mango]
print("Token:", [token.text for token in doc])
# Token: ['It', 'cost', '$', '5', 'million', 'and', 'Companies', 'Email', 'is', 'ansarijuned4all@gmail.com', 'and', 'i', 'live', 'in', 'india', ',', 'right', 'now', 'I', 'am', 'eating', 'mango']
print("Token Index:", [token.i for token in doc])
# Token Index: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
print("\n")
print("Named Entity Recongnisation(NER):")
for token in doc.ents:
  print(token.text,":::Label(N E R):::",token.label_,"::: Explain:::",spacy.explain(token.label_))
"""
Named Entity Recongnisation(NER):
$5 million :::Label(N E R):::: MONEY ::: Explain::: Monetary values, including unit
Companies Email :::Label(N E R):::: ORG ::: Explain::: Companies, agencies, institutions, etc.
ansarijuned4all@gmail.com :::Label(N E R):::: ORG ::: Explain::: Companies, agencies, institutions, etc.
india :::Label(N E R):::: GPE ::: Explain::: Countries, cities, states
"""
print("\n")
print("POS_:",[token.pos_ for token in doc])
# POS_: ['PRON', 'VERB', 'SYM', 'NUM', 'NUM', 'CCONJ', 'PROPN', 'PROPN', 'AUX', 'X', 'CCONJ', 'PRON', 'VERB', 'ADP', 'PROPN', 'PUNCT', 'ADV', 'ADV', 'PRON', 'AUX', 'VERB', 'NOUN']
print("tag_:",[token.tag_ for token in doc])
# tag_: ['PRP', 'VBD', '$', 'CD', 'CD', 'CC', 'NNPS', 'NNP', 'VBZ', 'ADD', 'CC', 'PRP', 'VBP', 'IN', 'NNP', ',', 'RB', 'RB', 'PRP', 'VBP', 'VBG', 'NN']
print("tag_:",[spacy.explain(token.tag_) for token in doc])
# tag_: ['pronoun, personal', 'verb, past tense', 'symbol, currency', 'cardinal number', 'cardinal number', 'conjunction, coordinating', 'noun, proper plural', 'noun, proper singular', 'verb, 3rd per
print("lemma_:",[token.lemma_ for token in doc])
# lemma_: ['-PRON-', 'cost', '$', '5', 'million', 'and', 'Companies', 'Email', 'be', 'ansarijuned4all@gmail.com', 'and', 'i', 'live', 'in', 'india', ',', 'right', 'now', '-PRON-', 'be', 'eat', 'mango
print("is_stop:",[token.is_stop for token in doc])
# is_stop: [True, False, False, False, False, True, False, False, True, False, True, True, False, True, False, False, False, True, True, True, False, False]
print("is_alpha:", [token.is_alpha for token in doc])
# is_alpha: [True, True, False, False, True, True, True, True, True, False, True, True, True, True, True, False, True, True, True, True, True, True]
print("is_punct:", [token.is_punct for token in doc])
# is_punct: [False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, True, False, False, False, False, False, False]
print("like_num:", [token.like_num for token in doc])
# like_num: [False, False, False, True, True, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False]
print("like_email:", [token.like_email for token in doc])
# like_email: [False, False, False, False, False, False, False, False, False, True, False, False, False, False, False, False, False, False, False, False, False, False, False]
print("doc.ents Length:",len(doc.ents))
#doc.ents Length: 4
```



Feature Extraction:

Corpus: it is a collection of documents **Vocabulary:** collection of unique words present in corpus **BOW:** total unique words in all the documents




```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
cv = CountVectorizer(stop_words='english')
tv = TfidfVectorizer(stop_words='english')

messages=['Cricket Game teach us teamwork','Chess Game teach
cv_transformed_messages = cv.fit_transform(messages)
```

```python
from gensim.models import Word2Vec
mymodel_cbow = Word2Vec(tokenized_sentences, min_count=1, size = 100)
mymodel_skipgram = Word2Vec(tokenized_sentences, min_count=1, size = 100,sg=1)
mymodel_cbow.similarity("python","Java")
mymodel_cbow.most_similar("Data")
```

```python
from textblob import TextBlob
text = "this is a good movie ever seen"
textblob(text).sentiment
```

NLP Libraries

Scikit-learn: It provides a wide range of algorithms for building machine learning models in Python.
Natural language Toolkit (NLTK): NLTK is a complete toolkit for all NLP techniques.
Pattern: It is a web mining module for NLP and machine learning.
TextBlob: It provides an easy interface to learn basic NLP tasks like sentiment analysis, noun phrase extraction, or pos-tagging, spelling correction.
Quepy: Quepy is used to transform natural language questions into queries in a database query language.

```
tqa = pipeline(task="table-question-answering", model="google/tapas-base-finetuned-wtq")
```

SpaCy: SpaCy is an open-source NLP library which is used for Data Extraction, Data Analysis, Sentiment Analysis, and Text Summarization.
Gensim: Gensim works with large datasets and processes data streams.
GPT1,2
GPT-Neo: clone of gpt-3 but let dimension.
T5(Text to Text Transfer Transformer): Context + Answer → Generate Question

- AutomaticSpeechRecognitionPipeline
- ConversationalPipeline
- FeatureExtractionPipeline
- FillMaskPipeline
- ImageClassificationPipeline
- QuestionAnsweringPipeline
- SummarizationPipeline
- TextClassificationPipeline
- TextGenerationPipeline
- TokenClassificationPipeline
- TranslationPipeline
- ZeroShotClassificationPipeline
- Text2TextGenerationPipeline
- TableQuestionAnsweringPipeline

```python
from transformers import pipeline
```

```
!pip install autoviml
from autoviml.Auto_NLP import Auto_NLP
nlp_column = 'reviews'
target = 'target'
train_nlp, test_nlp, nlp_transformer, preds = Auto_NLP(
                nlp_column, train, test, target, score_type='balanced_accuracy',
                modeltype='Classification',top_num_features=50, verbose=2,
                build_model=True)
nlp_transformer.predict(test[nlp_column])
```