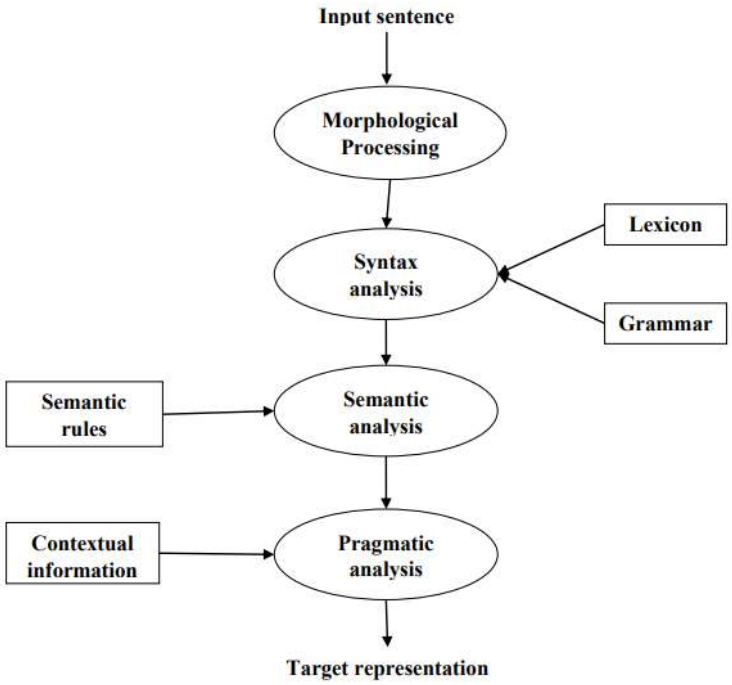
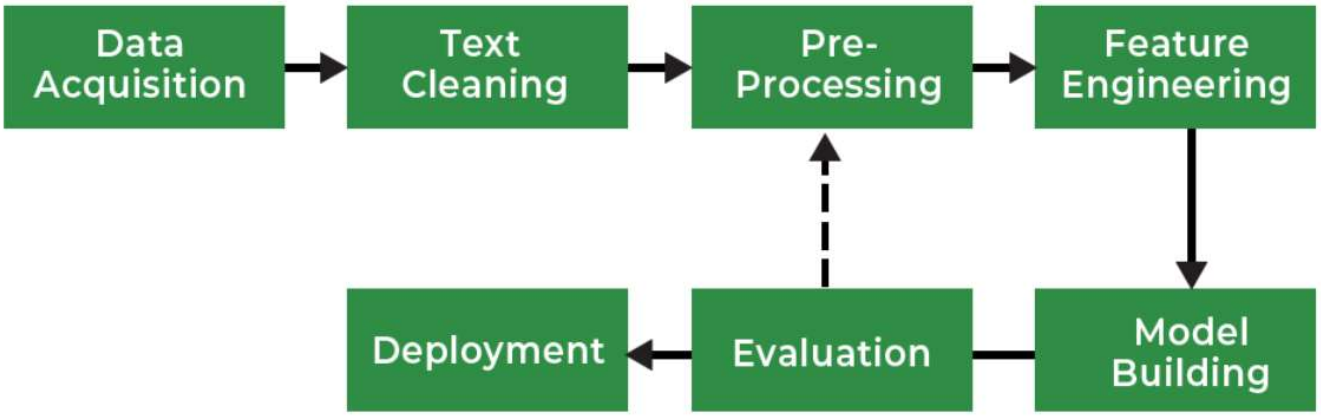
NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) is a machine learning technology that gives computers the ability to interpret, manipulate, and comprehend human language. Organizations today have large volumes of voice and text data from various communication channels like emails, text messages, social media newsfeeds, video, audio, and more. They use NLP software to automatically process this data, analyze the intent or sentiment in the message, and respond in real time to human communication.

Phases of NLP :



NLP Pipeline :



Stemming :

Stemming is the process of reducing a word to its root or stem form. For example, the stem of the word "running" is "run". Stemming algorithms are based on rules and heuristics that remove affixes and suffixes to generate the stem of the word. Stemming is a relatively simple and fast process that can be applied to large amounts of text data.

In [2]:
```
import nltk
from nltk.stem import PorterStemmer

text = "This is a sample text for stemming."

# Tokenize the text
tokens = nltk.word_tokenize(text)

# Stem the tokens
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(token) for token in tokens]

print(stemmed_tokens)
```

```
['thi', 'is', 'a', 'sampl', 'text', 'for', 'stem', '.']
```

Lemmatization:

