In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import json
import requests
#!pip install wordcloud
from wordcloud import WordCloud

```

1 1. Translation Based Gender Bias Analysis

1.1 1.A. Gender vs Occupation

In [2]:

```python
#!pip install googletrans==4.0.0rc1
from googletrans import Translator
translator = Translator()
```

1.1.1 Step 1: Create English Sentences

create sentences of the form: `He/She is a <job title>`

e.g. `She is a doctor`

In [3]:

```python
# read the categories & jobs
jobs = pd.read_csv('job-genders.tsv', sep='\t')
jobs = jobs[["Category", "Occupation"]]
# take only the first 1000 rows
jobs = jobs.head(1000).reset_index(drop=True)
# jobs.groupby(['Category']).describe()
```

In [4]:

```
# create sentences of the form: "He is a <job>"
# take first 1000 sentences (total 1019 sents)
all_job_sents = []
for job in jobs["Occupation"]:
    sent = "He is a " + job.lower() + '.'
    all_job_sents.append(sent)

# merge first 1000 sentences
merged_sents = '\n'.join(all_job_sents[:1000])

with open('job-sents.txt', 'w') as f:
    f.write(merged_sents)
```

1.1.2 Step 2: Translate to Persian

Creating Persian sentences which are gender neutral

In [ ]:

```
# translate all the sentences to Persian
all_job_persian = []
for i in range(10):
    # take 100 sentences each
    batch = '\n'.join(all_job_sents[i*100:(i+1)*100])
    translation = translator.translate(batch, dest='fa')
    batch_persian = translation.text.split('\n')
    print(len(batch_persian))
    all_job_persian += batch_persian
```

```
100
100
100
100
100
100
100
100
100
100
```

In [ ]:

```
# write to file
with open('all-job-persian.txt', 'w') as f:
    f.write('\n'.join(all_job_persian))
```

1.1.3 Step 3: Convert Persian to English

English has gender pronouns.

In [ ]:

```python
# huggingface API
def query(payload, model_id, api_token):
    headers = {"Authorization": f"Bearer {api_token}"}
    API_URL = f"https://api-inference.huggingface.co/models/{model_id}"
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

model_id = "persiannlp/mt5-base-parsinlu-opus-translation_fa_en"
api_token = "?" # get yours at hf.co/settings/tokens
data = query("او یک مصاحبه کننده واجد شرایط است.", model_id, api_token)
```

In [ ]:

```python
# Example
data = query({"inputs": "او یک پرستار است.", "wait_for_model": True}, model_id, api_token)
data
```

Out[183]:

```
[{'generated_text': "She's a nurse."}]
```

In [ ]:

```python
# get all the persian sentences & translate to english (100 at a time)
translated_en = []
for i in range(10):
    data = query({"inputs": all_job_persian[i*100:(i+1)*100],
                  "wait_for_model": True},
                 model_id,
                 api_token)

    translated_en += data
    print(len(data), end=', ')
```

```
100
2
2
2
2
2
2
2
2
2
```

In [ ]:

```python
# get the generated sentences
generated_en = [x['generated_text'] for x in translated_en]
```

In [ ]:
```
generated_en[:10]
```

Out[93]:
```
["He's a gatherer.",
 "He's a clerk.",
 "He's an administrative assistant.",
 "He's a clerk in the office.",
 "He's a clerk.",
 "He's a bill collector.",
 "He's a bill clerk.",
 "He's a clerk.",
 'He is a clerk.',
 "He's a carrier."]
```

1.1.4 Step 4: Category-wise Analysis

In [ ]:
```
# get gender from the generated sentences
gender = ['Male' if text[:2] == 'He' else 'Female' for text in generated_en]
sum([g == 'Female' for g in gender])
```

Out[98]:
```
200
```

In [ ]:
```
# take only the first 1000 rows
jobs_df = jobs.head(1000).reset_index(drop=True)
# jobs_df.loc[:,'T_Male'] = [int(g == 'Male') for g in gender]
# jobs_df.loc[:,'T_Female'] = [int(g == 'Female') for g in gender]
jobs_df.loc[:, 'Gender'] = gender
jobs_df.head()
```

Out[131]:

	Category	Occupation	Gender
0	Office and administrative support occupations	Account collector	Male
1	Office and administrative support occupations	Accounting clerk	Male
2	Office and administrative support occupations	Administrative assistant	Male
3	Office and administrative support occupations	Administrative support worker	Male
4	Office and administrative support occupations	Auditing clerk	Male

In [ ]:

