Bollywood Movie Analysis

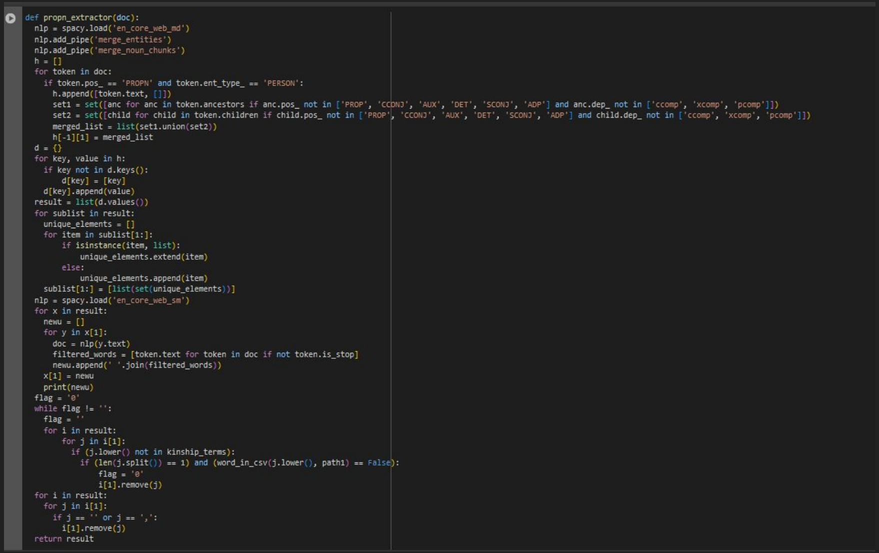
Using SpaCy

Using SpaCy, I have created a pipeline to break down the plots of Bollywood movies, isolate the characters and their characteristics, mainly:

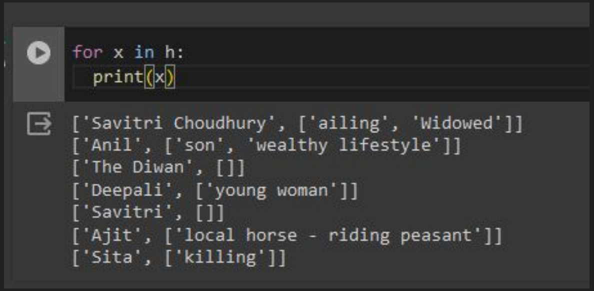
- Familial relations
- Professions
- Qualities

This was done by using dependency parsing to isolate verbs and adjectives, while filtering out non-meaningful stopwords in the process.

A function was created to isolate entities with the label PERSON, then further isolate their constituent traits, then create a multi-dimensional list consisting of their names + relevant traits.



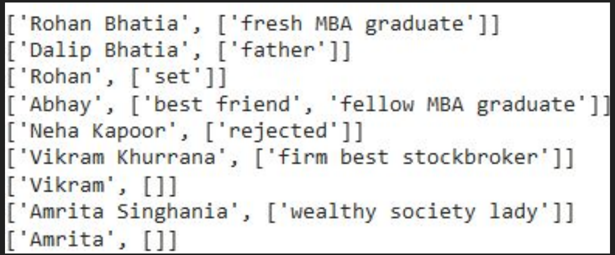
```
def propn_extractor(doc):
  nlp = spacy.load('en_core_web_md')
  nlp.add_pipe('merge_entities')
  nlp.add_pipe('merge_noun_chunks')
  h = []
  for token in doc:
    if token.pos_ == 'PROPN' and token.ent_type_ == 'PERSON':
      h.append([token.text, []])
      set1 = set([anc for anc in token.ancestors if anc.pos_ not in ['PROP', 'CCONJ', 'AUX', 'DET', 'SCONJ', 'ADP'] and anc.dep_ not in ['ccomp', 'xcomp', 'pcomp']])
      set2 = set([child for child in token.children if child.pos_ not in ['PROP', 'CCONJ', 'AUX', 'DET', 'SCONJ', 'ADP'] and child.dep_ not in ['ccomp', 'xcomp', 'pcomp']])
      merged_list = list(set1.union(set2))
      h[-1][1] = merged_list
  d = {}
  for key, value in h:
    if key not in d.keys():
        d[key] = [key]
    d[key].append(value)
  result = list(d.values())
  for sublist in result:
    unique_elements = []
    for item in sublist[1:]:
        if isinstance(item, list):
            unique_elements.extend(item)
        else:
            unique_elements.append(item)
    sublist[1:] = [list(set(unique_elements))]
  nlp = spacy.load('en_core_web_sm')
  for x in result:
    newu = []
    for y in x[1]:
      doc = nlp(y.text)
      filtered_words = [token.text for token in doc if not token.is_stop]
      newu.append(' '.join(filtered_words))
    x[1] = newu
    print(newu)
  flag = '0'
  while flag != '':
    flag = ''
    for i in result:
        for j in i[1]:
          if (j.lower() not in kinship_terms):
            if (len(j.split()) == 1) and (word_in_csv(j.lower(), path1) == False):
                flag = '0'
                i[1].remove(j)
  for i in result:
    for j in i[1]:
      if j == '' or j == ',':
        i[1].remove(j)
  return result
```



```
for x in h:
  print(x)
```

```
['Savitri Choudhury', ['ailing', 'Widowed']]
['Anil', ['son', 'wealthy lifestyle']]
['The Diwan', []]
['Deepali', ['young woman']]
['Savitri', []]
['Ajit', ['local horse - riding peasant']]
['Sita', ['killing']]
```

Output for the synopsis of the movie Aan Milo Sajna



```
['Rohan Bhatia', ['fresh MBA graduate']]
['Dalip Bhatia', ['father']]
['Rohan', ['set']]
['Abhay', ['best friend', 'fellow MBA graduate']]
['Neha Kapoor', ['rejected']]
['Vikram Khurrana', ['firm best stockbroker']]
['Vikram', []]
['Amrita Singhania', ['wealthy society lady']]
['Amrita', []]
```

Cell 5, function propn_extractor

Output for the synopsis of the movie Jo Hum Chahhein

After this, the attributes were labelled as regressive (-) or progressive (+) ones based on the following parameters:

- Defined solely by family relation (identified via kinship terminology list lookup)
- Defined solely by the external gaze (wealth, attractiveness):

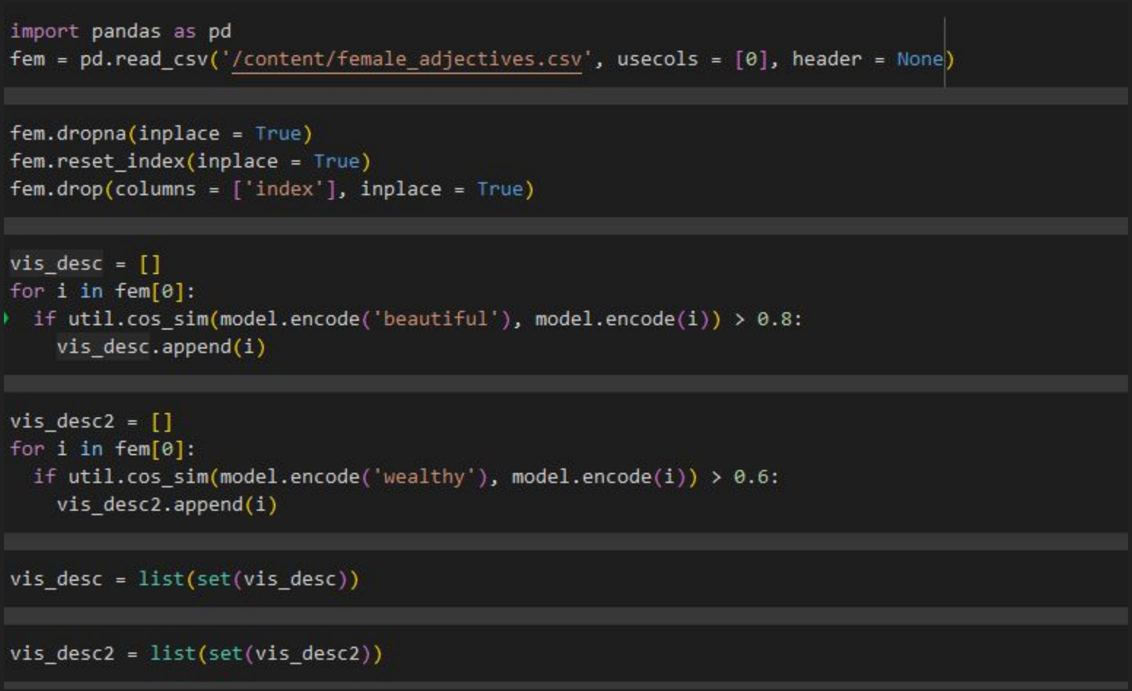
  Adjectives that matched this description in the ‘female adjectives’ list were collated into a list. Then the attributes are compared against this list, and an average of the semantic similarity is taken. If it is above a certain threshold, it is labelled as regressive. See the next slide for how this was done.

+ Presence of a profession
+ Centrality

  (determined by frequency of nsubjj or dobj (nominal subject and direct object) classifications by the SpaCy tagger)

These values are added up for the female characters, then a final ‘sexism’ index is computed to determine whether the synopsis was written with sexist undertones.

Female Adjectives collation using Pandas and Semantic analyzers



```python
import pandas as pd
fem = pd.read_csv('/content/female_adjectives.csv', usecols = [0], header = None)
```

```python
fem.dropna(inplace = True)
fem.reset_index(inplace = True)
fem.drop(columns = ['index'], inplace = True)
```

```python
vis_desc = []
for i in fem[0]:
  if util.cos_sim(model.encode('beautiful'), model.encode(i)) > 0.8:
    vis_desc.append(i)
```

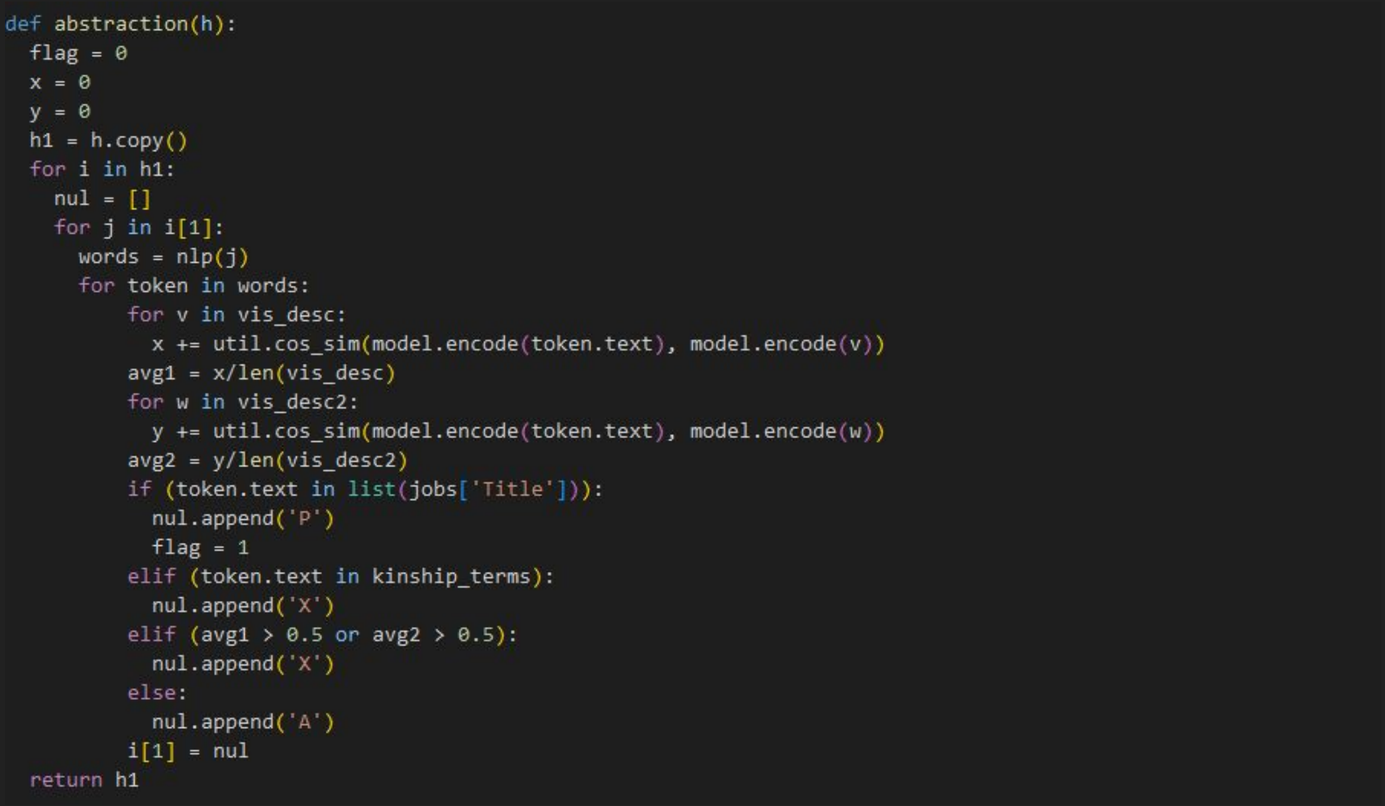
```python
vis_desc2 = []
for i in fem[0]:
  if util.cos_sim(model.encode('wealthy'), model.encode(i)) > 0.6:
    vis_desc2.append(i)
```

```python
vis_desc = list(set(vis_desc))
```

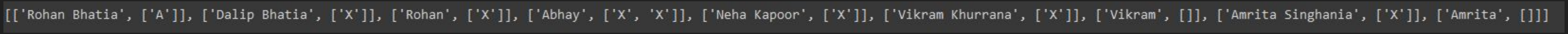
```python
vis_desc2 = list(set(vis_desc2))
```

The .csv file was read into Pandas, blank columns were dropped, then all the words that were determined to be semantically similar were collated into lists.

Abstraction



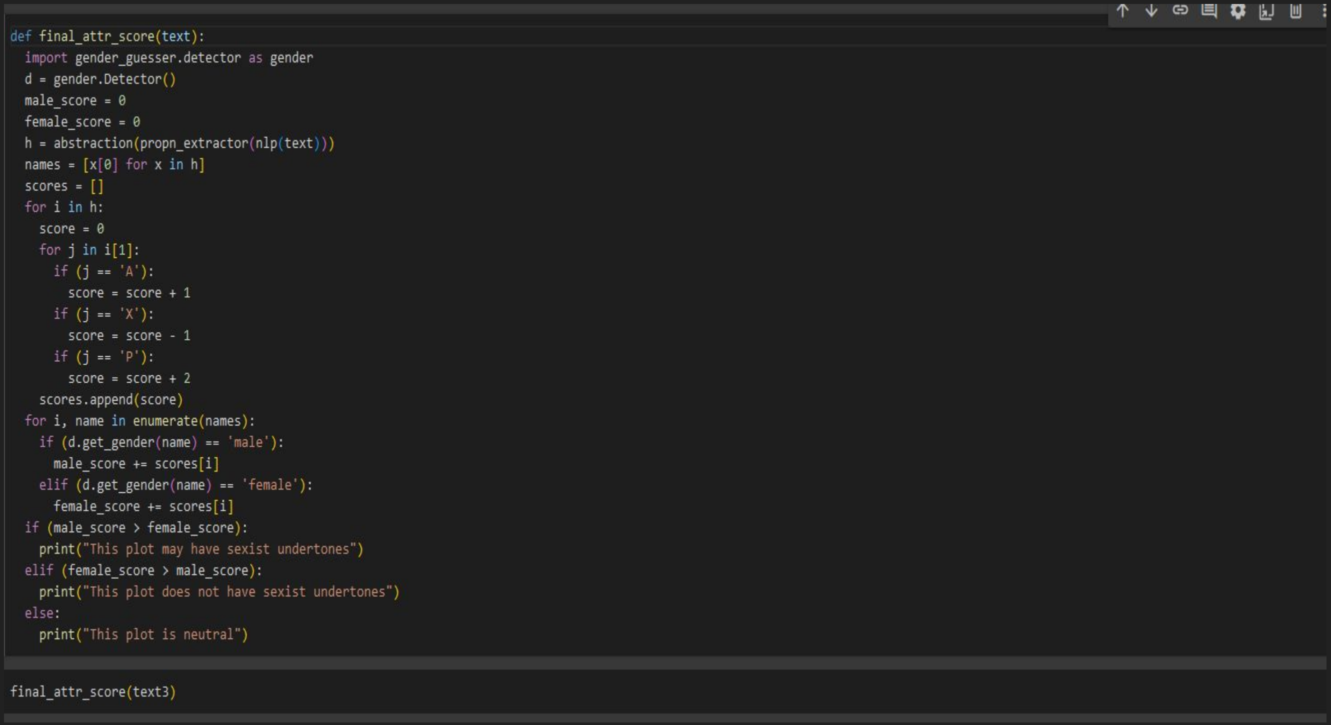
```
def abstraction(h):
  flag = 0
  x = 0
  y = 0
  h1 = h.copy()
  for i in h1:
    nul = []
    for j in i[1]:
      words = nlp(j)
      for token in words:
          for v in vis_desc:
            x += util.cos_sim(model.encode(token.text), model.encode(v))
          avg1 = x/len(vis_desc)
          for w in vis_desc2:
            y += util.cos_sim(model.encode(token.text), model.encode(w))
          avg2 = y/len(vis_desc2)
          if (token.text in list(jobs['Title'])):
            nul.append('P')
            flag = 1
          elif (token.text in kinship_terms):
            nul.append('X')
          elif (avg1 > 0.5 or avg2 > 0.5):
            nul.append('X')
          else:
            nul.append('A')
          i[1] = nul
  return h1
```



```
[['Rohan Bhatia', ['A']], ['Dalip Bhatia', ['X']], ['Rohan', ['X']], ['Abhay', ['X', 'X']], ['Neha Kapoor', ['X']], ['Vikram Khurrana', ['X']], ['Vikram', []], ['Amrita Singhania', ['X']], ['Amrita', []]]
```

Attributes are abstracted down into values.

Final score computation

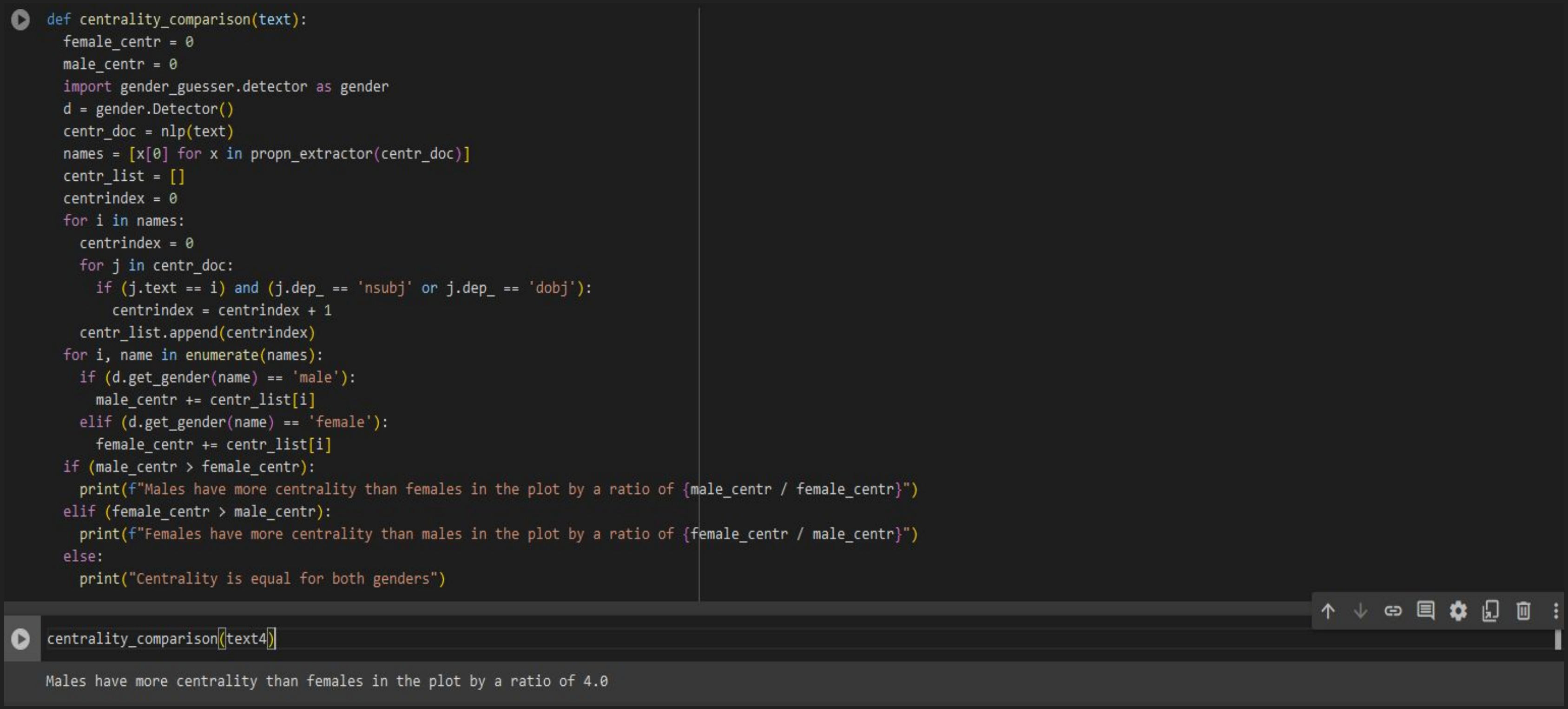


```
def final_attr_score(text):
  import gender_guesser.detector as gender
  d = gender.Detector()
  male_score = 0
  female_score = 0
  h = abstraction(propn_extractor(nlp(text)))
  names = [x[0] for x in h]
  scores = []
  for i in h:
    score = 0
    for j in i[1]:
      if (j == 'A'):
        score = score + 1
      if (j == 'X'):
        score = score - 1
      if (j == 'P'):
        score = score + 2
    scores.append(score)
  for i, name in enumerate(names):
    if (d.get_gender(name) == 'male'):
      male_score += scores[i]
    elif (d.get_gender(name) == 'female'):
      female_score += scores[i]
  if (male_score > female_score):
    print("This plot may have sexist undertones")
  elif (female_score > male_score):
    print("This plot does not have sexist undertones")
  else:
    print("This plot is neutral")
```

```
final_attr_score(text3)
```

A final score is computed by adding up the abstracted values of the male and female characters in the plot and comparing them.

Centrality comparison

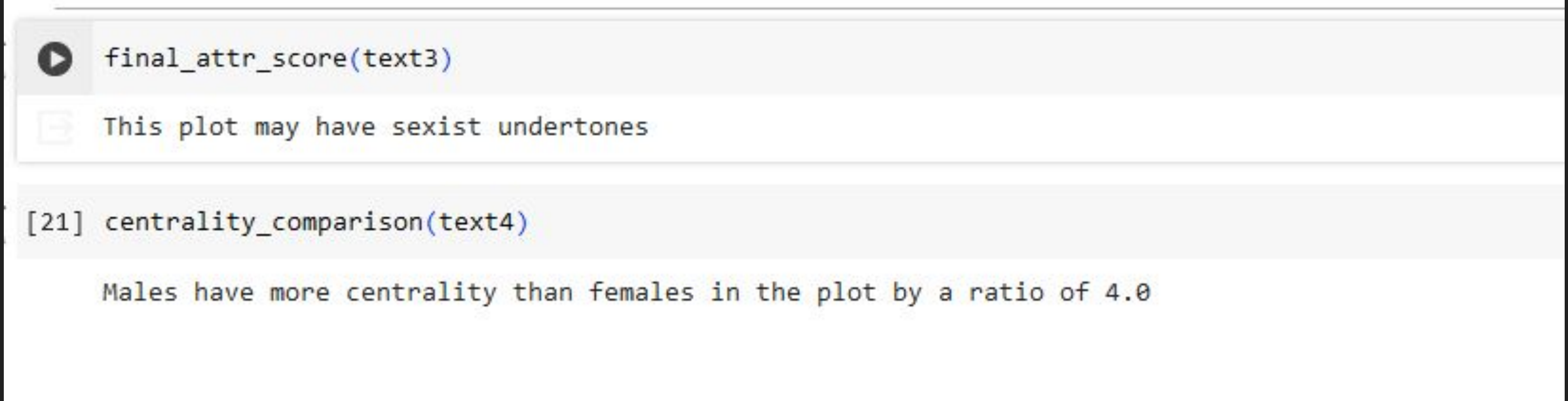


```
def centrality_comparison(text):
  female_centr = 0
  male_centr = 0
  import gender_guesser.detector as gender
  d = gender.Detector()
  centr_doc = nlp(text)
  names = [x[0] for x in propn_extractor(centr_doc)]
  centr_list = []
  centrindex = 0
  for i in names:
    centrindex = 0
    for j in centr_doc:
      if (j.text == i) and (j.dep_ == 'nsubj' or j.dep_ == 'dobj'):
        centrindex = centrindex + 1
    centr_list.append(centrindex)
  for i, name in enumerate(names):
    if (d.get_gender(name) == 'male'):
      male_centr += centr_list[i]
    elif (d.get_gender(name) == 'female'):
      female_centr += centr_list[i]
  if (male_centr > female_centr):
    print(f"Males have more centrality than females in the plot by a ratio of {male_centr / female_centr}")
  elif (female_centr > male_centr):
    print(f"Females have more centrality than males in the plot by a ratio of {female_centr / male_centr}")
  else:
    print("Centrality is equal for both genders")
```

```
centrality_comparison(text4)
```

```
Males have more centrality than females in the plot by a ratio of 4.0
```

Final conclusion



```
final_attr_score(text3)
```

```
This plot may have sexist undertones
```

```
[21] centrality_comparison(text4)
```

```
Males have more centrality than females in the plot by a ratio of 4.0
```

For the movie Jo Hum Chhahein (2011)