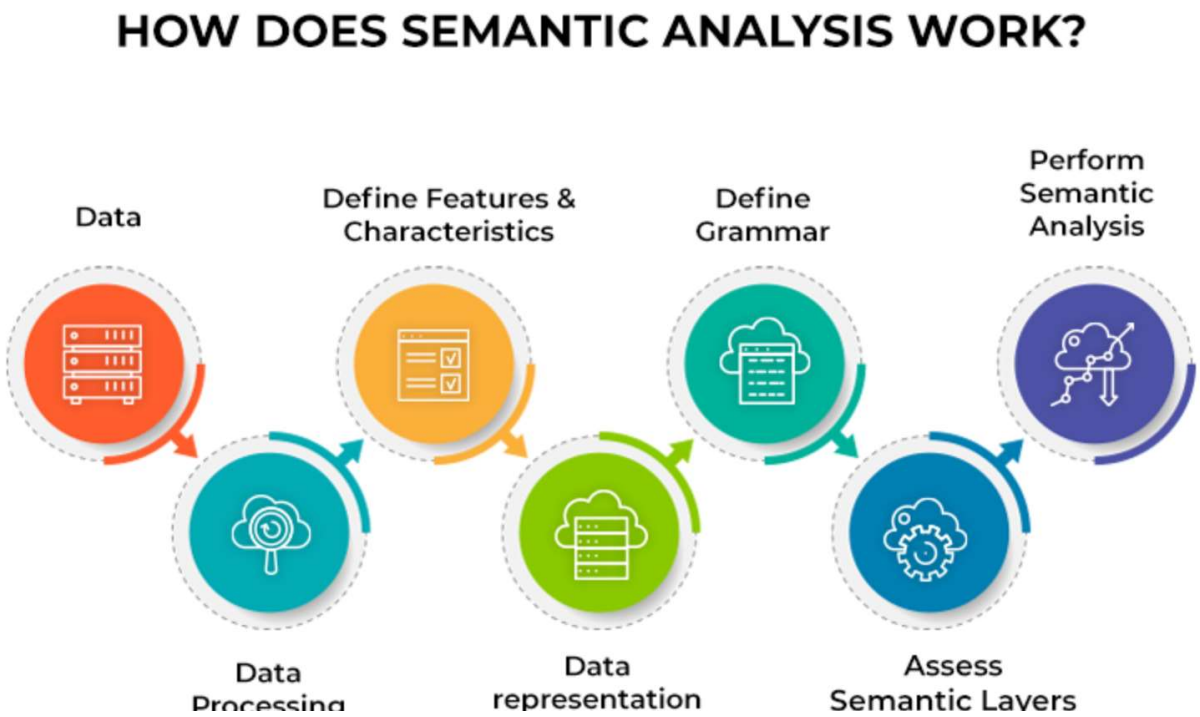
Lemmatization is the process of reducing a word to its base or dictionary form, which is called the lemma. Unlike stemming, lemmatization takes into consideration the context of the word and its grammar, which makes it more accurate. Lemmatization requires access to a dictionary or knowledge base that maps words to their lemma forms.

In [7]:
```
import nltk
from nltk.stem import WordNetLemmatizer

text = "Hello i am Data Scientist and i love Reading books."

# Tokenize the text
tokens = nltk.word_tokenize(text)

# Lemmatize the tokens
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]

print(lemmatized_tokens)
```

```
['Hello', 'i', 'am', 'Data', 'Scientist', 'and', 'i', 'love', 'Reading', 'book', '.']
```

Semantic Analysis :

Semantic analysis refers to a process of understanding natural language (text) by extracting insightful information such as context, emotions, and sentiments from unstructured data. It gives computers and systems the ability to understand, interpret, and derive meanings from sentences, paragraphs, reports, registers, files, or any document of a similar kind.



Regular Expressions :

A regular expression (RE) is a language for specifying text search strings. RE helps us to match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are used to search texts in UNIX as well as in MS WORD in identical way. We have various search engines using a number of RE features.

\w	Matches any alphanumeric character including letters and digits
\d	Matches any digit
\s	Matches any whitespace character
\b	Matches a word boundary

Dot (.)	Matches any single character except a newline character
Asterisk (*)	Matches zero or more instances of the preceding character
Plus (+)	Matches one or more instances of the preceding character.
Question mark (?)	Matches zero or one instance of the preceding character
Caret (^)	Matches the start of a line
Dollar sign ($)	Matches the end of a line

Working on NLP using Semantic Analysis :

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv('Restaurant_Reviews.tsv', delimiter='\t', quoting=3)
df.head()
```

```
                                                Review  Liked
0                            Wow... Loved this place.      1
1                                  Crust is not good.      0
2          Not tasty and the texture was just nasty.      0
3  Stopped by during the late May bank holiday of...      1
4  The selection on the menu was great and so wer...      1
```

Stemming: Chops off the ends of words based on heuristis like 'SS' -> 'S' Lemmatization: Respects words and reduces to morphological root

```
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()

def rep(review):
    review = re.sub('[^a-zA-Z]', ' ', review)
    review = review.lower()
    reviews = review.split()
    reviews = [stemmer.stem(x) for x in reviews if x not in
stopwords.words('english')]
    review = ' '.join(reviews)
    return review


df['Review'] = df['Review'].apply(rep)
df.head()
```

```
                                                Review  Liked
0                                    wow love place      1
1                                        crust good      0
2                                  tasti textur nasti      0
3  stop late may bank holiday rick steve recommen...      1
4                             select menu great price      1
```

```
corpus = df['Review'].tolist()
corpus[:5]
```

```
[u'wow love place',
 'crust good',
 u'tasti textur nasti',
 u'stop late may bank holiday rick steve recommend love',
 u'select menu great price']
```

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=1500)
X = cv.fit_transform(corpus).toarray()
y = df['Liked'].tolist()

from sklearn.model_selection import tyrain_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, random_state=0)


from sklearn.naive_bayes import GaussianNB

clf = GaussianNB()
clf.fit(X_train, y_train)
print 'Accuracy:{0: .1f}%'.format(clf.score(X_test, y_test) * 100)

Accuracy: 73.0%
```