**Name: - Anuj Kumar**

**Roll: - 18ETCS224001**

**Module: - MIS503**

**Lab Exam report**

**Date: - 14 – Dec \_2019**

========================Reg no 1=======================================

1. For the given textual data perform the following tasks
2. **Perform suitable text pre-processing and POS Tagging.**

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

stop\_words = set(stopwords.words('english'))

filtered\_sentence = [w for w in word\_tokens if not w in stop\_words]

filtered\_sentence = []

for w in word\_tokens:

if w not in stop\_words:

filtered\_sentence.append(w)

print("Word tokens are:",word\_tokens)

print("#########################################################################")

print("words after stoppedwords are: ",filtered\_sentence)

#POS Tagging

for tex in word\_tokens:

pos\_tag=nltk.pos\_tag([tex])

print(pos\_tag)

Given Text (short description:)

“A strike by farmers in Maharashtra continues to affect normal life, despite the State government’s announcement of an end to the strike last week. The farmers, whose demands include full waiver of farm loans, hikes in the minimum support price for agricultural produce and writing off of pending electricity bills …..”

* Text Pre-processing: -
  + Two Methods are implemented:
  + Word tokenization
  + Stop Word Removal
* Output of Word Tokenization: -

['A', 'strike', 'by', 'farmers', 'in', 'Maharashtra', 'continues', 'to', 'affect', 'normal', 'life', ',', 'despite', 'the', 'State', 'government', '’', 's', 'announcement', 'of', 'an', 'end', 'to', 'the', 'strike', 'last', 'week', '.', 'The', 'farmers', ',', 'whose', 'demands', 'include', 'full', 'waiver', 'of', 'farm', 'loans', ',', 'hikes', 'in', 'the', 'minimum', 'support', 'price', 'for', 'agricultural', 'produce', 'and', 'writing', 'off', 'of', 'pending', 'electricity', 'bills', ',', 'have', 'been', 'on', 'an', 'indefinite', 'strike', 'since', 'June’, …….. ]

* Output of Stop Word Removal :

words after stoppedwords are: ['A', 'strike', 'farmers', 'Maharashtra', 'continues', 'affect', 'normal', 'life', ',', 'despite', 'State', 'government', '’', 'announcement', 'end', 'strike', 'last', 'week', '.', 'The', 'farmers', ',', 'whose', 'demands', 'include', 'full', 'waiver', 'farm', 'loans', ',', 'hikes', 'minimum', 'support', 'price', 'agricultural', 'produce', 'writing', 'pending', 'electricity', 'bills', ',', 'indefinite', 'strike', 'since', 'June', '1', '.', 'As', 'strike', 'nears', 'end', 'first', 'week', ',', 'prices', 'essential', 'goods', 'milk’,

* POS Tagging

Output: -

[('A', 'DT')] [('strike', 'NN')] [('by', 'IN')] [('farmers', 'NNS')] [('in', 'IN')] [('Maharashtra', 'NNP')] [('continues', 'VBZ')] [('to', 'TO')] [('affect', 'NN')] [('normal', 'JJ')] [('life', 'NN')] [(',', ',')] [('despite', 'IN')] [('the', 'DT')] [('State', 'NN')] [('government', 'NN')] [('’', 'NN')] [('s', 'NN')] [('announcement', 'NN’)]

1. **Perform Noun to Noun mapping.**

import spacy

print(spacy.\_\_version\_\_)

### extracting dependency parsing because POS tagging is not sufficient many times

nlp = spacy.load('en\_core\_web\_sm')

#Example

doc = nlp("The 22-year-old recently won ATP Challenger tournament.")

for tok in doc:

print(tok.text, "...", tok.dep\_)

def get\_relation(sent):

doc = nlp(sent)

# Matcher class object

matcher = Matcher(nlp.vocab)

#define the pattern

pattern = [

#{'DEP':'ROOT'},

#{'DEP':'prep','OP':"?"},

#{'DEP':'agent','OP':"?"},

#{'POS':'ADJ','OP':"?"},

#{"POS": "ADV", "OP": "\*"},

#{'DEP':'amod', 'OP':"?"},

{'DEP':'pobj','OP':'+'},

{'DEP':'nsubj','OP':'?'},

{'POS':'nsubj','OP':'?'}

]

matcher.add("matching\_1", None, pattern)

matches = matcher(doc)

k = len(matches) - 1

span = doc[matches[k][1]:matches[k][2]]

return(span.text)