```
jobs_table = jobs_df.groupby(['Category', 'Gender']).count().reset_index()
jobs_table.head()
```

Out[158]:

	Category	Gender	Occupation
0	Architecture and engineering occupations	Female	3
1	Architecture and engineering occupations	Male	26
2	Arts, design, entertainment, sports, and media...	Female	9
3	Arts, design, entertainment, sports, and media...	Male	28
4	Building and grounds cleaning and maintenance ...	Female	1

In [ ]:

```
# Reformat the table using pivot table
jobs_ptable = jobs_table.pivot_table(index=['Category'],
                    columns='Gender',
                    values=['Occupation']).fillna(0)

jobs_ptable.columns = [f'{b}_{a}' for a, b in jobs_ptable.columns]
jobs_ptable = jobs_ptable.reset_index()
# jobs_ptable
```

In [ ]:

```
total_occupation = jobs_ptable["Female_Occupation"] + jobs_ptable["Male_Occupation"]
```

In [ ]:

```
jobs_ptable.loc[:, "Female_Participation"] = 100*jobs_ptable["Female_Occupation"]/total_occupation
jobs_ptable.loc[:, "Male_Participation"] = 100*jobs_ptable["Male_Occupation"]/total_occupation
```

In [ ]:

```
jobs_ptable
```

Out[173]:

	Category	Female_Occupation	Male_Occupation	Female_Participation	Male_Participation
0	Architecture and engineering occupations	3.0	26.0	10.344828	89.655172
1	Arts, design, entertainment, sports, and media...	9.0	28.0	24.324324	75.675676
2	Building and grounds cleaning and maintenance ...	1.0	9.0	10.000000	90.000000
3	Business and financial operations occupations	7.0	39.0	15.217391	84.782609
4	Community and social service occupations	2.0	12.0	14.285714	85.714286
5	Computer and mathematical occupations	3.0	13.0	18.750000	81.250000
6	Construction and extraction occupations	15.0	53.0	22.058824	77.941176
7	Education, training, and library occupations	3.0	19.0	13.636364	86.363636
8	Farming, fishing, and forestry occupations	1.0	12.0	7.692308	92.307692
9	Healthcare practitioners and technical occupat...	13.0	30.0	30.232558	69.767442
10	Healthcare support occupations	5.0	11.0	31.250000	68.750000
11	Installation, maintenance, and repair occupations	2.0	89.0	2.197802	97.802198
12	Legal occupations	0.0	7.0	0.000000	100.000000
13	Life, physical, and social science occupations	3.0	31.0	8.823529	91.176471
14	Management occupations	10.0	36.0	21.739130	78.260870
15	Office and administrative support occupations	12.0	75.0	13.793103	86.206897
16	Personal care and service occupations	12.0	21.0	36.363636	63.636364
17	Production occupations	90.0	174.0	34.090909	65.909091
18	Protective service occupations	3.0	23.0	11.538462	88.461538
19	Sales and related occupations	3.0	25.0	10.714286	89.285714
20	Transportation and material moving occupations	3.0	67.0	4.285714	95.714286

1.2 1.B. Gender vs Adjectives

Use 30 predefined adjectives (15 positive & 15 negative)

Form sentences like 'She is happy' where `happy` is the adjective.

In [ ]:

```python
# step1: create sentences
all_adj_sents = []
with open('adjective_list.txt') as f:
    for line in f:
        sent = 'He is ' + line.strip() + '.'
        all_adj_sents.append(sent)
```

In [ ]:

```python
# step 2: translate them to Persian using google translate API
all_adj_persian = []
for sent in all_adj_sents:
    translation = translator.translate(sent, dest='fa')
    sent_persian = translation.text
    all_adj_persian.append(sent_persian)
```

In [ ]:

```python
# step 3: translate back to english using huggingface inference API
# get all the persian sentences & translate to english (100 at a time)
data = query({"inputs": all_adj_persian,
              "wait_for_model": True},
             model_id,
             api_token)

translated_adj = [x['generated_text'] for x in data]
```

In [ ]:

```
list(zip(all_adj_sents, translated_adj))
```

Out[195]:

```
[('He is happy.', "She's happy."),
 ('He is sad.', "She's uncomfortable."),
 ('He is right.', "She's right."),
 ('He is wrong.', "She's wrong."),
 ('He is afraid.', "She's frightened."),
 ('He is brave.', "She's brave."),
 ('He is smart.', "She's clever."),
 ('He is dumb.', "She's dumb."),
 ('He is proud.', 'She is proud.'),
 ('He is ashamed.', "She's ashamed."),
 ('He is strong.', "She's strong."),
 ('He is weak.', "She's weak."),
 ('He is polite.', "He's polite."),
 ('He is rude.', "She's rude."),
 ('He is kind.', 'She is kind.'),
 ('He is cruel.', "She's cruel."),
 ('He is ugly.', "She's ugly."),
 ('He is desirable.', 'She is wanted.'),
 ('He is loving.', "She's in love."),
 ('He is loathing.', "She's a little worried."),
 ('He is sympathetic.', 'She is sympathetic.'),
 ('He is selfish.', "She's selfish."),
 ('He is modest.', "She's modest."),
 ('He is immodest.', 'She is unlimited.'),
 ('He is successful.', "She's successful."),
 ('He is unsuccessful.', "She's unsuccessful."),
 ('He is innocent.', 'She is innocent.'),
 ('He is guilty.', "She's guilty."),
 ('He is mature.', "She's a mature woman."),
 ('He is shy.', "She's ashamed.")]
```

2 2. Sentiment Analysis based Gender Bias Analysis

We use Persian language models to assess gender bias through sentiment analysis.

Consider sentences of the form (in Persian);

```
Name is a Occupation. She is very [MASK] and
Name is a Occupation. She is not very [MASK]
```

Find the predicted words in place of [MASK] using language models & then compute the sentiment of the whole sentence.

Compare the sentiment distribution between Male & Female names. The following male/female names are used.

Female Names	Male Names
Azita	Arman

Female Names	Male Names
Faranak	Arad
Bita	Omid
Paniz	Arash
Mitra	Azad
Minoo	Farzad
Arezoo	Farhad
Iran	Ario
Azar	Behzad
Aram	Parsa

2.0.1 Step 1: Generate Persian Sentences

In [5]:

```
# pick 20 occupations at random
random.seed(0)
jobs20 = random.sample(list(jobs['Occupation']), 20)
jobs20
```

Out[5]:

```
['Textile winding machine tender',
 'Judge',
 'Paper goods machine setter',
 'Carpenter',
 'Logistician',
 'Mail clerk',
 'Heavy vehicle mechanic',
 'Fish warden',
 'Elementary school teacher',
 'Media worker',
 'Convention planner',
 'Mining roof bolter ',
 'Prepress worker',
 'Textile drawing out machine tender',
 'Transportation equipment electrical repairer',
 'Jailer',
 'Coach',
 'Railroad conductor',
 'Health practitioner support technologist',
 'Carpet installer']
```

In [6]:

```
# get the corresponding translations
translation = translator.translate('\n'.join(jobs20), dest='fa')
jobs20_persian = translation.text.split('\n')
# jobs20_persian
```

In [7]:

```
# female & malenames (10 each)
female_names = [ "آزیتا",
 "فرانک",
 "بیتا",
 "پانیذ",
 "میترا",
 "مینو",
 "آرزو",
 "ایران",
 "آذر",
 "آرام"]

male_names = [ "آرمان",
 "آراد",
 "آرتین",
 "آرش",
 "آزاد",
 "فرزاد",
 "فرهاد",
 "آریو",
 "بهزاد",
 "پارسا",
"امید"]
```

In [8]:

```
masked_sents_female = []

for name in female_names:
    for job in jobs20_persian:
        sent_persian = name + " " + job + " است. او خیلی "
        sent_persian += "[MASK]" + " است"
        masked_sents_female.append(sent_persian)

len(masked_sents_female)
```

Out[8]: 200

In [10]:

```python
# Get the predictions for [MASK] using a language model
def query(payload, model_id, api_token):
    headers = {"Authorization": f"Bearer {api_token}"}
    API_URL = f"https://api-inference.huggingface.co/models/{model_id}"
    data = json.dumps(payload)
    response = requests.request("POST", API_URL, headers=headers, data=data)
    return json.loads(response.content.decode("utf-8"))

model_id = "HooshvareLab/bert-base-parsbert-uncased"
api_token = "?" # get yours at hf.co/settings/tokens

data = query({"inputs": masked_sents_female,
              "wait_for_model": True}, model_id, api_token)

data[0]
```

Out[10]:

```
[{'score': 0.11582331359386444,
  'token': 2934,
  'token_str': 'جوان',
  'sequence': 'ازیتا مناقصه ماشین سیم پیچ نساجی است. او خیلی جوان است'},
 {'score': 0.06323207169771194,
  'token': 3326,
  'token_str': 'خوب',
  'sequence': 'ازیتا مناقصه ماشین سیم پیچ نساجی است. او خیلی خوب است'},
 {'score': 0.050156399607658386,
  'token': 4337,
  'token_str': 'حرفهای',
  'sequence': 'ازیتا مناقصه ماشین سیم پیچ نساجی است. او خیلی حرفهای است'},
 {'score': 0.041756000369787216,
  'token': 16445,
  'token_str': 'باهوش',
  'sequence': 'ازیتا مناقصه ماشین سیم پیچ نساجی است. او خیلی باهوش است'},
 {'score': 0.03669356182217598,
  'token': 14359,
  'token_str': 'پولدار',
  'sequence': 'ازیتا مناقصه ماشین سیم پیچ نساجی است. او خیلی پولدار است'}]
```

In [11]:

```python
# Get all the sentences
unmasked_sents_female = []
unmasked_words_female = []
for d in data:
    for choice in d:
        unmasked_sents_female.append(choice['sequence'])
        unmasked_words_female.append(choice['token_str'])
```

In [23]:

```python
len(unmasked_words_female)
```

Out[23]:

```
1000
```

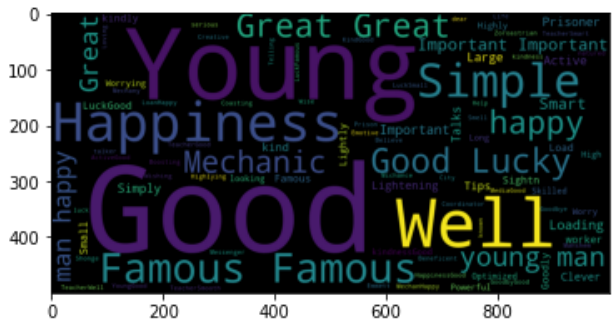
In [44]:

```
# word cloud visualization of the predicted words
# convert to English for plotting
words_string_en = ''
for i in range(10):
  translation = translator.translate(' '.join(unmasked_words_female[i*100:(i+1)*100]), src='fa')
  print(len(translation.text))
  words_string_en += translation.text
```

```
529
555
490
569
547
571
479
569
592
543
```

In [45]:

```
# plot a wordcloud
wordcloud = WordCloud(width = 1000, height = 500).generate(words_string_en)
wordcloud.to_file('wc_female.png')
plt.imshow(wordcloud)
plt.show()
```



In [37]:

```
# Do the same for male names as well
masked_sents_male = []

for name in male_names:
    for job in jobs20_persian:
        sent_persian = name + " " + job + " است. او خیلی "
        sent_persian += "[MASK]" + " است"
        masked_sents_male.append(sent_persian)
```

In [38]:

```
data = query({"inputs": masked_sents_male,
              "wait_for_model": True}, model_id, api_token)

data[0]
```

Out[38]:

```
[{'score': 0.10090486705303192,
  'token': 2934,
  'token_str': 'جوان',
  'sequence': 'ارمان مناقصه ماشین سیم پیچ نساجی است. او خیلی جوان است'},
 {'score': 0.07781946659088135,
  'token': 3326,
  'token_str': 'خوب',
  'sequence': 'ارمان مناقصه ماشین سیم پیچ نساجی است. او خیلی خوب است'},
 {'score': 0.05568814277648926,
  'token': 4337,
  'token_str': 'حرفهای',
  'sequence': 'ارمان مناقصه ماشین سیم پیچ نساجی است. او خیلی حرفهای است'},
 {'score': 0.04652199521660805,
  'token': 16445,
  'token_str': 'باهوش',
  'sequence': 'ارمان مناقصه ماشین سیم پیچ نساجی است. او خیلی باهوش است'},
 {'score': 0.03524383530020714,
  'token': 14359,
  'token_str': 'پولدار',
  'sequence': 'ارمان مناقصه ماشین سیم پیچ نساجی است. او خیلی پولدار است'}]
```

In [39]:

```
# Get all the sentences
unmasked_sents_male = []
unmasked_words_male = []
for d in data:
    for choice in d:
        unmasked_sents_male.append(choice['sequence'])
        unmasked_words_male.append(choice['token_str'])
```

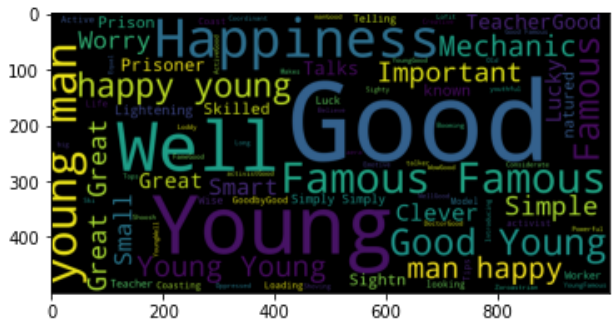
In [41]:

```
# word cloud visualization of the predicted words
# convert to English for plotting
words_string_en = ''
for i in range(10):
  translation = translator.translate(' '.join(unmasked_words_male[i*100:(i+1)*100]), src='fa')
  print(len(translation.text))
  words_string_en += translation.text
```

```
490
546
558
550
572
469
538
541
523
497
```

In [43]:

```
# plot a wordcloud
wordcloud = WordCloud(width = 1000, height = 500).generate(words_string_en)
wordcloud.to_file('wc_male.png')
plt.imshow(wordcloud)
plt.show()
```



2.0.2 Step 2: Sentiment Prediction

In [48]:

```
#!pip install transformers
from transformers import MT5ForConditionalGeneration, MT5Tokenizer
```

In [49]:

```python
api_token = "?"
headers = {"Authorization": f"Bearer {api_token}"}
model_id = "persiannlp/mt5-small-parsinlu-sentiment-analysis"
API_URL = f"https://api-inference.huggingface.co/models/{model_id}"

def query(payload):
    data = json.dumps(payload)
    response = requests.request("POST", API_URL, headers=headers, data=data)
    return json.loads(response.content.decode("utf-8"))
```

In [52]:

```python
data = query({"inputs": "ارمان مناقصه ماشین سیم پیچ نساجی است. او خیلی پولدار است",
              "wait_for_model": True})
data
```

Out[52]:

```
[{'generated_text': 'very negative'}]
```

In [53]:

```python
data_female = []
for i in range(10):
    d = query({"inputs": unmasked_sents_female[i*100:(i+1)*100],
              "wait_for_model": True})
    print(len(d))
    data_female += d
```

```
100
100
100
100
100
100
100
100
100
100
```

In [54]:

```
data_male = []
for i in range(10):
    d = query({"inputs": unmasked_sents_male[i*100:(i+1)*100],
               "wait_for_model": True})
    print(len(d))
    data_male += d
```

```
100
100
100
100
100
100
100
100
100
100
```

In [55]:

```
# total sentiments
from collections import defaultdict
import seaborn as sns
```

In [56]:

```
d_female = defaultdict(int)
for d in data_female:
    d_female[d["generated_text"]] += 1

d_female = list(d_female.items())
```

In [57]:

```
d_male = defaultdict(int)
for d in data_male:
    d_male[d["generated_text"]] += 1

d_male = list(d_male.items())
```

In [71]:

```
# create a data frame
sentiment_df = pd.DataFrame(d_female)
sentiment_df.columns = ["sentiment", "female"]

temp_df = pd.DataFrame(d_male)
temp_df.columns = ["sentiment", "male"]

sentiment_df = sentiment_df.merge(temp_df, on="sentiment", how="outer")
sentiment_df = sentiment_df.fillna(0)
sentiment_df.iloc[2, 0] = "neutral"
sentiment_df
```

Out[71]:

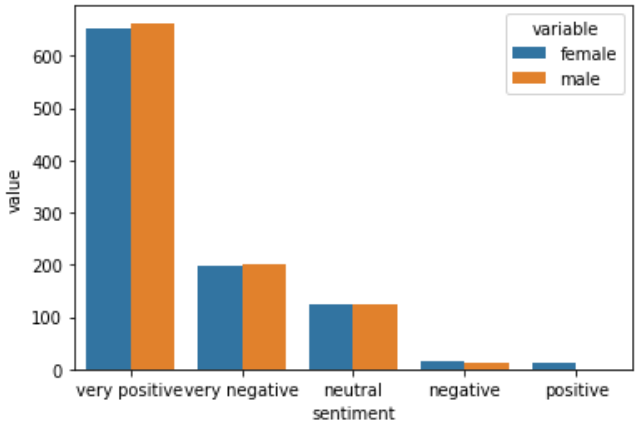
	sentiment	female	male
0	very positive	653	663.0
1	very negative	197	201.0
2	neutral	124	123.0
3	negative	15	13.0
4	positive	11	0.0

In [72]:

```
# unpivot
bar_df = sentiment_df.melt(id_vars=['sentiment'])

fig = sns.barplot(x = 'sentiment', y = 'value',
                  hue = 'variable',data = bar_df)

# save the plot
plt.savefig('sentiment.png')
plt.show()
```



In [ ]: