Week 6 Text Classification

Program 6.1: Simple Text Classification

Code:-

```
from textblob import TextBlob
text="I love studying natural language processing!"
blob=TextBlob(text)
print("sentiment polarity:",blob.sentiment.polarity)
```

output:-

```
sentiment polarity: 0.3125
```

Program 6.2: Advanced Text Classification with Logistic Regression

Code:-

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
texts=["I love this movie","I hate this film"]
labels=[1,0]
vectorizer=TfidfVectorizer()
x=vectorizer.fit_transform(texts)
model=LogisticRegression()
model.fit(x,labels)
test=vectorizer.transform(["I love this movie"])
prediction=model.predict(test)
print("prediction:",prediction[0])
```

output:-

```
prediction: 1
```

Week 7 Named Entity Recognition (NER)

Program 7.1: Simple NER using spaCy

Code:-

```
pip install spacy

!python -m spacy download en_core_web_sm

import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion.")
for ent in doc.ents:
  print(ent.text, ent.label_)
```

Output:-

```
Apple ORG
U.K. GPE
$1 billion MONE
```

Program 7.2: Advanced NER with Entity Filtering

Code:-

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("Google has offices in New York and London.")
gpe_entities = [ent.text for ent in doc.ents if ent.label_ =='GPE']
print("Locations:", gpe_entities)
```

Output:-

```
Locations: ['New York', 'London']
```

Week 8 Context Free Grammars (CFG)

Program 8.1: Simple CFG Parsing

Code:-

```
import nltk
grammar=nltk.CFG.fromstring("""
S -> NP VP
NP -> Det N
VP -> V NP
Det -> 'the'
N -> 'dog' | 'cat'
V -> 'chased' | 'saw'
""")
parser=nltk.ChartParser(grammar)
sentence=('the', 'dog', 'chased', 'the', 'cat')
for tree in parser.parse(sentence):
   print(tree)
```

Output:-

```
(S (NP (Det the) (N dog)) (VP (V chased) (NP (Det the) (N cat))))
```

Program 8.2: Advanced CFG with Ambiguity Resolution

Code:-

```
grammar=nltk.CFG.fromstring("""
S -> NP VP
NP -> Det N | N
VP -> V NP
Det -> 'the'
N -> 'dog' | 'cat'
V -> 'chased'
""")
parser=nltk.ChartParser(grammar)
sentence=('the', 'cat', 'chased','dog')
for tree in parser.parse(sentence):
   print(tree)
```

Output:-

```
(S (NP (Det the) (N cat)) (VP (V chased) (NP (N dog))))
```

Week 9 Dependency Parsing using Stanza

9.1. Simple Dependency Parsing Program

Code:-

```
pip install stanza

import stanza
stanza.download('en')

import stanza
nlp = stanza.Pipeline('en')
sentence = "The quick brown fox jumps over the lazy dog."
doc = nlp(sentence)
print("Word\tHead\tRelation")
for sent in doc.sentences:
    for word in sent.words:
        head = sent.words[word.head - 1].text if word.head > 0 else "ROOT"
        print(f"{word.text}\t{head}\t{word.deprel}")
```

Output:-

Word	Head	Relation
The	fox	det
quick	fox	amod
brown	fox	amod
fox	jumps	nsubj
jumps	ROOT	root
over	dog	case
the	dog	det
lazy	dog	amod
dog	jumps	obl
.	jumps	punct

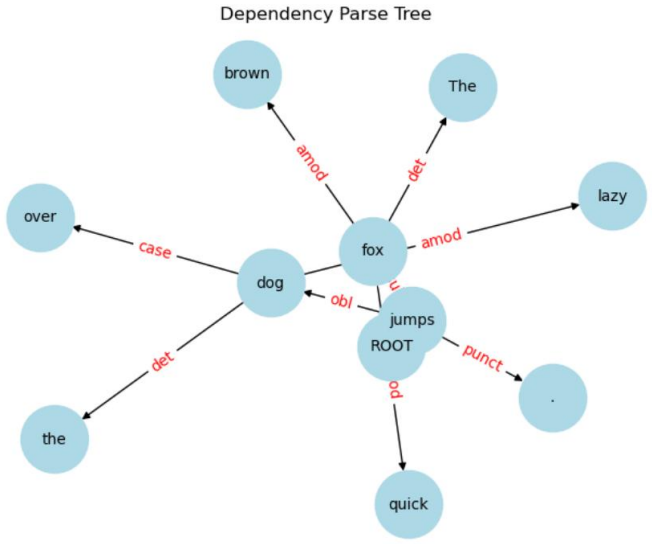
9.2. Advanced Dependency Tree Visualization using networkx + matplotlib

Code:-

```
import stanza
import networkx as nx
import matplotlib.pyplot as plt
nlp = stanza.Pipeline('en')
sentence = "The quick brown fox jumps over the lazy dog."
doc = nlp(sentence)
G = nx.DiGraph()
for sent in doc.sentences:
    for word in sent.words:
        head_text = "ROOT" if word.head == 0 else sent.words[word.head - 1].text
        G.add_edge(head_text, word.text, label=word.deprel)
pos = nx.spring_layout(G)
```

```
labels = nx.get_edge_attributes(G, 'label')
nx.draw(G, pos, with_labels=True, node_size=2000, node_color='lightblue', font_size=10)
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels, font_color='red')
plt.title("Dependency Parse Tree")
plt.show()
```

Output:-



Week 10 Word Sense Disambiguation (WSD)

Program 10.1: Simple WSD using Lesk Algorithm

Code:-

```
from nltk.wsd import lesk
from nltk.tokenize import word_tokenize
sentence = "I went to the bank to deposit money."
tokens = word_tokenize(sentence)
sense = lesk(tokens, 'bank')
print("Sense:", sense)
print("Definition:", sense.definition())
```

Output:-

Sense: Synset('savings_bank.n.02')

Definition: a container (usually with a slot in the top) for keeping money at home

Program 10.2: Advanced WSD with Multiple Words

Code:-

```
sentences = ["The crane is flying.", "He used a crane to lift the load."]
for sent in sentences:
    tokens = word_tokenize(sent)
    sense = lesk(tokens, 'crane')
    print("Sentence:", sent)
    print("Sense:", sense)
    print("Definition:", sense.definition())
```

Output:-
Sentence: The crane is flying.
Sense: Synset('crane.n.04')
Definition: lifts and moves heavy objects; lifting tackle is suspended from a pivoted boom that rotates around a vertical axis
Sentence: He used a crane to lift the load.
Sense: Synset('grus.n.01')
Definition: a small constellation in the southern hemisphere near Phoenix