* Noun to Noun Mapping: -

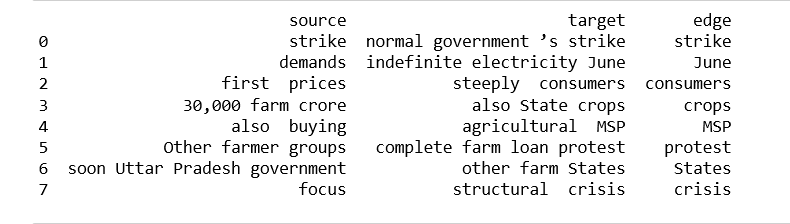
pattern = [{'DEP':'pobj','OP':'+'},

{'DEP':'nsubj','OP':'?'},

{'POS':'nsubj','OP':'?’}]

* Output: -

['strike', 'June', 'consumers', 'crops', 'MSP', 'protest', 'States', 'crisis’]



1. **Develop a Bayesian network with maximum of 10 nodes.**

#we create a dataframe of entities and predicates:

# extract subject

source = [i[0] for i in entity\_pairs]

# extract object

target = [i[1] for i in entity\_pairs]

kg\_df = pd.DataFrame({'source':source, 'target':target, 'edge':relations})

# In[58]:

#print(source[1:10])

#print(target[1:10])

print(kg\_df)

# In[59]:

# create a directed-graph from a dataframe

G=nx.from\_pandas\_edgelist(kg\_df, "source", "target",

edge\_attr=True, create\_using=nx.MultiDiGraph())

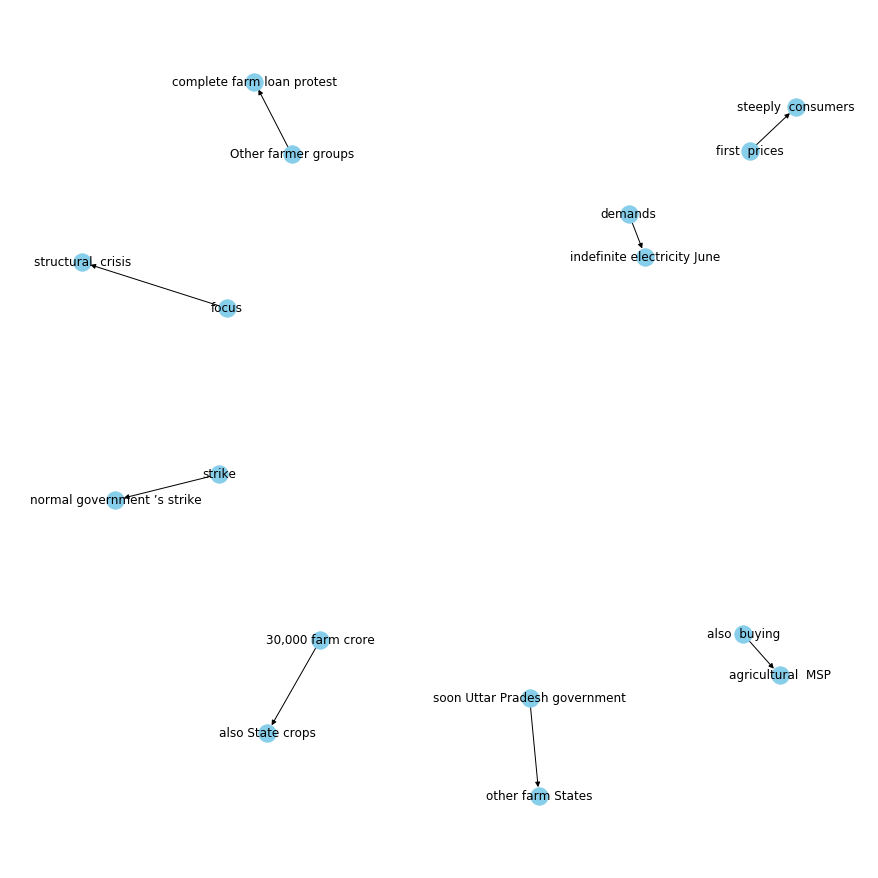
# In[60]:

plt.figure(figsize=(12,12))

pos = nx.spring\_layout(G)

nx.draw(G, with\_labels=True, node\_color='skyblue', edge\_cmap=plt.cm.Blues, pos = pos)

plt.show()



d) **Analyse the conditional probability distribution for any two nodes of the developed network.**

Code:

G=nx.from\_pandas\_edgelist(kg\_df[kg\_df['edge']=="crops"], "source", "target",

edge\_attr=True, create\_using=nx.MultiDiGraph())

