Group 15 TM names :

- Ahmed : 23PGAI0120
- Akash Deshwani: 23PGAI0035
- Harshada Suresh Jadhav: 23PGAI0101
- Rohan Mehta: 23PGAI0001

Global Award name list

In [1]:
```
AWARDS = ['best motion picture - drama', 'best motion picture - musical or comedy', 'b
```

Required Import Files

In [2]:
```
!pip install numpy
!pip install pandas
!pip install spacy
!pip install collections
!pip install python-Levenshtein
!python -m spacy download en_core_web_md
```

```
Requirement already satisfied: numpy in c:\users\coola\anaconda3\lib\site-packages
(1.21.5)
Requirement already satisfied: pandas in c:\users\coola\anaconda3\lib\site-packages
(1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\coola\anaconda3\lib\site-pack
ages (from pandas) (2021.3)
Requirement already satisfied: numpy>=1.18.5 in c:\users\coola\anaconda3\lib\site-pac
kages (from pandas) (1.21.5)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\coola\anaconda3\lib
\site-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\coola\anaconda3\lib\site-packages
(from python-dateutil>=2.8.1->pandas) (1.16.0)
Requirement already satisfied: spacy in c:\users\coola\anaconda3\lib\site-packages
(3.4.2)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4 in c:\users\cool
a\anaconda3\lib\site-packages (from spacy) (1.10.2)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in c:\users\coola\anaconda3\li
b\site-packages (from spacy) (2.0.8)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy) (2.4.5)
Requirement already satisfied: thinc<8.2.0,>=8.1.0 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy) (8.1.5)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in c:\users\coola\anaconda
3\lib\site-packages (from spacy) (1.0.3)
Requirement already satisfied: jinja2 in c:\users\coola\anaconda3\lib\site-packages
(from spacy) (2.11.3)
Requirement already satisfied: setuptools in c:\users\coola\anaconda3\lib\site-packag
es (from spacy) (61.2.0)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy) (2.0.7)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in c:\users\coola\anaconda3\li
b\site-packages (from spacy) (2.27.1)
Requirement already satisfied: pathy>=0.3.5 in c:\users\coola\anaconda3\lib\site-pack
ages (from spacy) (0.6.2)
Requirement already satisfied: packaging>=20.0 in c:\users\coola\anaconda3\lib\site-p
ackages (from spacy) (21.3)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.10 in c:\users\coola\anaconda
3\lib\site-packages (from spacy) (3.0.10)
Requirement already satisfied: wasabi<1.1.0,>=0.9.1 in c:\users\coola\anaconda3\lib\s
ite-packages (from spacy) (0.10.1)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in c:\users\coola\anaconda3\lib
\site-packages (from spacy) (3.0.8)
Requirement already satisfied: typer<0.5.0,>=0.3.0 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy) (0.4.2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy) (4.64.0)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in c:\users\coola\anaconda3
\lib\site-packages (from spacy) (1.0.9)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in c:\users\coola\anaconda3\li
b\site-packages (from spacy) (3.3.0)
Requirement already satisfied: numpy>=1.15.0 in c:\users\coola\anaconda3\lib\site-pac
kages (from spacy) (1.21.5)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\coola\anaconda3\l
ib\site-packages (from packaging>=20.0->spacy) (3.0.4)
Requirement already satisfied: smart-open<6.0.0,>=5.2.1 in c:\users\coola\anaconda3\l
ib\site-packages (from pathy>=0.3.5->spacy) (5.2.1)
Requirement already satisfied: typing-extensions>=4.1.0 in c:\users\coola\anaconda3\l
ib\site-packages (from pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4->spacy) (4.1.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\coola\anaconda3\lib\sit
e-packages (from requests<3.0.0,>=2.13.0->spacy) (2021.10.8)
```

```
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\coola\anaconda3
\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\coola\anaconda3\lib
\site-packages (from requests<3.0.0,>=2.13.0->spacy) (1.26.9)
Requirement already satisfied: idna<4,>=2.5 in c:\users\coola\anaconda3\lib\site-pack
ages (from requests<3.0.0,>=2.13.0->spacy) (3.3)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in c:\users\coola\anaconda3\lib\sit
e-packages (from thinc<8.2.0,>=8.1.0->spacy) (0.7.9)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in c:\users\coola\anaconda3\l
ib\site-packages (from thinc<8.2.0,>=8.1.0->spacy) (0.0.3)
Requirement already satisfied: colorama in c:\users\coola\anaconda3\lib\site-packages
(from tqdm<5.0.0,>=4.38.0->spacy) (0.4.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in c:\users\coola\anaconda3\lib\si
te-packages (from typer<0.5.0,>=0.3.0->spacy) (8.0.4)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\coola\anaconda3\lib\site-
packages (from jinja2->spacy) (2.0.1)
ERROR: Could not find a version that satisfies the requirement collections (from vers
ions: none)
ERROR: No matching distribution found for collections
```

```
Requirement already satisfied: python-Levenshtein in c:\users\coola\anaconda3\lib\sit
e-packages (0.20.8)
Requirement already satisfied: Levenshtein==0.20.8 in c:\users\coola\anaconda3\lib\si
te-packages (from python-Levenshtein) (0.20.8)
Requirement already satisfied: rapidfuzz<3.0.0,>=2.3.0 in c:\users\coola\anaconda3\li
b\site-packages (from Levenshtein==0.20.8->python-Levenshtein) (2.13.2)
Collecting en-core-web-md==3.4.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web
_md-3.4.1/en_core_web_md-3.4.1-py3-none-any.whl (42.8 MB)
Requirement already satisfied: spacy<3.5.0,>=3.4.0 in c:\users\coola\anaconda3\lib\si
te-packages (from en-core-web-md==3.4.1) (3.4.2)
Requirement already satisfied: wasabi<1.1.0,>=0.9.1 in c:\users\coola\anaconda3\lib\s
ite-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (0.10.1)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4 in c:\users\cool
a\anaconda3\lib\site-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (1.1
0.2)
Requirement already satisfied: numpy>=1.15.0 in c:\users\coola\anaconda3\lib\site-pac
kages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (1.21.5)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in c:\users\coola\anaconda3\li
b\site-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (3.3.0)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in c:\users\coola\anaconda
3\lib\site-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (1.0.3)
Requirement already satisfied: packaging>=20.0 in c:\users\coola\anaconda3\lib\site-p
ackages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (21.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (2.4.5)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in c:\users\coola\anaconda3\li
b\site-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (2.0.8)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.10 in c:\users\coola\anaconda
3\lib\site-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (3.0.10)
Requirement already satisfied: jinja2 in c:\users\coola\anaconda3\lib\site-packages
(from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (2.11.3)
Requirement already satisfied: setuptools in c:\users\coola\anaconda3\lib\site-packag
es (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (61.2.0)
Requirement already satisfied: pathy>=0.3.5 in c:\users\coola\anaconda3\lib\site-pack
ages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (0.6.2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (4.64.0)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (2.0.7)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in c:\users\coola\anaconda3\li
b\site-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (2.27.1)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in c:\users\coola\anaconda3
\lib\site-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (1.0.9)
Requirement already satisfied: thinc<8.2.0,>=8.1.0 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (8.1.5)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in c:\users\coola\anaconda3\lib
\site-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (3.0.8)
Requirement already satisfied: typer<0.5.0,>=0.3.0 in c:\users\coola\anaconda3\lib\si
te-packages (from spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (0.4.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\coola\anaconda3\l
ib\site-packages (from packaging>=20.0->spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1)
(3.0.4)
Requirement already satisfied: smart-open<6.0.0,>=5.2.1 in c:\users\coola\anaconda3\l
ib\site-packages (from pathy>=0.3.5->spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (5.
2.1)
Requirement already satisfied: typing-extensions>=4.1.0 in c:\users\coola\anaconda3\l
ib\site-packages (from pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4->spacy<3.5.0,>=3.4.0->en
-core-web-md==3.4.1) (4.1.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\coola\anaconda3\lib\sit
```

```
e-packages (from requests<3.0.0,>=2.13.0->spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1)
(2021.10.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\coola\anaconda3\lib
\site-packages (from requests<3.0.0,>=2.13.0->spacy<3.5.0,>=3.4.0->en-core-web-md==3.
4.1) (1.26.9)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\coola\anaconda3
\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy<3.5.0,>=3.4.0->en-core-web-md
==3.4.1) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\coola\anaconda3\lib\site-pack
ages (from requests<3.0.0,>=2.13.0->spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (3.3)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in c:\users\coola\anaconda3\l
ib\site-packages (from thinc<8.2.0,>=8.1.0->spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.
1) (0.0.3)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in c:\users\coola\anaconda3\lib\sit
e-packages (from thinc<8.2.0,>=8.1.0->spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (0.
7.9)
Requirement already satisfied: colorama in c:\users\coola\anaconda3\lib\site-packages
(from tqdm<5.0.0,>=4.38.0->spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (0.4.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in c:\users\coola\anaconda3\lib\si
te-packages (from typer<0.5.0,>=0.3.0->spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1)
(8.0.4)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\coola\anaconda3\lib\site-
packages (from jinja2->spacy<3.5.0,>=3.4.0->en-core-web-md==3.4.1) (2.0.1)
[+] Download and installation successful
You can now load the package via spacy.load('en_core_web_md')
```

In [3]:
```
import json
import pandas as pd
import spacy
from collections import defaultdict
import Levenshtein
from spacy.matcher import Matcher
from spacy.tokens import Span
from string import punctuation

nlp = spacy.load('en_core_web_md')
```

In [ ]:
```
```

In [4]:
```
award_names = AWARDS
```

In [5]:
```
def map_to_a_award(tweet):
    awards = {}
    tweet = tweet.lower()
    for award in award_names:
        split_award = award.split()
        award_dist = 0
        split_tweet = tweet.split()
        for word in tweet.split():
            min_val = float('inf')
            for word2 in award.split():
                min_val = min(min_val,Levenshtein.distance(word, word2))
            award_dist += min_val
        awards[award] = award_dist

    val = min(awards.values())
    res = [key for key in awards if awards[key] == val]
    return res
```

In [6]:
```
def find_mapping(matches, doc):
    winner_len = 0
    award_len = 0
    winner = ''
    award = ''
    for match_id, start, end in matches:
        string_id = nlp.vocab.strings[match_id]
        span = doc[start:end]
        span_len = len(span.text)
        if string_id == 'Person' and len(span)<= 2:
            if winner_len < span_len:
                winner = span.text
                winner_len = span_len
        # if string_id == 'Film':
        #     if winner_len > span_len:
        #         winner = span.text
        #         winner_len = span_len
        elif string_id == 'Award':
            if award_len < span_len:
                award = span.text
                award_len = span_len
    winner = str(winner).strip(punctuation).strip()
    award = str(award).strip(punctuation).strip()
    key = (winner.lower(),award)
    return key
```

In [7]:
```
def distance(t1,r1):
    t = t1.split()
    r = r1.split()
    if len(t) >= len(r):
        long = t
        short = r
        ret = t1
    else:
        long = r
        short = t
        ret = r1
    dist = 0
    for word in short:
        min_val = float('inf')
        for word2 in long:
            min_val = min(min_val,Levenshtein.distance(word, word2))
        dist += min_val
    return dist,ret
```

In [8]:
```
def commonwords(a,b):
    return len(set(a.split()).intersection(set(b.split())))
```

In [9]:
```
def get_hosts(df):

    host_df = df[df.str.contains('host|hosts|hostess|hosted|hosting', case = False)]
    df = df[~df.str.contains('think|thinking|should|maybe', case = False)]
    df = df.str.replace('#GoldenGlobes|golden|globes|globe', "", case = False)
    if host_df.size > 3000:
        host_df = host_df.sample(3000)

    hosts = {}
    hosts = defaultdict(lambda: 1, hosts)
    for i, value in host_df.iteritems():
```

```
        for entity in nlp(value).ents:
            if entity.label_ == 'PERSON':
                hosts[entity.text] = hosts[entity.text] +1

    hosts = sorted(hosts.items(), key=lambda item: item[1], reverse = True)

    top30 = hosts[:30]
    for name in range(5):
        first_name = top30[name][0].split()[0]
        for other in range(29-name):
            curr_name = top30[other+name+1][0]
            if first_name in curr_name:
                count = top30[name][1] + top30[other+name+1][1]
                top30[name] = (top30[name][0], count)

    final30 = sorted(top30, key=lambda item: item[1], reverse = True)
    hosts = (final30[0][0], final30[1][0])

    return hosts
```

In [10]:
```
# hosts  = get_hosts()
# print(hosts)
```

In [11]:
```
def get_awards(df):

    df = df[~df.str.contains('think|should|maybe', case = False)]
    df = df[df.str.contains('best', case = False)]
    df = df.str.replace('http\S+|www.\S+', '', case=False,regex=True)
    df = df.str.replace('TV|tv', "television", case = False, regex = True)
    best_df = df.str.replace('#GoldenGlobes|#GoldenGlobe|golden|globes|globes', "", ca

    if best_df.shape[0] >5000:
        best_df = best_df.sample(n=5000)
    best_df.shape[0]

    pattern = [{"LOWER":'best'}, {"DEP":{"IN":['compound','nmod','dobj','prep','det','
    pattern2 = [{"ENT_TYPE":{"IN":['PERSON','WORK_OF_ART']},'OP':'+'}]

    matcher = Matcher(nlp.vocab)
    matcher.add('Winner', [pattern])
    matcher2 = Matcher(nlp.vocab)
    matcher2.add('Deleter',[pattern2])

    awards = {}
    awards = defaultdict(lambda: 1, awards)
    for i, text in best_df.iteritems():
        doc = nlp(text)
        matches = matcher(doc)
        if len(matches) != 0:
            span = doc[matches[-1][1]:matches[-1][2]]
            awards[str(span).strip(punctuation).strip()] = awards[str(span).strip(punc

    for key in list(awards):
        doc = nlp(key)
    matches = matcher2(doc)
    if len(matches) != 0:
        awards.pop(key)
    awards = sorted(awards.items(), key=lambda item: item[1], reverse = True)
```

```
    new_awards = []
    for i in awards:
        if i[1] > 2:
            new_awards.append(i[0].lower())

    final_list = []
    final_list.append(new_awards[0])

    for awrd in new_awards[1:]:
        similar = False
        for i,ans in enumerate(final_list):
            dist,long = distance(awrd,ans)
            if dist <= 1:
                similar = True
                final_list[i] = long
        if (not similar):
            final_list.append(awrd)

    awards = final_list[0:26]
    return awards
```

In [12]:

```
def get_winner(df):

    df = df[df.str.contains('won|win|goes to|for best', case = False)]
    # present_df = df[df.str.contains('won|win|goes to|for best', case = False)]
    df = df[~df.str.contains('think|should|maybe|RT @', case = False)]
    df = df[df.str.contains('best', case = False)]
    df = df.str.replace('http\S+|www.\S+', '', case=False,regex=True)
    df = df.str.replace('TV|tv', "television", case = False, regex = True)
    wins_df = df.str.replace('#GoldenGlobes|#GoldenGlobe|golden|globes|globes', "", ca
    if wins_df.size > 8000:
        wins_df = wins_df.sample(8000)


    person_pattern = [{"ENT_TYPE": "PERSON", 'OP':'+'}]
    film_pattern = [{"ENT_TYPE": 'ORG', 'OP':'+'}]

    award_pattern = [{"LOWER":'best'}, {"DEP":{"IN":['compound','nmod','dobj','prep','

    matcher = Matcher(nlp.vocab)
    matcher.add('Person',[person_pattern])
    matcher.add('Award',[award_pattern])
    matcher.add('Film',[film_pattern])

    winners = {}
    winners = defaultdict(lambda: 0, winners)
    for i, text in wins_df.iteritems():
        doc = nlp(text)
        matches = matcher(doc)
        if len(matches) != 0:
            key = find_mapping(matches ,doc)
            if (key[0] != '' and key[1] != ''):
                winners[key] = winners[key] + 1

    sorte_d = sorted(winners.items(), key=lambda item: item[1], reverse = True)

    final_ans = {}
    for award in award_names:
        sub_dict = {}
```

```
        sub_dict = defaultdict(lambda: 0, sub_dict)
        final_ans[award] = sub_dict
    for pair,count in winners.items():
        award = pair[1]
        person = pair[0]
        possible_awards = map_to_a_award(award)
        for ard in possible_awards:
            final_ans[ard][person] = final_ans[ard][person] + count
    for ans in final_ans:
        if len(final_ans[ans].items()) > 0:
            final_ans[ans] = max(final_ans[ans], key=final_ans[ans].get)
        else:
            final_ans[ans] = ''

    winners = final_ans
    return winners
```

In [13]:
```
# win = get_winner()
# print(win)
```

In [14]:
```
def get_presenters(df):

    df = df[df.str.contains('present|announc|introduc', case = False)]
    df = df[~df.str.contains('think|should|maybe|RT @', case = False)]
    # df = df[df.str.contains('best', case = False)]
    df = df.str.replace('http\S+|www.\S+', '', case=False,regex=True)
    df = df.str.replace('TV|tv', "television", case = False, regex = True)
    wins_df = df.str.replace('#GoldenGlobes|#GoldenGlobe|golden|globes|globes', "", ca
    if wins_df.size > 8000:
        wins_df = wins_df.sample(8000)

    # person_pattern = [{"ENT_TYPE": "PERSON", 'OP':'+'}]
    # award_pattern = [{"LOWER":'best'}, {"DEP":{"IN":['compound','nmod','dobj','prep
    # pattern1 = [{"ENT_TYPE": "PERSON", 'OP':'+'}, {"LEMMA": "win"}, {"ORTH":'Best'},
    # pattern2 = [{"ENT_TYPE": "WORK_OF_ART", 'OP':'+'}, {"LEMMA": "win"}, {"ORTH":'Be
    # pattern3 = [{"ENT_TYPE": "PERSON", 'OP':'+'}, {"LEMMA": "win"}, {"ORTH":'Best'},
    # pattern4 = [{"ENT_TYPE": "WORK_OF_ART", 'OP':'+'}, {"LEMMA": "win"}, {"ORTH":'Be
    # pattern5 = [{"ORTH":'Best'}, {"ENT_TYPE": "WORK_OF_ART", 'OP':'+'},{"TEXT": "goe
    # pattern6 = [{"ORTH":'Best'}, {"DEP": 'compound', 'OP':'+'}, {"POS":'NOUN', 'OP':

    person_pattern = [{"ENT_TYPE": "PERSON", 'OP':'+'}]
    film_pattern = [{"ENT_TYPE": 'ORG', 'OP':'+'}]

    award_pattern = [{"LOWER":'best'}, {"DEP":{"IN":['compound','nmod','dobj','prep','

    matcher = Matcher(nlp.vocab)
    # matcher.add('Person',[person_pattern])
    # matcher.add('Award',[award_pattern])
    matcher.add('Person',[person_pattern])
    matcher.add('Award',[award_pattern])
    matcher.add('Film',[film_pattern])

    presenters = {}
    presenters = defaultdict(lambda: 0, presenters)
    for i, text in wins_df.iteritems():
        doc = nlp(text)
        matches = matcher(doc)
        if len(matches) != 0:
            key = find_mapping(matches ,doc)
```

```
            if (key[0] != '' and key[1] != ''):
                presenters[key] = presenters[key] +1


    presenters = dict(presenters)
    sort_d = sorted(presenters.items(), key=lambda item: item[1], reverse = True)

    final_ans = {}
    for award in award_names:
        sub_dict = {}
        sub_dict = defaultdict(lambda: 0, sub_dict)
        final_ans[award] = sub_dict
    for pair,count in presenters.items():
        award = pair[1]
        person = pair[0]
        possible_awards = map_to_a_award(award)
        for ard in possible_awards:
            final_ans[ard][person] = final_ans[ard][person] + count
    for ans in final_ans:
        if len(final_ans[ans].items()) > 0:
            final_ans[ans] = max(final_ans[ans], key=final_ans[ans].get)
        else:
            final_ans[ans] = ''
        # if len(final_ans[ans].items()) > 0:
        #     m = max(final_ans[ans], key=final_ans[ans].get)
        #     final_ans[ans].pop(m)
        #     m2 = max(final_ans[ans], key=final_ans[ans].get)

        #     while commonwords(m, m2) != 0:
        #         final_ans[ans].pop(m2)
        #         try:
        #             m2 = max(final_ans[ans], key=final_ans[ans].get)
        #         except:
        #             m2 = ''

        #     final_ans[ans] = [m,m2]
        # else:
        #     final_ans[ans] = []

    presenters = final_ans
    return presenters
```

In [15]:
```
# presenter = get_presenters()
# print(presenter)
```

In [16]:
```
def winner_presenter_nominees_helper_1(df): # This function is to help the nominees be
    winners = get_winner(df)
    presenters = get_presenters(df)

    new_award_names = [0]*len(award_names)
    for i in range(len(award_names)):
        new_award_names[i] = award_names[i].replace(" in ", " ").replace(" a ", " ").r

    awardnominees = {}
    awardnominees = defaultdict(lambda: 1, awardnominees)

    for award in award_names:
        awardnominees[award] = []
```

```
    df = df[~df.str.contains('RT @', case = False)]
    df = df[df.str.contains('Best', case = True)]
    nominees_df = df.str.replace('#GoldenGlobes|golden|globes|globe', "", case = False
    # nominees_df = df[df.str.contains('nominated', case = False)]
    if nominees_df.size > 6500:
        nominees_df = nominees_df.sample(6500)

    for i, text in nominees_df.iteritems():
        for j, award in enumerate(new_award_names):
            awardlst = award.split()
            awardlst2 = award.replace("television ", "").split()
            if all(x in text.lower() for x in awardlst) or all(x in text.lower() for x
                for ent in nlp(text).ents:
                    if (ent.label_ == "PERSON" or ent.label == "WORK_OF_ART") and '#'
                        awardnominees[list(awardnominees)[j]].append(ent.text)

    for lst in awardnominees:
        dd = defaultdict( int )
        for word in awardnominees[lst]:
            dd[word] += 1

        awardnominees[lst] = dd

    nominees_final = {}
    for i, each in enumerate(awardnominees):
        nominees_final[each] = [winners[each]]

        noms = sorted(awardnominees[each].items(), key=lambda item: item[1], reverse =

        if noms == []:
            continue

        for j, (name, num) in enumerate(noms):
            name = name.lower()
            # if j == 0:
            #     nominees_final[each].append(name)
            # else:
            common = 1
            for inlst in nominees_final[each]:
                if commonwords(inlst, name) != 0:
                    common = 0

            if common == 1 and len(nominees_final[each]) < 5 and name not in presenter
                nominees_final[each].append(name)

    return winners, presenters, nominees_final
```

In [17]:
```
# [winners, presenters, nominees] = winner_presenter_nominees_helper_1()
# print(winners)
# print(presenters)
# print(nominees)
```

In [18]:
```
def main():
    df = pd.read_json('gg2015.json')
    df = df['text']
    json_2015 = {}

    [winners, presenters, nominees] = winner_presenter_nominees_helper_1(df)
    awards = get_awards(df)
```

```
    hosts = get_hosts(df)

    print("Hosts: ", *hosts, sep=", ")
    print("Award: ", awards)
    print("Presenters: ", *presenters, sep=", ")
    print("Nominees: ", *nominees, sep=", ")
    print("Winner: ", *winners)
    print("\n")
    json_2015["awards"] = awards
    json_2015["Presenters"] = presenters
    json_2015["Nominees"] = nominees
    json_2015["Winner"] = winners


    print("Overall Awards:-------------")
    print("\n")

    json_2015["Host"] = hosts
    with open('answers1.json', 'w') as f:
        json.dump(json_2015, f, indent=4)
    return
```

In [19]:
```
if __name__ == '__main__':
    main()
```

```
C:\Users\coola\AppData\Local\Temp\ipykernel_5580\213769274.py:5: FutureWarning: The d
efault value of regex will change from True to False in a future version.
  df = df.str.replace('#GoldenGlobes|golden|globes|globe', "", case = False)
```

```
Hosts: , Amy Poehler, Tina Fey
Award:  ['best original song at the', 'best supporting actor in a television series g
oes to matt bomer for', 'best actress in a motion picture - comedy or musical', 'best
supporting actress in a television movie', 'best actor in a drama television series f
or #houseofcards', 'best animated feature film at the', 'best actress in a motion pic
ture drama goes to julianne moore for', 'best actor in a motion picture comedy goes t
o michael keaton for', 'best original song winners for #selma', 'best animated featur
e film at the', 'best actor in a drama television series for #houseofcards', 'best ac
tor in a drama television series for #houseofcards', 'best actress in a television se
ries - drama', 'best director - richard linklater', 'best dressed men at the', 'best
nudez', 'best television series actress - comedy or musical', 'best gif of the nigh
t', 'best actress in a mini-series or television movie', 'best photobomb at the', 'be
st screenplay in a motion picture goes to', 'best foreign film - leviathan', 'best mi
ni horse', 'best original score - jóhann jóhannsson', 'best supporting actor in a tel
evision movie win at the @. #thenormalheart', 'best original song in a motion picture
goes to john legend and common']
Presenters: , best motion picture - drama, best motion picture - musical or comedy, b
est performance by an actress in a motion picture - drama, best performance by an act
or in a motion picture - drama, best performance by an actress in a motion picture -
musical or comedy, best performance by an actor in a motion picture - musical or come
dy, best performance by an actress in a supporting role in any motion picture, best p
erformance by an actor in a supporting role in any motion picture, best director - mo
tion picture, best screenplay - motion picture, best motion picture - animated, best
motion picture - foreign language, best original score - motion picture, best origina
l song - motion picture, best television series - drama, best television series - mus
ical or comedy, best television limited series or motion picture made for television,
best performance by an actress in a limited series or a motion picture made for telev
ision, best performance by an actor in a limited series or a motion picture made for
television, best performance by an actress in a television series - drama, best perfo
rmance by an actor in a television series - drama, best performance by an actress in
a television series - musical or comedy, best performance by an actor in a television
series - musical or comedy, best performance by an actress in a supporting role in a
series, limited series or motion picture made for television, best performance by an
actor in a supporting role in a series, limited series or motion picture made for tel
evision, cecil b. demille award
Nominees: , best motion picture - drama, best motion picture - musical or comedy, bes
t performance by an actress in a motion picture - drama, best performance by an actor
in a motion picture - drama, best performance by an actress in a motion picture - mus
ical or comedy, best performance by an actor in a motion picture - musical or comedy,
best performance by an actress in a supporting role in any motion picture, best perfo
rmance by an actor in a supporting role in any motion picture, best director - motion
picture, best screenplay - motion picture, best motion picture - animated, best motio
n picture - foreign language, best original score - motion picture, best original son
g - motion picture, best television series - drama, best television series - musical
or comedy, best television limited series or motion picture made for television, best
performance by an actress in a limited series or a motion picture made for televisio
n, best performance by an actor in a limited series or a motion picture made for tele
vision, best performance by an actress in a television series - drama, best performan
ce by an actor in a television series - drama, best performance by an actress in a te
levision series - musical or comedy, best performance by an actor in a television ser
ies - musical or comedy, best performance by an actress in a supporting role in a ser
ies, limited series or motion picture made for television, best performance by an act
or in a supporting role in a series, limited series or motion picture made for televi
sion, cecil b. demille award
Winner:  best motion picture - drama best motion picture - musical or comedy best per
formance by an actress in a motion picture - drama best performance by an actor in a
motion picture - drama best performance by an actress in a motion picture - musical o
r comedy best performance by an actor in a motion picture - musical or comedy best pe
rformance by an actress in a supporting role in any motion picture best performance b
y an actor in a supporting role in any motion picture best director - motion picture
```

```
best screenplay - motion picture best motion picture - animated best motion picture -
foreign language best original score - motion picture best original song - motion pic
ture best television series - drama best television series - musical or comedy best t
elevision limited series or motion picture made for television best performance by an
actress in a limited series or a motion picture made for television best performance
by an actor in a limited series or a motion picture made for television best performa
nce by an actress in a television series - drama best performance by an actor in a te
levision series - drama best performance by an actress in a television series - music
al or comedy best performance by an actor in a television series - musical or comedy
best performance by an actress in a supporting role in a series, limited series or mo
tion picture made for television best performance by an actor in a supporting role in
a series, limited series or motion picture made for television cecil b. demille award


Overall Awards:-------------
```