lab_exam

December 3, 2019

[1]:
```
import os
base_path="C:\\Users\\i345144\\OneDrive\\Documents\\MSRUS\\Probabilistic␣
 ↪Graphical Models\\Lab Exam\\Lab Exam"

os.chdir(base_path)

# Importing necessary library
import pandas as pd
import numpy as np
import nltk
import os
import nltk.corpus
# importing word_tokenize from nltk
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk import ne_chunk

import nltk

# uncomment and run when using first time
#nltk.download('stopwords')
#nltk.download('averaged_perceptron_tagger')
#nltk.download('words')
#nltk.download('maxent_ne_chunker')
```

[2]:
```
my_file = os.path.join(base_path + "/" + "MIS503 Lab Exam Data.txt")

f = open(my_file, "r")
extracted_text= f.read()
print(extracted_text)
```

```
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 1. As the strike nears the end of
```

its first week, prices of essential goods such as milk, fruits and vegetables have risen steeply, causing distress to consumers. Some farmer groups agreed to call off their strike after Chief Minister Devendra Fadnavis promised that his government would waive farm loans of small and marginal farmers worth about ?30,000 crore, increase power subsidies, hike the price for milk procurement, and also set up a State commission to look into the matter of raising the MSP for crops. He also promised that buying agricultural produce below their MSP would soon be made a criminal offence. Other farmer groups, meanwhile, have stuck to their demand for a complete farm loan waiver and continued with their protest. It is notable that the protests have come soon after the Uttar Pradesh government waived farm loans earlier this year, setting off similar demands in other States. Yet, while Maharashtra's farmers have caught the attention of the government, the focus on quick fixes has pushed aside the real structural issues behind the crisis.

[3]:
```
story=extracted_text
# we do this to maintain consistency in the code based on code in assignment
print(story)
```

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 1. As the strike nears the end of its first week, prices of essential goods such as milk, fruits and vegetables have risen steeply, causing distress to consumers. Some farmer groups agreed to call off their strike after Chief Minister Devendra Fadnavis promised that his government would waive farm loans of small and marginal farmers worth about ?30,000 crore, increase power subsidies, hike the price for milk procurement, and also set up a State commission to look into the matter of raising the MSP for crops. He also promised that buying agricultural produce below their MSP would soon be made a criminal offence. Other farmer groups, meanwhile, have stuck to their demand for a complete farm loan waiver and continued with their protest. It is notable that the protests have come soon after the Uttar Pradesh government waived farm loans earlier this year, setting off similar demands in other States. Yet, while Maharashtra's farmers have caught the attention of the government, the focus on quick fixes has pushed aside the real structural issues behind the crisis.

[4]:
```
# Passing the string text into word tokenize for breaking the sentences
word_tokens = word_tokenize(story)
print(word_tokens)
```

['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', '1', '.', 'As', 'the', 'strike', 'nears', 'the', 'end', 'of',
'its', 'first', 'week', ',', 'prices', 'of', 'essential', 'goods', 'such', 'as',
'milk', ',', 'fruits', 'and', 'vegetables', 'have', 'risen', 'steeply', ',',
'causing', 'distress', 'to', 'consumers', '.', 'Some', 'farmer', 'groups',
'agreed', 'to', 'call', 'off', 'their', 'strike', 'after', 'Chief', 'Minister',
'Devendra', 'Fadnavis', 'promised', 'that', 'his', 'government', 'would',
'waive', 'farm', 'loans', 'of', 'small', 'and', 'marginal', 'farmers', 'worth',
'about', '?', '30,000', 'crore', ',', 'increase', 'power', 'subsidies', ',',
'hike', 'the', 'price', 'for', 'milk', 'procurement', ',', 'and', 'also', 'set',
'up', 'a', 'State', 'commission', 'to', 'look', 'into', 'the', 'matter', 'of',
'raising', 'the', 'MSP', 'for', 'crops', '.', 'He', 'also', 'promised', 'that',
'buying', 'agricultural', 'produce', 'below', 'their', 'MSP', 'would', 'soon',
'be', 'made', 'a', 'criminal', 'offence', '.', 'Other', 'farmer', 'groups', ',',
'meanwhile', ',', 'have', 'stuck', 'to', 'their', 'demand', 'for', 'a',
'complete', 'farm', 'loan', 'waiver', 'and', 'continued', 'with', 'their',
'protest', '.', 'It', 'is', 'notable', 'that', 'the', 'protests', 'have',
'come', 'soon', 'after', 'the', 'Uttar', 'Pradesh', 'government', 'waived',
'farm', 'loans', 'earlier', 'this', 'year', ',', 'setting', 'off', 'similar',
'demands', 'in', 'other', 'States', '.', 'Yet', ',', 'while', 'Maharashtra',
''', 's', 'farmers', 'have', 'caught', 'the', 'attention', 'of', 'the',
'government', ',', 'the', 'focus', 'on', 'quick', 'fixes', 'has', 'pushed',
'aside', 'the', 'real', 'structural', 'issues', 'behind', 'the', 'crisis', '.']
```

[5]:
```
# finding the frequency distinct in the tokens
# Importing FreqDist library from nltk and passing token into FreqDist
fdist = FreqDist(word_tokens)
fdist
```

[5]:
```
FreqDist({',': 15, 'the': 15, 'of': 8, '.': 8, 'to': 6, 'strike': 5, 'and': 5,
'have': 5, 'farmers': 4, 'government': 4, …})
```

[6]:
```
# To find the frequency of top 10 words
fdist1 = fdist.most_common(10)
fdist1
```

[6]:
```
[(',', 15),
 ('the', 15),
 ('of', 8),
 ('.', 8),
 ('to', 6),
 ('strike', 5),
 ('and', 5),
 ('have', 5),
```

```
 ('farmers', 4),
 ('government', 4)]
```

[8]:
```
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)
```

```
Word tokens are: ['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', '1', '.', 'As', 'the',
'strike', 'nears', 'the', 'end', 'of', 'its', 'first', 'week', ',', 'prices',
'of', 'essential', 'goods', 'such', 'as', 'milk', ',', 'fruits', 'and',
'vegetables', 'have', 'risen', 'steeply', ',', 'causing', 'distress', 'to',
'consumers', '.', 'Some', 'farmer', 'groups', 'agreed', 'to', 'call', 'off',
'their', 'strike', 'after', 'Chief', 'Minister', 'Devendra', 'Fadnavis',
'promised', 'that', 'his', 'government', 'would', 'waive', 'farm', 'loans',
'of', 'small', 'and', 'marginal', 'farmers', 'worth', 'about', '?', '30,000',
'crore', ',', 'increase', 'power', 'subsidies', ',', 'hike', 'the', 'price',
'for', 'milk', 'procurement', ',', 'and', 'also', 'set', 'up', 'a', 'State',
'commission', 'to', 'look', 'into', 'the', 'matter', 'of', 'raising', 'the',
'MSP', 'for', 'crops', '.', 'He', 'also', 'promised', 'that', 'buying',
'agricultural', 'produce', 'below', 'their', 'MSP', 'would', 'soon', 'be',
'made', 'a', 'criminal', 'offence', '.', 'Other', 'farmer', 'groups', ',',
'meanwhile', ',', 'have', 'stuck', 'to', 'their', 'demand', 'for', 'a',
'complete', 'farm', 'loan', 'waiver', 'and', 'continued', 'with', 'their',
'protest', '.', 'It', 'is', 'notable', 'that', 'the', 'protests', 'have',
'come', 'soon', 'after', 'the', 'Uttar', 'Pradesh', 'government', 'waived',
'farm', 'loans', 'earlier', 'this', 'year', ',', 'setting', 'off', 'similar',
'demands', 'in', 'other', 'States', '.', 'Yet', ',', 'while', 'Maharashtra',
'’', 's', 'farmers', 'have', 'caught', 'the', 'attention', 'of', 'the',
```

```
'government', ',', 'the', 'focus', 'on', 'quick', 'fixes', 'has', 'pushed',
'aside', 'the', 'real', 'structural', 'issues', 'behind', 'the', 'crisis', '.']
########################################################################
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', ',', 'fruits', 'vegetables', 'risen', 'steeply',
',', 'causing', 'distress', 'consumers', '.', 'Some', 'farmer', 'groups',
'agreed', 'call', 'strike', 'Chief', 'Minister', 'Devendra', 'Fadnavis',
'promised', 'government', 'would', 'waive', 'farm', 'loans', 'small',
'marginal', 'farmers', 'worth', '?', '30,000', 'crore', ',', 'increase',
'power', 'subsidies', ',', 'hike', 'price', 'milk', 'procurement', ',', 'also',
'set', 'State', 'commission', 'look', 'matter', 'raising', 'MSP', 'crops', '.',
'He', 'also', 'promised', 'buying', 'agricultural', 'produce', 'MSP', 'would',
'soon', 'made', 'criminal', 'offence', '.', 'Other', 'farmer', 'groups', ',',
'meanwhile', ',', 'stuck', 'demand', 'complete', 'farm', 'loan', 'waiver',
'continued', 'protest', '.', 'It', 'notable', 'protests', 'come', 'soon',
'Uttar', 'Pradesh', 'government', 'waived', 'farm', 'loans', 'earlier', 'year',
',', 'setting', 'similar', 'demands', 'States', '.', 'Yet', ',', 'Maharashtra',
''', 'farmers', 'caught', 'attention', 'government', ',', 'focus', 'quick',
'fixes', 'pushed', 'aside', 'real', 'structural', 'issues', 'behind', 'crisis',
'.']
```

[9]:
```python
#POS Tagging
for tex in word_tokens:
    pos_tag=nltk.pos_tag([tex])
    print(pos_tag)
```

```
[('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')]
[('of', 'IN')]
[('an', 'DT')]
[('end', 'NN')]
[('to', 'TO')]
[('the', 'DT')]
[('strike', 'NN')]
[('last', 'JJ')]
[('week', 'NN')]
[('.', '.')]
[('The', 'DT')]
[('farmers', 'NNS')]
[(',', ',')]
[('whose', 'WP$')]
[('demands', 'NNS')]
[('include', 'NN')]
[('full', 'JJ')]
[('waiver', 'NN')]
[('of', 'IN')]
[('farm', 'NN')]
[('loans', 'NNS')]
[(',', ',')]
[('hikes', 'NNS')]
[('in', 'IN')]
[('the', 'DT')]
[('minimum', 'NN')]
[('support', 'NN')]
[('price', 'NN')]
[('for', 'IN')]
[('agricultural', 'JJ')]
[('produce', 'NN')]
[('and', 'CC')]
[('writing', 'VBG')]
[('off', 'IN')]
[('of', 'IN')]
[('pending', 'VBG')]
[('electricity', 'NN')]
[('bills', 'NNS')]
[(',', ',')]
[('have', 'VB')]
[('been', 'VBN')]
[('on', 'IN')]
[('an', 'DT')]
[('indefinite', 'NN')]
[('strike', 'NN')]
[('since', 'IN')]
```

```
[('June', 'NNP')]
[('1', 'CD')]
[('.', '.')]
[('As', 'IN')]
[('the', 'DT')]
[('strike', 'NN')]
[('nears', 'NNS')]
[('the', 'DT')]
[('end', 'NN')]
[('of', 'IN')]
[('its', 'PRP$')]
[('first', 'RB')]
[('week', 'NN')]
[(',', ',')]
[('prices', 'NNS')]
[('of', 'IN')]
[('essential', 'JJ')]
[('goods', 'NNS')]
[('such', 'JJ')]
[('as', 'IN')]
[('milk', 'NN')]
[(',', ',')]
[('fruits', 'NNS')]
[('and', 'CC')]
[('vegetables', 'NNS')]
[('have', 'VB')]
[('risen', 'NN')]
[('steeply', 'NN')]
[(',', ',')]
[('causing', 'VBG')]
[('distress', 'NN')]
[('to', 'TO')]
[('consumers', 'NNS')]
[('.', '.')]
[('Some', 'DT')]
[('farmer', 'NN')]
[('groups', 'NNS')]
[('agreed', 'VBD')]
[('to', 'TO')]
[('call', 'NN')]
[('off', 'IN')]
[('their', 'PRP$')]
[('strike', 'NN')]
[('after', 'IN')]
[('Chief', 'NN')]
[('Minister', 'NNP')]
[('Devendra', 'NN')]
[('Fadnavis', 'NN')]
```

```
[('promised', 'VBN')]
[('that', 'IN')]
[('his', 'PRP$')]
[('government', 'NN')]
[('would', 'MD')]
[('waive', 'NN')]
[('farm', 'NN')]
[('loans', 'NNS')]
[('of', 'IN')]
[('small', 'JJ')]
[('and', 'CC')]
[('marginal', 'JJ')]
[('farmers', 'NNS')]
[('worth', 'NN')]
[('about', 'IN')]
[('?', '.')]
[('30,000', 'CD')]
[('crore', 'NN')]
[(',', ',')]
[('increase', 'NN')]
[('power', 'NN')]
[('subsidies', 'NNS')]
[(',', ',')]
[('hike', 'NN')]
[('the', 'DT')]
[('price', 'NN')]
[('for', 'IN')]
[('milk', 'NN')]
[('procurement', 'NN')]
[(',', ',')]
[('and', 'CC')]
[('also', 'RB')]
[('set', 'NN')]
[('up', 'RB')]
[('a', 'DT')]
[('State', 'NN')]
[('commission', 'NN')]
[('to', 'TO')]
[('look', 'NN')]
[('into', 'IN')]
[('the', 'DT')]
[('matter', 'NN')]
[('of', 'IN')]
[('raising', 'VBG')]
[('the', 'DT')]
[('MSP', 'NN')]
[('for', 'IN')]
[('crops', 'NNS')]
```

```
[('.', '.')]
[('He', 'PRP')]
[('also', 'RB')]
[('promised', 'VBN')]
[('that', 'IN')]
[('buying', 'NN')]
[('agricultural', 'JJ')]
[('produce', 'NN')]
[('below', 'IN')]
[('their', 'PRP$')]
[('MSP', 'NN')]
[('would', 'MD')]
[('soon', 'RB')]
[('be', 'VB')]
[('made', 'VBN')]
[('a', 'DT')]
[('criminal', 'JJ')]
[('offence', 'NN')]
[('.', '.')]
[('Other', 'JJ')]
[('farmer', 'NN')]
[('groups', 'NNS')]
[(',', ',')]
[('meanwhile', 'RB')]
[(',', ',')]
[('have', 'VB')]
[('stuck', 'NN')]
[('to', 'TO')]
[('their', 'PRP$')]
[('demand', 'NN')]
[('for', 'IN')]
[('a', 'DT')]
[('complete', 'JJ')]
[('farm', 'NN')]
[('loan', 'NN')]
[('waiver', 'NN')]
[('and', 'CC')]
[('continued', 'JJ')]
[('with', 'IN')]
[('their', 'PRP$')]
[('protest', 'NN')]
[('.', '.')]
[('It', 'PRP')]
[('is', 'VBZ')]
[('notable', 'JJ')]
[('that', 'IN')]
[('the', 'DT')]
[('protests', 'NNS')]
```

```
[('have', 'VB')]
[('come', 'VB')]
[('soon', 'RB')]
[('after', 'IN')]
[('the', 'DT')]
[('Uttar', 'NN')]
[('Pradesh', 'NN')]
[('government', 'NN')]
[('waived', 'VBN')]
[('farm', 'NN')]
[('loans', 'NNS')]
[('earlier', 'RBR')]
[('this', 'DT')]
[('year', 'NN')]
[(',', ',')]
[('setting', 'VBG')]
[('off', 'IN')]
[('similar', 'JJ')]
[('demands', 'NNS')]
[('in', 'IN')]
[('other', 'JJ')]
[('States', 'NNS')]
[('.', '.')]
[('Yet', 'RB')]
[(',', ',')]
[('while', 'IN')]
[('Maharashtra', 'NNP')]
[('’', 'NN')]
[('s', 'NN')]
[('farmers', 'NNS')]
[('have', 'VB')]
[('caught', 'NN')]
[('the', 'DT')]
[('attention', 'NN')]
[('of', 'IN')]
[('the', 'DT')]
[('government', 'NN')]
[(',', ',')]
[('the', 'DT')]
[('focus', 'NN')]
[('on', 'IN')]
[('quick', 'NN')]
[('fixes', 'NNS')]
[('has', 'VBZ')]
[('pushed', 'VBN')]
[('aside', 'RB')]
[('the', 'DT')]
[('real', 'JJ')]
```

```
[('structural', 'JJ')]
[('issues', 'NNS')]
[('behind', 'IN')]
[('the', 'DT')]
[('crisis', 'NN')]
[('.', '.')]
```

[5]:
```
## part C begins here
```

[14]:
```
#conda install -c conda-forge spacy=2.2.1
# conda install -c conda-forge spacy-model-en_core_web_sm
#https://www.analyticsvidhya.com/blog/2019/10/
 ↪how-to-build-knowledge-graph-text-using-spacy/
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_)
```

```
2.2.1
The … det
22-year … npadvmod
- … punct
old … nsubj
recently … advmod
won … ROOT
ATP … compound
Challenger … compound
tournament … dobj
. … punct
```

[15]:
```
import re
import pandas as pd
import bs4
import requests
import spacy
from spacy import displacy
nlp = spacy.load('en_core_web_sm')

from spacy.matcher import Matcher
```

```
from spacy.tokens import Span

import networkx as nx

import matplotlib.pyplot as plt
from tqdm import tqdm

pd.set_option('display.max_colwidth', 200)
%matplotlib inline
```

[16]:
```
doc=nlp(story)

for tok in doc:
  print(tok.text, "...", tok.dep_)
```

```
A … det
strike … nsubj
by … prep
farmers … pobj
in … prep
Maharashtra … pobj
continues … ROOT
to … aux
affect … xcomp
normal … amod
life … dobj
, … punct
despite … prep
the … det
State … compound
government … compound
’s … compound
announcement … pobj
of … prep
an … det
end … pobj
to … prep
the … det
strike … pobj
last … amod
week … npadvmod
. … punct
The … det
farmers … nsubj
, … punct
whose … poss
demands … nsubj
```

include … relcl
full … amod
waiver … dobj
of … prep
farm … compound
loans … pobj
, … punct
hikes … conj
in … prep
the … det
minimum … amod
support … compound
price … pobj
for … prep
agricultural … amod
produce … pobj
and … cc
writing … conj
off … prt
of … prep
pending … amod
electricity … compound
bills … pobj
, … punct
have … aux
been … ROOT
on … prep
an … det
indefinite … amod
strike … pobj
since … prep
June … pobj
1 … nummod
. … punct
As … mark
the … det
strike … nsubj
nears … advcl
the … det
end … dobj
of … prep
its … poss
first … amod
week … pobj
, … punct
prices … nsubj
of … prep
essential … amod

goods … pobj
such … amod
as … prep
milk … pobj
, … punct
fruits … conj
and … cc
vegetables … conj
have … aux
risen … ROOT
steeply … advmod
, … punct
causing … advcl
distress … dobj
to … dative
consumers … pobj
. … punct
Some … det
farmer … compound
groups … nsubj
agreed … ROOT
to … aux
call … xcomp
off … prt
their … poss
strike … dobj
after … mark
Chief … compound
Minister … compound
Devendra … compound
Fadnavis … nsubj
promised … advcl
that … mark
his … poss
government … nsubj
would … aux
waive … ccomp
farm … compound
loans … dobj
of … prep
small … amod
and … cc
marginal … conj
farmers … pobj
worth … amod
about … prep
? … punct
30,000 … nummod

crore … nsubj
, … punct
increase … ROOT
power … compound
subsidies … dobj
, … punct
hike … conj
the … det
price … dobj
for … prep
milk … compound
procurement … pobj
, … punct
and … cc
also … advmod
set … conj
up … prt
a … det
State … compound
commission … dobj
to … aux
look … advcl
into … prep
the … det
matter … pobj
of … prep
raising … pcomp
the … det
MSP … dobj
for … prep
crops … pobj
. … punct
He … nsubj
also … advmod
promised … ROOT
that … mark
buying … csubjpass
agricultural … amod
produce … dobj
below … prep
their … poss
MSP … pobj
would … aux
soon … advmod
be … auxpass
made … ccomp
a … det
criminal … amod

offence … oprd
. … punct
Other … amod
farmer … compound
groups … nsubj
, … punct
meanwhile … advmod
, … punct
have … aux
stuck … ROOT
to … prep
their … poss
demand … pobj
for … prep
a … det
complete … amod
farm … compound
loan … compound
waiver … pobj
and … cc
continued … conj
with … prep
their … poss
protest … pobj
. … punct
It … nsubj
is … ROOT
notable … acomp
that … mark
the … det
protests … nsubj
have … aux
come … ccomp
soon … advmod
after … mark
the … det
Uttar … compound
Pradesh … compound
government … nsubj
waived … advcl
farm … compound
loans … dobj
earlier … advmod
this … det
year … npadvmod
, … punct
setting … advcl
off … prt

```
similar … amod
demands … dobj
in … prep
other … amod
States … pobj
. … punct
Yet … advmod
, … punct
while … mark
Maharashtra … compound
's … compound
farmers … nsubj
have … aux
caught … advcl
the … det
attention … dobj
of … prep
the … det
government … pobj
, … punct
the … det
focus … nsubj
on … prep
quick … amod
fixes … pobj
has … aux
pushed … ROOT
aside … advmod
the … det
real … amod
structural … amod
issues … dobj
behind … prep
the … det
crisis … pobj
. … punct

 …
```

[17]:
```
#The main idea is to go through a sentence and extract the subject and the␣
 ↪object as and when they are encountered so that we can
# have nodes and edges for the graph

def get_entities(sent):
  ## chunk 1
  ent1 = ""
  ent2 = ""
```

```
 prv_tok_dep = ""    # dependency tag of previous token in the sentence
 prv_tok_text = ""   # previous token in the sentence

 prefix = ""
 modifier = ""

 #############################################################

 for tok in nlp(sent):
   ## chunk 2
   # if token is a punctuation mark then move on to the next token
   if tok.dep_ != "punct":
     # check: token is a compound word or not
     if tok.dep_ == "compound":
       prefix = tok.text
       # if the previous word was also a 'compound' then add the current word␣
↪to it
       if prv_tok_dep == "compound":
         prefix = prv_tok_text + " "+ tok.text

     # check: token is a modifier or not
     if tok.dep_.endswith("mod") == True:
       modifier = tok.text
       # if the previous word was also a 'compound' then add the current word␣
↪to it
       if prv_tok_dep == "compound":
         modifier = prv_tok_text + " "+ tok.text

     ## chunk 3
     if tok.dep_.find("subj") == True:
       ent1 = modifier +" "+ prefix + " "+ tok.text
       prefix = ""
       modifier = ""
       prv_tok_dep = ""
       prv_tok_text = ""

     ## chunk 4
     if tok.dep_.find("obj") == True:
       ent2 = modifier +" "+ prefix +" "+ tok.text

     ## chunk 5
     # update variables
     prv_tok_dep = tok.dep_
     prv_tok_text = tok.text
 #############################################################
```

```
  return [ent1.strip(), ent2.strip()]
```

[ ]:

[21]:
```
from nltk import sent_tokenize

sentence=sent_tokenize(story)
print (sentence)
```

```
['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 1.', 'As the strike nears
the end of its first week, prices of essential goods such as milk, fruits and
vegetables have risen steeply, causing distress to consumers.', 'Some farmer
groups agreed to call off their strike after Chief Minister Devendra Fadnavis
promised that his government would waive farm loans of small and marginal
farmers worth about ?30,000 crore, increase power subsidies, hike the price for
milk procurement, and also set up a State commission to look into the matter of
raising the MSP for crops.', 'He also promised that buying agricultural produce
below their MSP would soon be made a criminal offence.', 'Other farmer groups,
meanwhile, have stuck to their demand for a complete farm loan waiver and
continued with their protest.', 'It is notable that the protests have come soon
after the Uttar Pradesh government waived farm loans earlier this year, setting
off similar demands in other States.', 'Yet, while Maharashtra's farmers have
caught the attention of the government, the focus on quick fixes has pushed
aside the real structural issues behind the crisis.']
```

[22]:
```
entity_pairs = []

for i in tqdm(sentence):
    entity_pairs.append(get_entities(i))
```

```
100%|
    | 8/8 [00:00<00:00, 59.70it/s]
```

[23]:
```
entity_pairs[10:20]
```

[23]:
```
[]
```

[52]:
```
#Our hypothesis is that the predicate is actually the main verb in a sentence.
#The function below is capable of capturing such predicates from the sentences.
#The pattern defined in the function tries to find the ROOT word or the main␣
 ↪verb in the sentence.
#Once the ROOT is identified, then the pattern checks whether it is followed by␣
 ↪a preposition ('prep')
```

```
#or an agent word. If yes, then it is added to the ROOT word.


# some patterns and their meaning

#GDP --> nsubj --> NOUN
#in --> prep --> ADP
#developing --> amod --> VERB
#countries --> pobj --> NOUN
#such --> amod --> ADJ
#as --> prep --> ADP
#Vietnam --> pobj --> PROPN
#will --> aux --> VERB
#continue --> ROOT --> VERB
#growing --> xcomp --> VERB
#at --> prep --> ADP
#a --> det --> DET
#high --> amod --> ADJ
#rate --> pobj --> NOUN
#. --> punct --> PUNCT

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)
```

[53]:
```
relations = [get_relation(i) for i in tqdm(sentence)]
```

```
  0%|
| 0/8 [00:00<?, ?it/s]
```

```
 75%|
| 6/8 [00:00<00:00, 57.14it/s]
```

```
100%|
     | 8/8 [00:00<00:00, 48.78it/s]
```

[54]:
```
print(relations)
```

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

[55]:
```
#most frequent relations or predicates that we have just extracted:
pd.Series(relations).value_counts()[:10]
```

[55]:
```
consumers    1
crops        1
June         1
protest      1
strike       1
MSP          1
States       1
crisis       1
dtype: int64
```

[56]:
```
total= len(relations)
total
### joint probability table
prob_table = pd.Series(relations).value_counts()[:10]/ total
```

```
print(prob_table)
```

```
consumers    0.125
crops        0.125
June         0.125
protest      0.125
strike       0.125
MSP          0.125
States       0.125
crisis       0.125
dtype: float64
```

[ ]:

[57]:
```
#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})
```

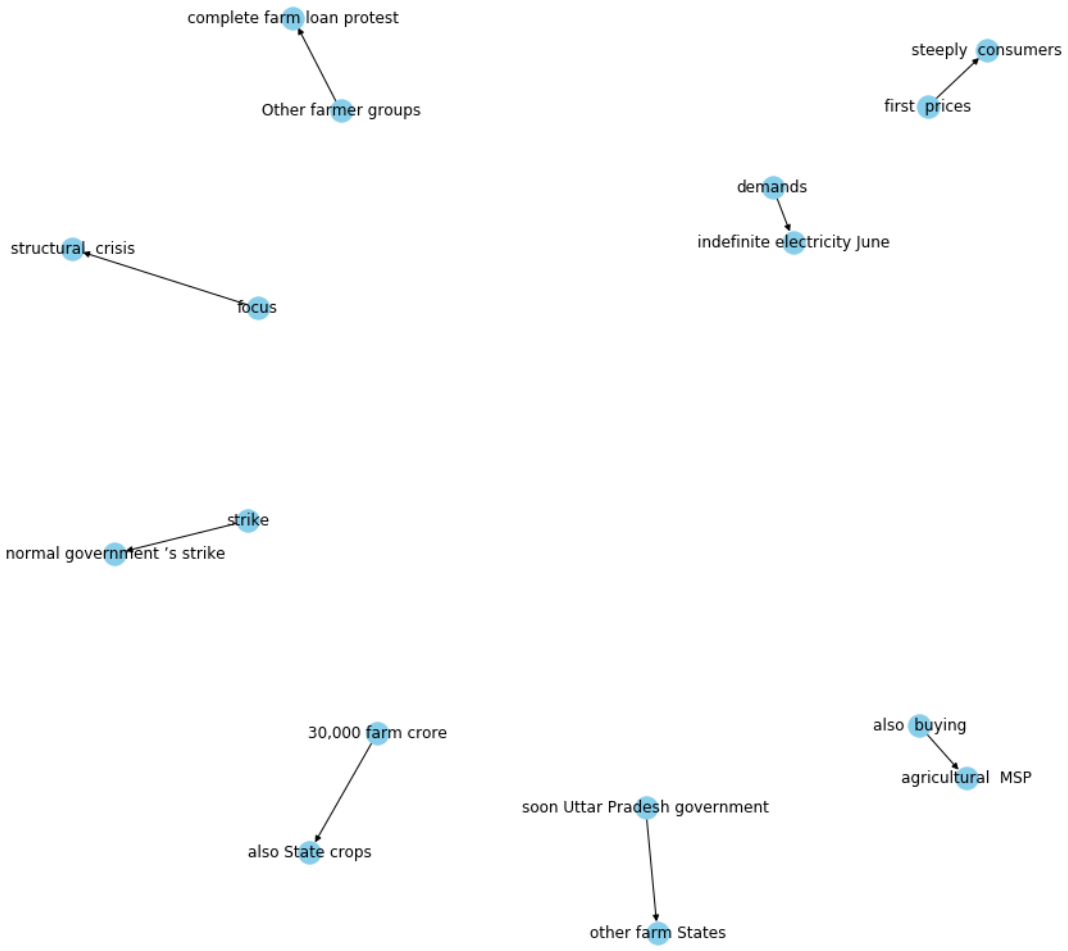
[58]:
```
#print(source[1:10])
#print(target[1:10])

print(kg_df)
```

```
                          source                          target       edge
0                         strike  normal government 's strike     strike
1                        demands  indefinite electricity June       June
2                  first  prices           steeply  consumers  consumers
3              30,000 farm crore             also State crops      crops
4                   also  buying            agricultural  MSP        MSP
5            Other farmer groups   complete farm loan protest    protest
6  soon Uttar Pradesh government            other farm States     States
7                           focus           structural  crisis     crisis
```

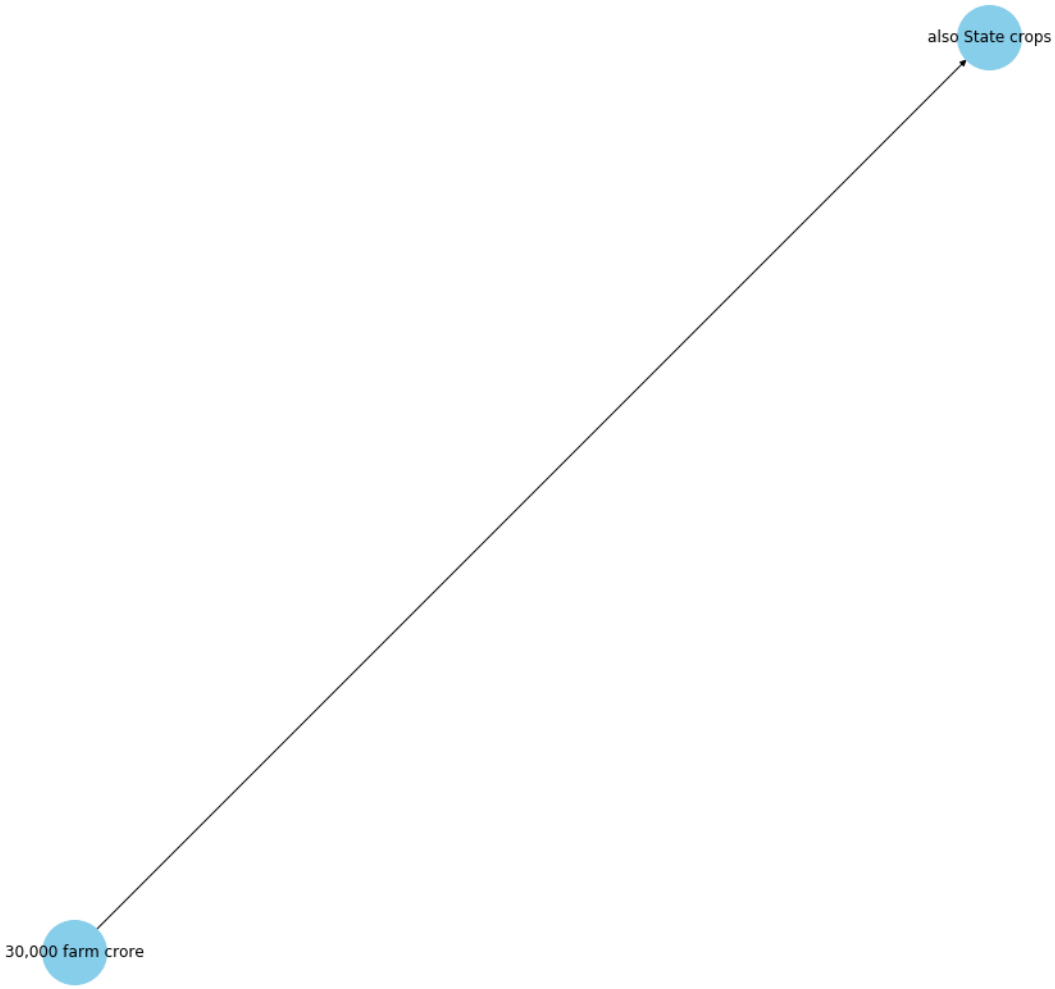
[59]:
```
# create a directed-graph from a dataframe
G=nx.from_pandas_edgelist(kg_df, "source", "target",
                          edge_attr=True, create_using=nx.MultiDiGraph())
```

[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()
```



[61]:
```
G=nx.from_pandas_edgelist(kg_df[kg_df['edge']=="crops"], "source", "target",
                          edge_attr=True, create_using=nx.MultiDiGraph())

plt.figure(figsize=(12,12))
pos = nx.spring_layout(G, k=0.5) # k regulates the distance between nodes
nx.draw(G, with_labels=True, node_color='skyblue', node_size=2500,␣
 ↪edge_cmap=plt.cm.Blues, pos = pos)
plt.show()
```


also State crops
30,000 farm crore

[ ]: