In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
# from unidecode import unidecode # For replacing non-english characters
from nltk.corpus import stopwords
from nltk import word_tokenize
STOPWORDS = set(stopwords.words('english'))
import re
from collections import Counter
import networkx as nx # For graph visualization
```

In [2]:
```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', None)
```

Step 1: Read and display the script data

In [3]:
```
scripts_df = pd.read_csv('../Data/lotr/lotr_scripts.csv')
```

In [4]:
```
scripts_df.head(2)
```

Out[4]:

	Unnamed: 0	char	dialog	movie
0	0	DEAGOL	Oh Smeagol Ive got one! , Ive got a fish Smeagol, Smeagol!	The Return of the King
1	1	SMEAGOL	Pull it in! Go on, go on, go on, pull it in!	The Return of the King

In [5]:
```
scripts_df.shape
```

Out[5]:
```
(2390, 4)
```

Step 2: We will start with editing scripts_df

In [6]:
```
scripts_df.drop('Unnamed: 0', inplace=True, axis=1) #Remove dummy first colu
```

In [7]:
```
scripts_df.describe() # Describe the dataframe
```

Out[7]:

	char	dialog	movie
count	2390	2389	2390
unique	118	2325	3
top	FRODO	DEATH!	The Two Towers
freq	225	6	1010

In [8]:
```
scripts_df.dtypes # Look at the types. We notice that they are all object sc
```

Out[8]:
```
char      object
dialog    object
movie     object
dtype: object
```

In [9]:
```
# Utility function to get character classes
def get_char_classes(df, field):
    print(f"We have {len(df[field].unique())} characters as follows for Lorc
```

In [10]:
```
get_char_classes(scripts_df, 'char')
```

```
We have 118 characters as follows for Lord of The Rings:
 ['DEAGOL' 'SMEAGOL' '(GOLLUM' 'FRODO' 'MERRY' 'GIMLI' 'GOLLUM' 'SAM'
 'GANDALF' 'ARAGORN' 'PIPPIN' 'HOBBIT' 'ROSIE' 'BILBO' 'TREEBEARD'
 'SARUMAN' 'THEODEN' 'GALADRIL' 'ELROND' 'GRIMA' 'FRODO VOICE OVER'
 'WITCH KING' 'EOWYN' 'FARAMIR' 'ORC' '\xa0GANDALF' 'SOLDIERS ON GATE'
 'GOTHMOG' 'GENERAL' 'CAPTAIN' 'SOLDIER' 'MOUTH OF SAURON' 'EOMER' 'ARMY'
 'BOSON' 'MERCENARY' 'EOWYN/MERRY' 'DENETHOR' 'ROHIRRIM'
 'GALADRIEL VOICEOVER' 'LEGOLAS' 'GALADRIEL' 'KING OF THE DEAD' 'GRIMBOLD'
 'IROLAS' 'ORCS' 'GAMLING' 'MADRIL' 'DAMROD' 'SOLDIERS'
 'SOLDIERS IN MINAS TIRITH' 'GANDALF VOICEOVER' 'SOLDIER 1' 'SOLDIER 2'
 'WOMAN' 'HALDIR' 'SAM VOICEOVER' 'OLD MAN' 'BOROMIR' 'CROWD' 'ARWEN'
 'ELROND VOICEOVER' 'ARWEN VOICEOVER' 'ARAGORN ' 'HAMA' 'SHARKU' 'PEOPLE'
 'LADY' 'FREDA' 'MORWEN' 'EYE OF SAURON' 'ROHAN STABLEMAN' 'GORBAG'
 'ARGORN' 'GANDALF VOICE OVER' 'BOROMIR ' 'UGLUK' 'SHAGRAT'
 'SARUMAN VOICE OVER' 'SARUMAN VOICE OVER ' 'FRODO ' 'URUK-HAI' 'SNAGA'
 'GRISHNAKH' 'MERRY and PIPPIN' 'WILDMAN' 'STRIDER' 'GALADRIEL VOICE-OVER'
 'EOTHAIN' 'ROHAN HORSEMAN' 'SAURON VOICE' 'SAM ' 'FRODO VOICE'
 'GALADRIEL VOICE OVER' 'FARMER MAGGOT' 'WHITE WIZARD' 'MERRY AND PIPPIN'
 'GAFFER' 'NOAKES' 'SANDYMAN' 'FIGWIT' 'GENERAL SHOUT' 'GRISHNAK'
 'URUK HAI' 'SARUMAN VOICEOVER' 'MRS BRACEGIRDLE' 'BILBO VOICEOVER'
 'PROUDFOOT HOBBIT' 'GATEKEEPER' 'GATEKEEPR' 'MAN' 'CHILDREN HOBBITS'
 'BARLIMAN' 'RING' 'MEN' 'VOICE' 'SAURON' 'GAN DALF']
```

We see several problems:

1. Upercase all characters
2. '(GOLLUM' has punctuation so need to remove that
3. There are some garbage characters trailing and leading
4. Whenever, VOICE OVER is a suffix, that is the same as the actual character speaking for our purposes
5. Some character names are mispelled such as GRISHNAKH is spelled in two different ways and others like STRIDER and ARAGORN are the same person
6. (optional): remove all white-space. This is a last reserve Solution: we will apply each transformation separately and observe the results

In [11]:
```
def remove_voice(character):
    if character == 'VOICE':
        character = 'NARRATOR'
    character = character.replace('VOICEOVER', '')
    character = character.replace('VOICE OVER', '')
    character = character.replace('VOICE', '')
    character = character.strip()
    return character
```

In [12]:
```
def fix_spelling(character):
    if character == 'STRIDER':
        character = 'ARAGORN'
    if 'GAN' in character and 'DALF' in character:
        character = 'GANDALF'
    if 'SOLDIER' in character:
        character = 'SOLDIER'
    if 'GRISHNAK' in character:
        character = 'GRICKNAK'
    if 'URUK' in character:
        character = 'URUKHAI'
    if 'GATEKEEP' in character:
        character = 'GATEKEEPER'
    if 'ORC' in character:
        character = 'ORC'
    if 'GALAD' in character:
        character = 'GALADRIEL'
    return character
```

In [13]:
```
def clean_dialog(text):


    REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
    BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
    STOPWORDS = set(stopwords.words('english'))

    text = text.lower()
    text = REPLACE_BY_SPACE_RE.sub(' ', text)
    text = BAD_SYMBOLS_RE.sub('', text)
    text = [word for word in text.split() if word not in STOPWORDS]
    #text = [i for i in text if i] ### remove empty strings
    text = ' '.join(text)

    return text
```

In [14]:
```
def clean_df(df, field):
    df = df.astype(str)
    if field == 'char':

        df[field] = df[field].str.strip()
        df[field] = df[field].str.upper()
        #1. Remove punctuation
        df[field] = scripts_df[field].str.replace('[{}]'.format(string.punct
        #2.Remove trailing and leading garbage characters
        df[field] = df[field].str.strip()
        #3. Remove voice
        df[field] = df[field].apply(remove_voice)
        #4. Fix spelling
        df[field] = df[field].apply(fix_spelling)

    elif field == 'movie':

        df[field] = df[field].str.strip()
        df[field] = df[field].str.upper()

    elif field == 'dialog':

        df[field] = df[field].apply(clean_dialog)

#         df[field] = df[field].apply(lemmatize_sentence)

        df = df[df[field].apply(lambda x: len(x)>5)]




    return df
```

In [15]:
```
scripts_df = clean_df(scripts_df, 'char')
```

In [16]:
```
get_char_classes(scripts_df, 'char')
```

```
We have 85 characters as follows for Lord of The Rings:
 ['DEAGOL' 'SMEAGOL' 'GOLLUM' 'FRODO' 'MERRY' 'GIMLI' 'SAM' 'GANDALF'
 'ARAGORN' 'PIPPIN' 'HOBBIT' 'ROSIE' 'BILBO' 'TREEBEARD' 'SARUMAN'
 'THEODEN' 'GALADRIEL' 'ELROND' 'GRIMA' 'WITCH KING' 'EOWYN' 'FARAMIR'
 'ORC' 'SOLDIER' 'GOTHMOG' 'GENERAL' 'CAPTAIN' 'MOUTH OF SAURON' 'EOMER'
 'ARMY' 'BOSON' 'MERCENARY' 'EOWYNMERRY' 'DENETHOR' 'ROHIRRIM' 'LEGOLAS'
 'KING OF THE DEAD' 'GRIMBOLD' 'IROLAS' 'GAMLING' 'MADRIL' 'DAMROD'
 'WOMAN' 'HALDIR' 'OLD MAN' 'BOROMIR' 'CROWD' 'ARWEN' 'HAMA' 'SHARKU'
 'PEOPLE' 'LADY' 'FREDA' 'MORWEN' 'EYE OF SAURON' 'ROHAN STABLEMAN'
 'GORBAG' 'ARGORN' 'UGLUK' 'SHAGRAT' 'URUKHAI' 'SNAGA' 'GRICKNAK'
 'MERRY and PIPPIN' 'WILDMAN' 'EOTHAIN' 'ROHAN HORSEMAN' 'SAURON'
 'FARMER MAGGOT' 'WHITE WIZARD' 'MERRY AND PIPPIN' 'GAFFER' 'NOAKES'
 'SANDYMAN' 'FIGWIT' 'GENERAL SHOUT' 'MRS BRACEGIRDLE' 'PROUDFOOT HOBBIT'
 'GATEKEEPER' 'MAN' 'CHILDREN HOBBITS' 'BARLIMAN' 'RING' 'MEN' 'NARRATOR']
```

In [17]:
```
scripts_df.char.value_counts()
```

Out[17]:

FRODO	229
SAM	218
GANDALF	215
ARAGORN	212
PIPPIN	163
MERRY	137
GOLLUM	134
GIMLI	116
THEODEN	110
FARAMIR	65
EOWYN	56
LEGOLAS	55
SMEAGOL	49
BILBO	48
TREEBEARD	46
DENETHOR	45
SOLDIER	41
SARUMAN	41
BOROMIR	41
ARWEN	40
EOMER	36
ELROND	31
GRIMA	25
ORC	25
GALADRIEL	19
GAMLING	15
GOTHMOG	14
UGLUK	9
SHAGRAT	9
WITCH KING	8
KING OF THE DEAD	8
MADRIL	7
HALDIR	7
HAMA	6
URUKHAI	6
MORWEN	5
MOUTH OF SAURON	5
GRICKNAK	5
BARLIMAN	4
MERRY and PIPPIN	3
CHILDREN HOBBITS	3
FREDA	3
GATEKEEPER	3
HOBBIT	3
GAFFER	3
WOMAN	3
GORBAG	3
IROLAS	3
WILDMAN	3
SAURON	3
DEAGOL	3
CROWD	3
SNAGA	2
SANDYMAN	2
PEOPLE	2
GENERAL	2

```
DAMROD                2
ARGORN                2
MERRY AND PIPPIN      2
ROSIE                 2
ROHIRRIM              2
CAPTAIN               2
ARMY                  2
ROHAN HORSEMAN        2
NARRATOR              2
FARMER MAGGOT         1
EOWYNMERRY            1
ROHAN STABLEMAN       1
LADY                  1
MRS BRACEGIRDLE       1
GRIMBOLD              1
SHARKU                1
GENERAL SHOUT         1
RING                  1
MERCENARY             1
OLD MAN               1
MAN                   1
MEN                   1
EYE OF SAURON         1
WHITE WIZARD          1
NOAKES                1
FIGWIT                1
PROUDFOOT HOBBIT      1
EOTHAIN               1
BOSON                 1
Name: char, dtype: int64
```

In [18]:
```
# Extract top 10 chars in each movie
```

In [19]:
```
# Extract the top 10 characters by lines
top10_chars_allmovies = scripts_df.char.value_counts().index.tolist()[:10]
```

In [21]:
```
top10_chars_allmovies
```

Out[21]:
```
['FRODO',
 'SAM',
 'GANDALF',
 'ARAGORN',
 'PIPPIN',
 'MERRY',
 'GOLLUM',
 'GIMLI',
 'THEODEN',
 'FARAMIR']
```

Based on the character value counts, it appears that Frodo and Sam have the most lines in all the movies and in general, the top 4 characters have the most lines in all the movies

Based on this scripts value counts, most of the lines are unique

In [22]:
```
scripts_df.movie.value_counts()
```

Out[22]:
```
The Two Towers                1010
The Return of the King         873
The Fellowship of the Ring     507
Name: movie, dtype: int64
```

Based on the movie value counts, the second movie "The Two Towers" has the most dialogue.

In [23]:
```
scripts_df.movie.unique()
```

Out[23]:
```
array(['The Return of the King ', 'The Two Towers ',
       'The Fellowship of the Ring '], dtype=object)
```

In [24]:
```
scripts_df = clean_df(scripts_df, 'movie')
```

We now start preprocessing the actual dialog data

In [25]:
```
scripts_df[['dialog']].head(50)
```

Out[25]:

	dialog
0	Oh Smeagol Ive got one! , Ive got a fish Smeagol, Smeagol!
1	Pull it in! Go on, go on, go on, pull it in!
2	Arrghh!
3	Deagol!
4	Deagol!
5	Deagol!
6	Give us that! Deagol my love
7	Why?
8	Because' , it's my birthday and I wants it.
9	My precious.
10	They cursed us
11	Murderer)
12	'Murderer' they called us. They cursed us and drove us away.
13	Gollum' Gollum' Gollum' , and we wept precious. We wept to be so alone.
14	and cool, so nice for feet' and we only wish to catch a fish , so juicy sweet.
15	And we forgot the taste of bread, the sound of trees and the softness of the wind. We even forgot our own name.
16	Gandalf?
17	Oooohhh!
18	Frodo!
19	Aaaahh!
20	Gimli!
21	My precious!
22	Wake up! , Wake up! ,Wake up sleepies! We must go, yes, we must go at once.
23	Haven't you had any sleep Mr Frodo?
24	And I've gone and had too much. , It must be getting late.
25	No, it isn't. It isn't midday yet. , The days are growing darker.
26	Come on! Must go! No time!
27	Not before Mr Frodo's had something to eat.
28	No time to lose silly!
29	Here.
30	What about you?
31	Oh no, I'm not hungry, leastways not for lembas bread.
32	Sam!
33	Aragorn!

	dialog
34	Now come the days of the King. May they be blessed.
35	This day does not belong to one man, but to all. Let us together rebuild this world, that we may share in the days of peace.
36	Et E'rello Endorenna ut'lien. Sinome maruvan ar Hildinyar tenn' Ambar-metta!
37	Hannon le.
38	My friends, , you bow to no-one.
39	,And thus it was a Fourth Age of Middle Earth began. And the Fellowship of the Ring, though eternally bound by friendship and love was ended. Thirteen months to the day since Gandalf sent us on our long journey we find ourselves looking upon a familiar sight.
40	We were home.
41	Alright!
42	Hey watch the pumpkin.
43	Goodnight lads.
44	Oh alright, but we don't have that much left. We have to be careful or we are going to run out. You go ahead and eat that Mr Frodo. I've rationed it, there should be enough.
45	For what?
46	The journey home.
47	Come Hobbitses, very close now. Very close to Mordor. No safe places here. Hurry.
48	It's good. Definitely from the Shire. ,Longbottom leaf eh?
49	uhh huh , I feel like I'm back at the Green Dragon.

In [26]:
```
len(scripts_df.dialog.unique())
```

Out[26]: 2326

In [27]:
```
scripts_df = clean_df(scripts_df, 'dialog')
```

In [28]:
```
scripts_df[['dialog']].head(10)
```

Out[28]:

	dialog
0	oh smeagol ive got one ive got fish smeagol smeagol
1	pull go go go pull
2	arrghh
3	deagol
4	deagol
5	deagol
6	give us deagol love
8	birthday wants
9	precious
10	cursed us

In [29]:
```
scripts_df.dialog.isna().any()
```

Out[29]:
```
False
```

In [30]:
```
scripts_df['corpus_dialog'] = scripts_df.dialog.apply(lambda x: Counter(x.sp
```

In [31]:
```
scripts_df[['corpus_dialog']].head(10)
```

Out[31]:

	corpus_dialog
0	{'oh': 1, 'smeagol': 3, 'ive': 2, 'got': 2, 'one': 1, 'fish': 1}
1	{'pull': 2, 'go': 3}
2	{'arrghh': 1}
3	{'deagol': 1}
4	{'deagol': 1}
5	{'deagol': 1}
6	{'give': 1, 'us': 1, 'deagol': 1, 'love': 1}
8	{'birthday': 1, 'wants': 1}
9	{'precious': 1}
10	{'cursed': 1, 'us': 1}

In [32]:
```
scripts_df = scripts_df[scripts_df.char.isin(top10_chars_allmovies)]
```

In [33]:
```
scripts_df.to_csv('../Data/lotr/prepped_data.csv',index=False)
```

Visualizations

Step 3: Now we will look at the vocabulary of each character over the course of the movies to see any transformations such as growth/regression

In [34]:
```
fellowship_df = scripts_df[scripts_df.movie=='THE FELLOWSHIP OF THE RING']
two_towers_df = scripts_df[scripts_df.movie=='THE TWO TOWERS']
return_king_df = scripts_df[scripts_df.movie=='THE RETURN OF THE KING']
```

Get top 10 chars for each movies separately and then find the intersection

In [35]:
```
top10_chars_1 = fellowship_df.char.value_counts().index.tolist()[:10]
top10_chars_2 = two_towers_df.char.value_counts().index.tolist()[:10]
top10_chars_3 = return_king_df.char.value_counts().index.tolist()[:10]
```

fellowship_df.groupby('char')['dialog'].count().reset_index(name='obs').sort_values(['obs'], ascending=False).head(10)

In [36]:
```
two_towers_df.groupby('char')['dialog'].count().reset_index(name='obs').sort
```

Out[36]:

	char	obs
0	ARAGORN	85
8	SAM	81
5	GOLLUM	68
2	FRODO	66
9	THEODEN	59
4	GIMLI	51
3	GANDALF	46
6	MERRY	44
7	PIPPIN	42
1	FARAMIR	34

In [37]:
```
return_king_df.groupby('char').size().reset_index(name='obs').sort_values(['
```

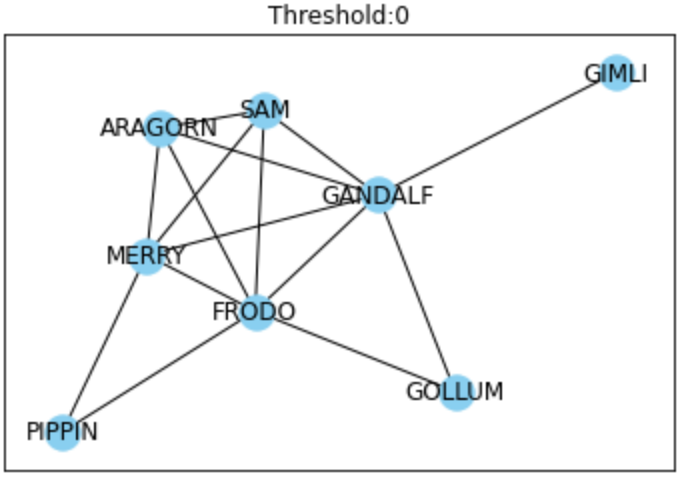
Out[37]:

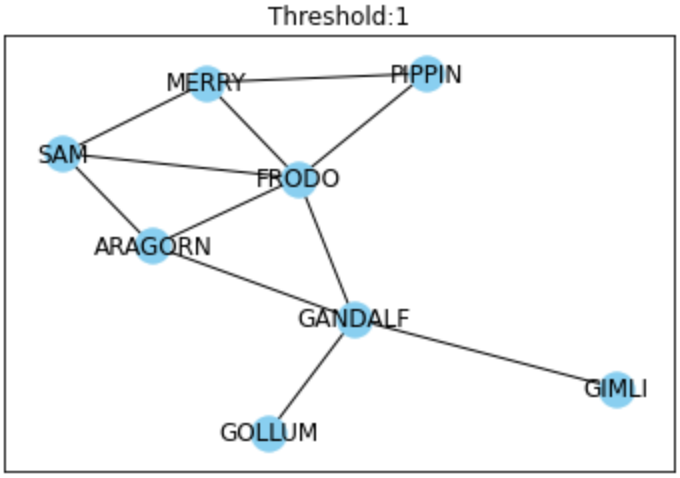
	char	obs
3	GANDALF	84
8	SAM	77
2	FRODO	58
7	PIPPIN	52
0	ARAGORN	49
5	GOLLUM	47
9	THEODEN	39
6	MERRY	33
4	GIMLI	30
1	FARAMIR	22

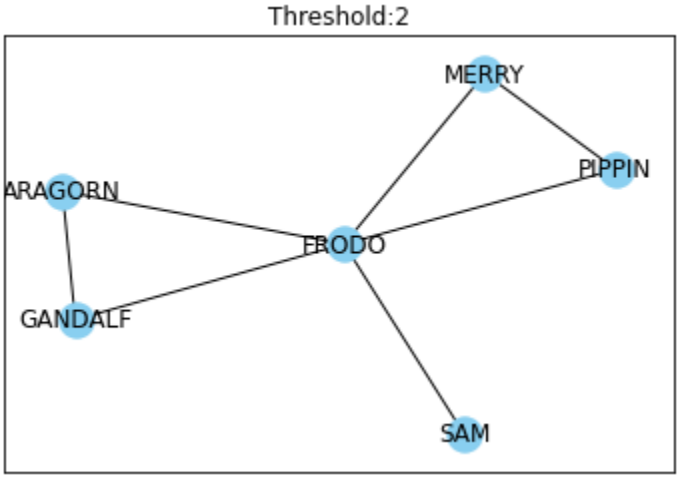
In [38]:
```
def consolidate_char_vocab(df, character):
    char_vocabularies = df[df.char == character].corpus_dialog
    #Dummy dictionary for plotting character growth
    if len(char_vocabularies) == 0:
        my_dict = {'NA1':0, 'NA2':0, 'NA3':0, 'NA4':0, 'NA5':0}
        return my_dict
    list_of_dicts = char_vocabularies.tolist()
    master_dict = list_of_dicts[0]
    for dictionary in list_of_dicts[1:]:
        master_dict = master_dict + dictionary
    return master_dict
```

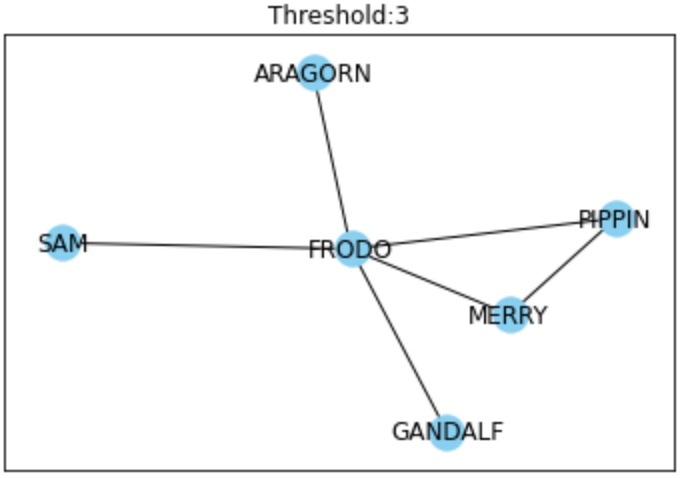
In [39]:
```
def get_relationships(df, char_list, thresh):
    # Get char vocabularies for a specific movie
    dict_of_char_vocabs = {}
    for char in char_list:
        dict_of_char_vocabs[char] = consolidate_char_vocab(df, char)

    # Create visual


    edge_list = []
    visited = set()
    for char in dict_of_char_vocabs:

        vocab = dict_of_char_vocabs[char]


        for word in vocab:
            if word.upper() in dict_of_char_vocabs and vocab[word] > thresh:
                visited.add(word)
                edge_list.append((char, word.upper()))

    G = nx.Graph()
    G.add_edges_from(edge_list)

    nx.draw_networkx(G=G, node_size=300, node_color='#89CFF0')
    plt.title(f"Threshold:{i}")

    plt.show()
```

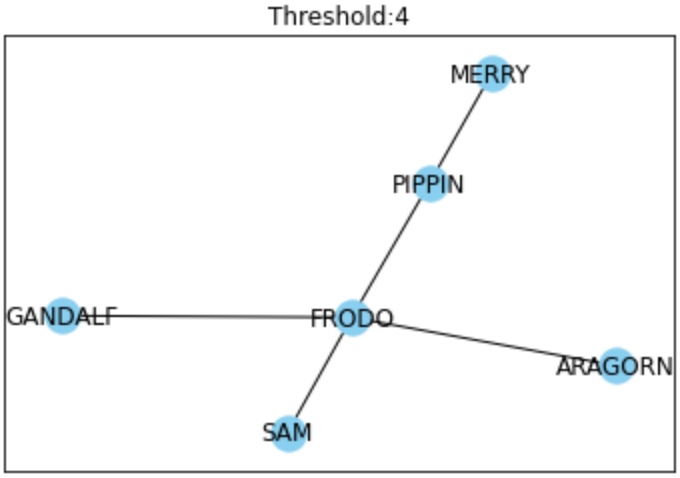
In [40]:
```
for i in range(34):
    get_relationships(fellowship_df, fellowship_df.char.unique().tolist(), i
```

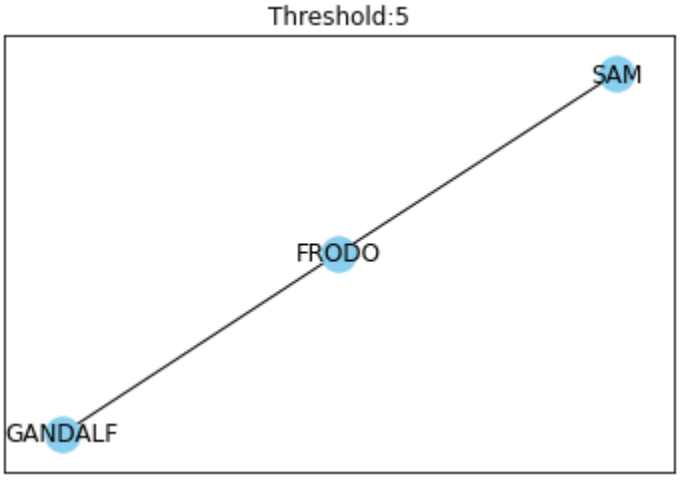


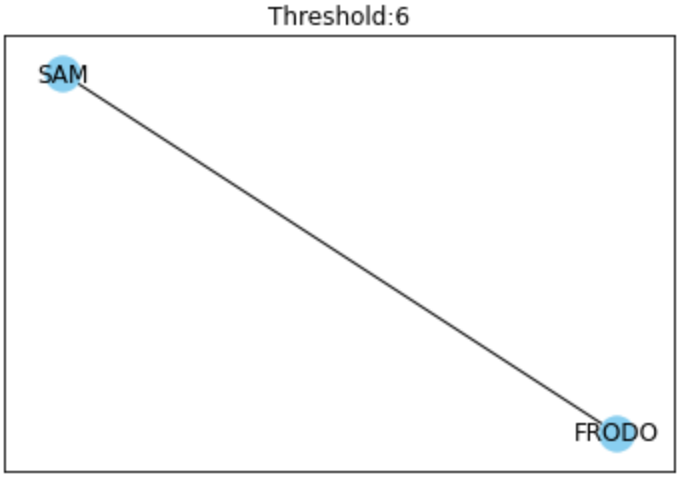

Threshold:1
MERRY
PIPPIN
SAM
FRODO
ARAGORN
GANDALF
GIMLI
GOLLUM

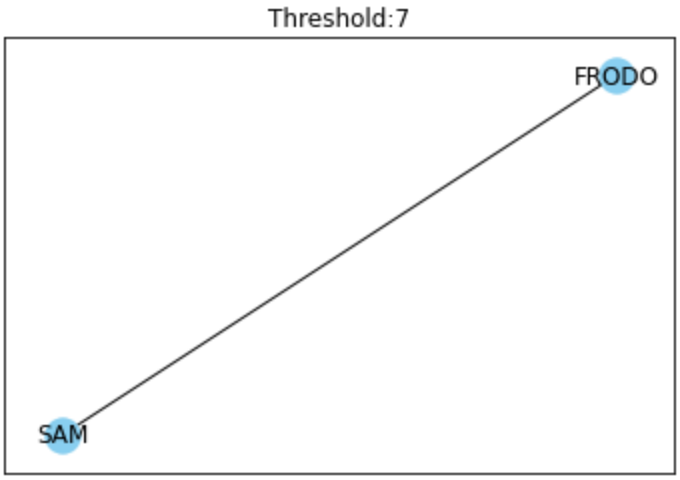

Threshold:2
MERRY
PIPPIN
ARAGORN
FRODO
GANDALF
SAM

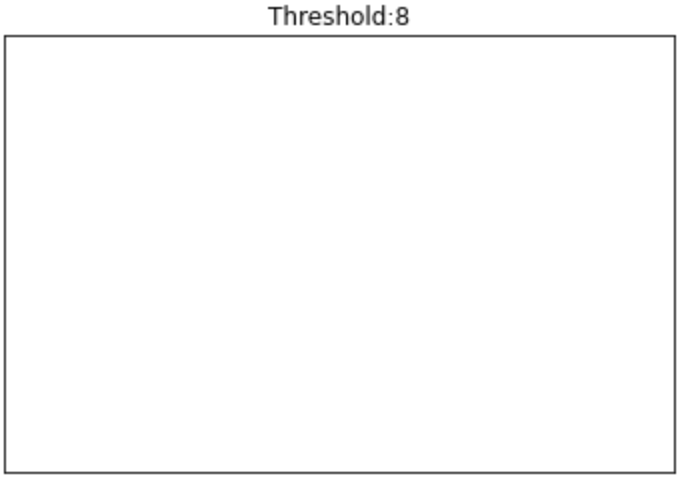

Threshold:3
ARAGORN
PIPPIN
SAM
FRODO
MERRY
GANDALF

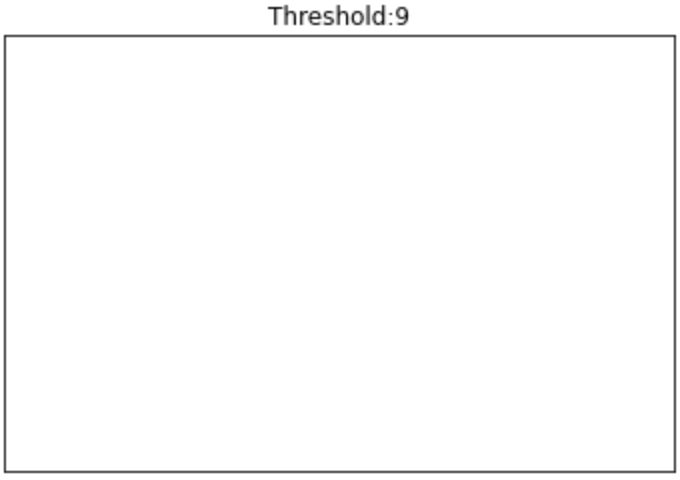

Threshold:4
MERRY
PIPPIN
GANDALF
FRODO
ARAGORN
SAM


Threshold:5
SAM
FRODO
GANDALF


Threshold:6
SAM
FRODO


Threshold:7
FRODO
SAM


Threshold:8


Threshold:9

Threshold:10

Threshold:11

Threshold:12

Threshold:13

Threshold:14

Threshold:15

Threshold:16

Threshold:17

Threshold:18

Threshold:19

Threshold:20

Threshold:21

Threshold:22

Threshold:23

Threshold:24

Threshold:25

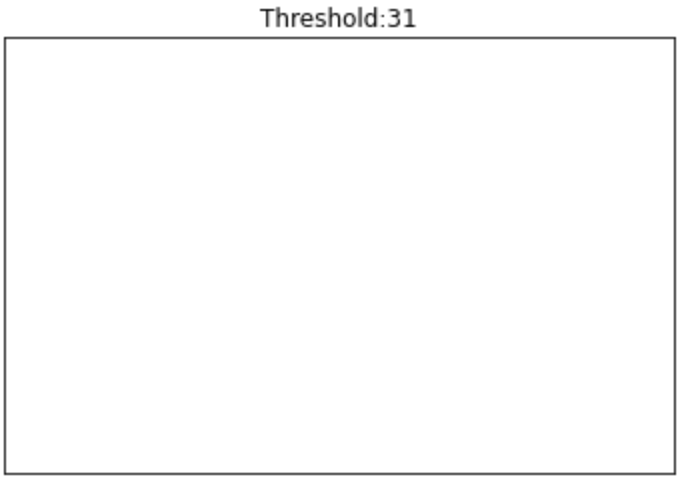
Threshold:26

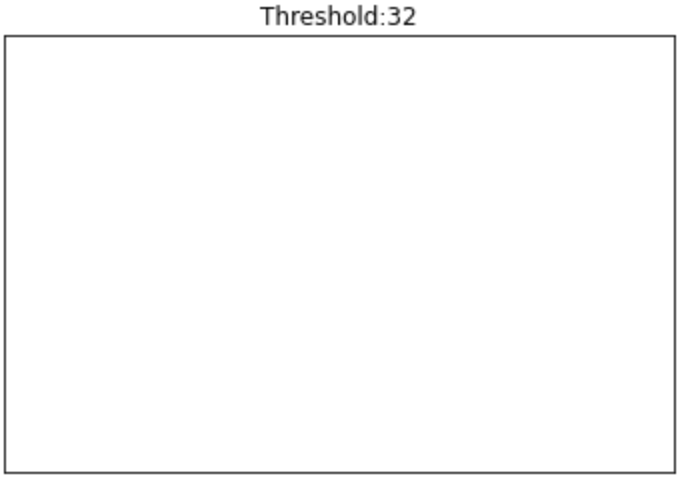
Threshold:27

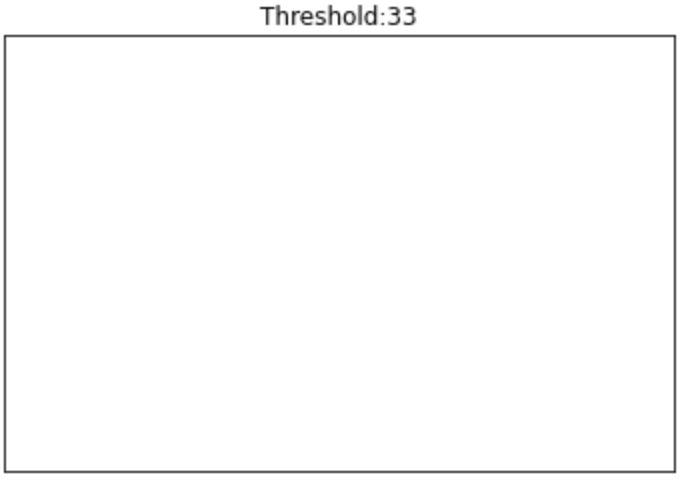
Threshold:28

Threshold:29

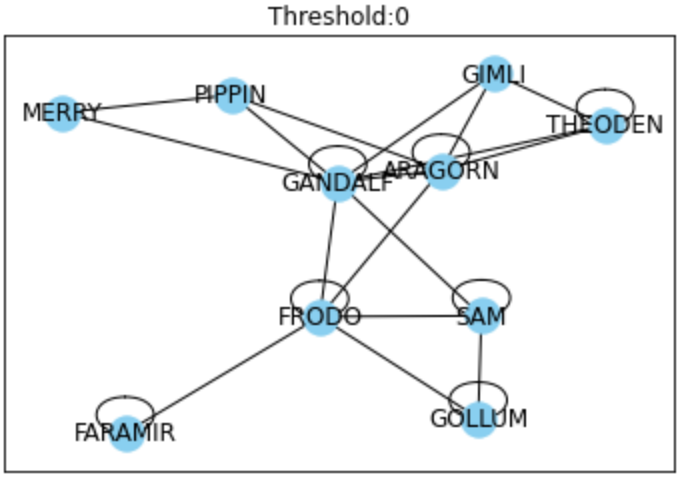
Threshold:30

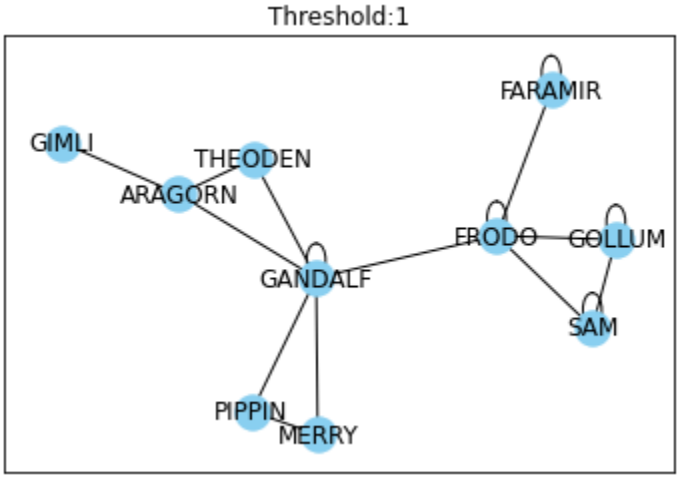


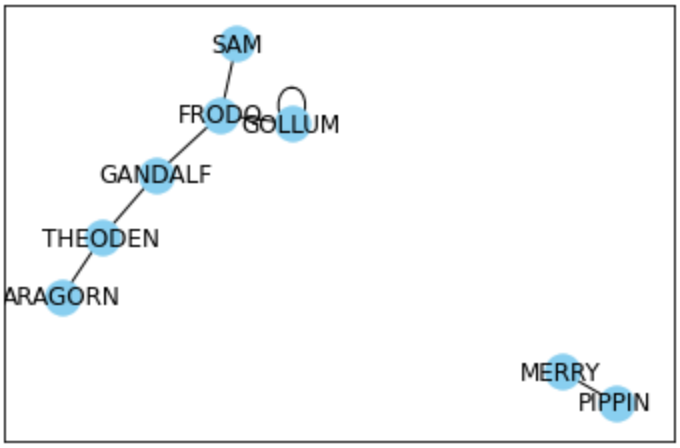


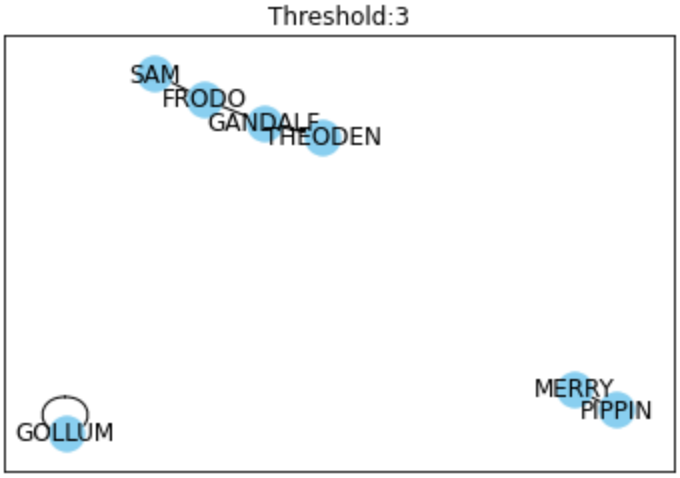


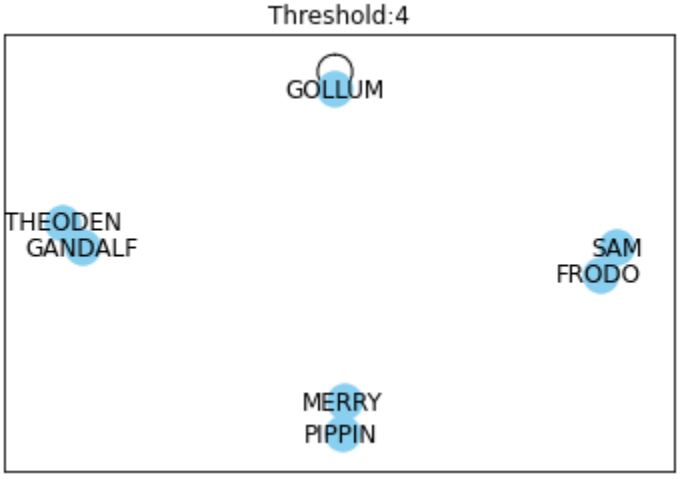
In [41]:
```
for i in range(34):
    get_relationships(two_towers_df, two_towers_df.char.unique().tolist(), i
```

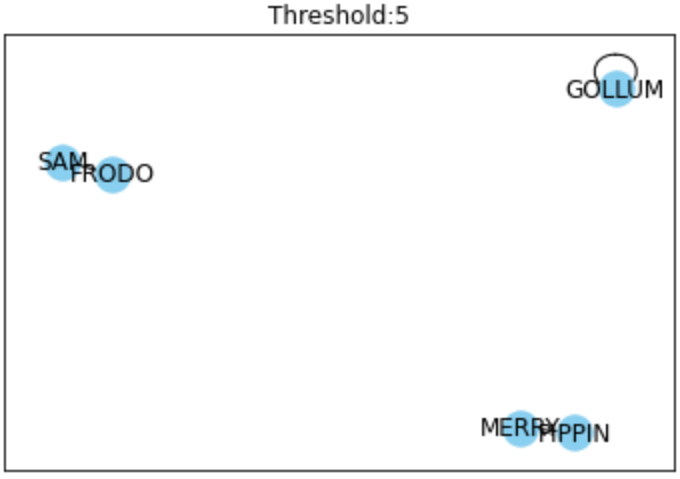

Threshold:0
GIMLI
PIPPIN
MERRY
THEODEN
ARAGORN
GANDALF
FRODO
SAM
FARAMIR
GOLLUM

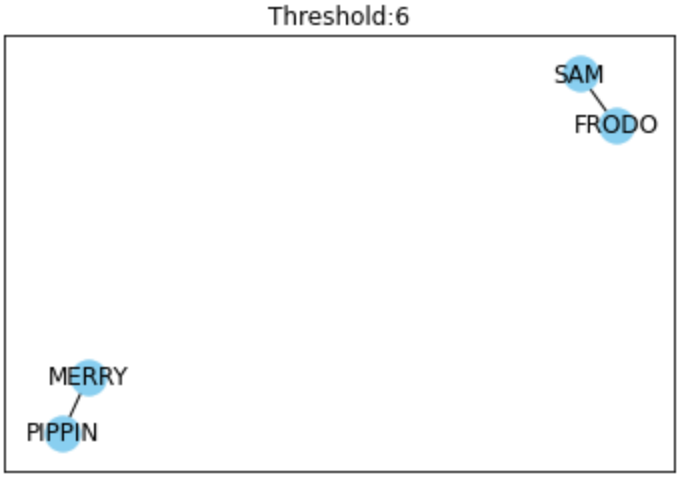

Threshold:1
FARAMIR
GIMLI
THEODEN
ARAGORN
FRODO
GOLLUM
GANDALF
SAM
PIPPIN
MERRY

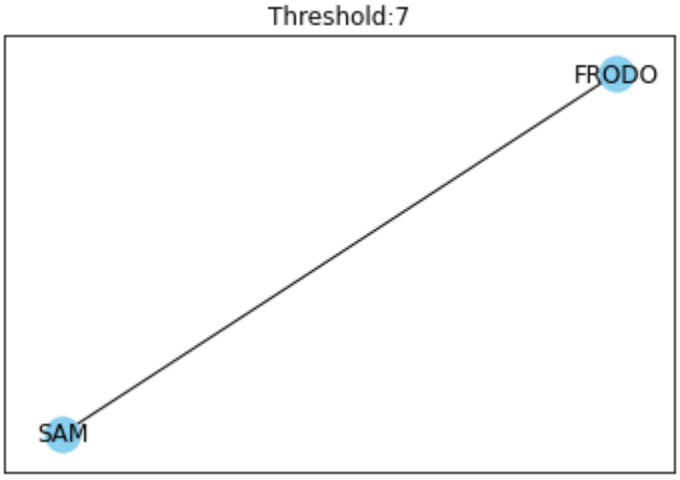

Threshold:2
SAM
FRODO
GOLLUM
GANDALF
THEODEN
ARAGORN
MERRY
PIPPIN

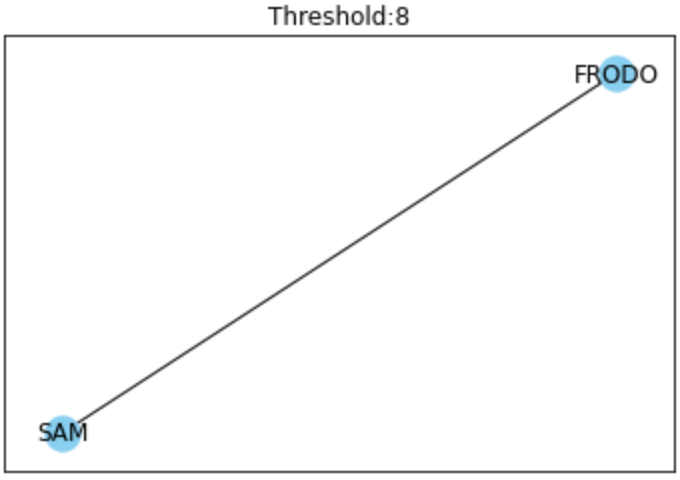

Threshold:3
SAM
FRODO
GANDALF
THEODEN
MERRY
PIPPIN
GOLLUM

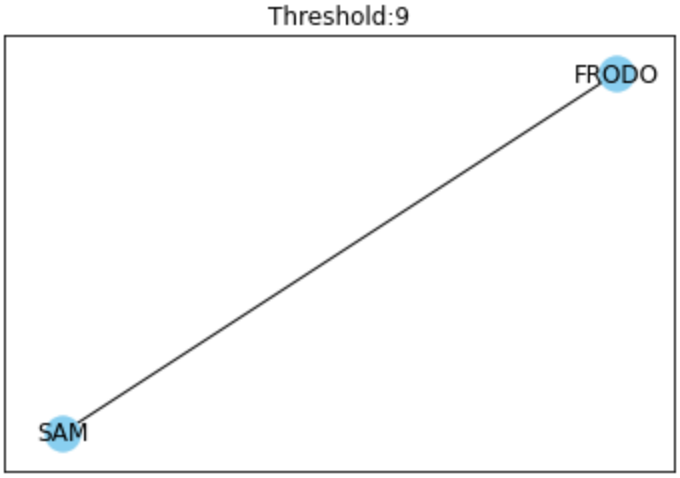

Threshold:4
GOLLUM
THEODEN
GANDALF
SAM
FRODO
MERRY
PIPPIN

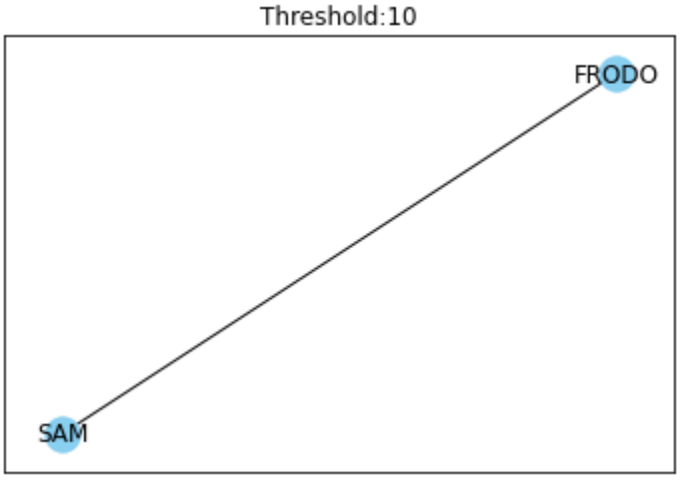

Threshold:5
GOLLUM
SAM
FRODO
MERRY
PIPPIN

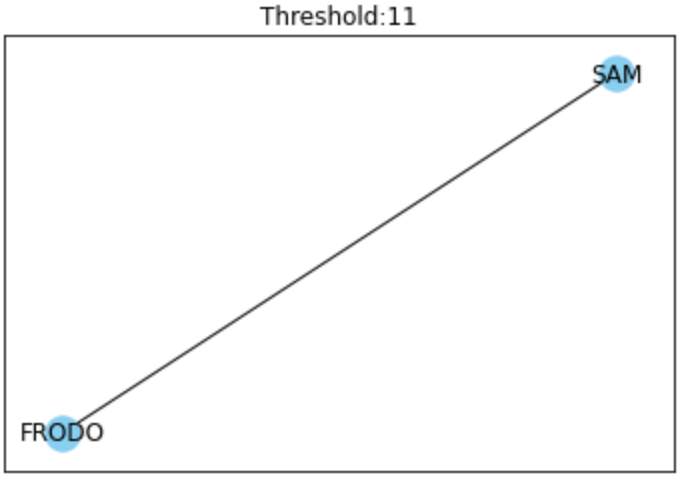

Threshold:6
SAM
FRODO
MERRY
PIPPIN

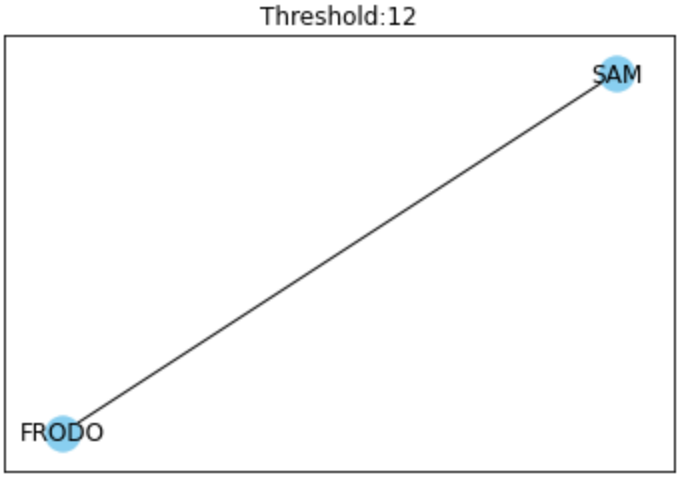

Threshold:7
FRODO
SAM

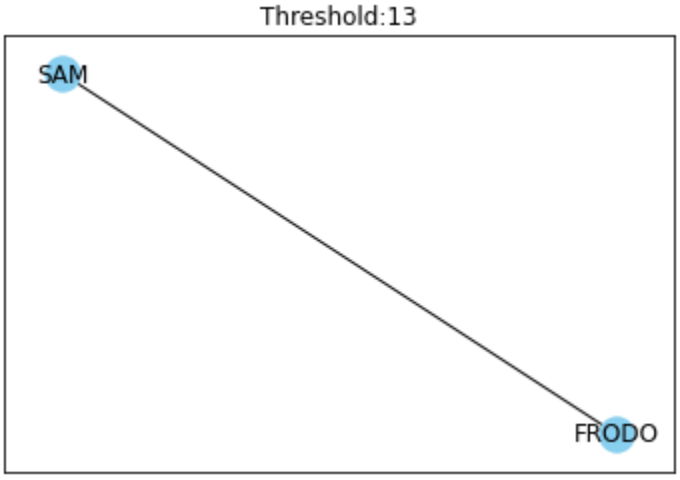

Threshold:8
FRODO
SAM

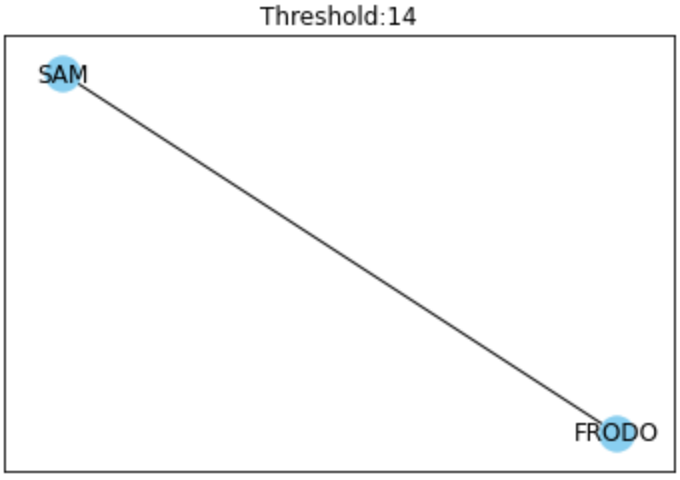

Threshold:9
FRODO
SAM

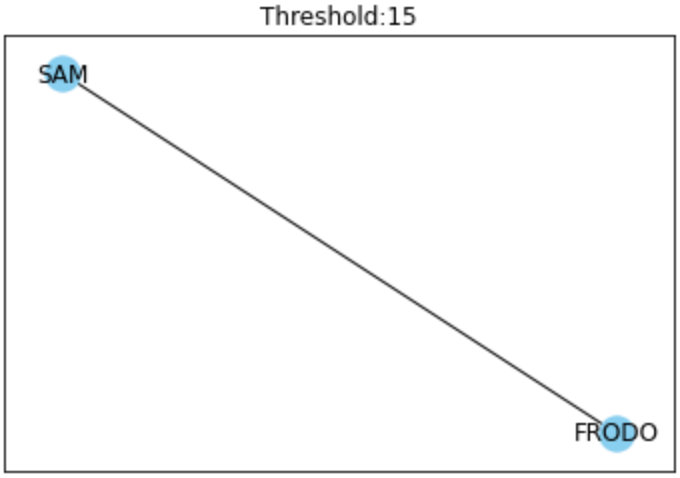

Threshold:10
FRODO
SAM

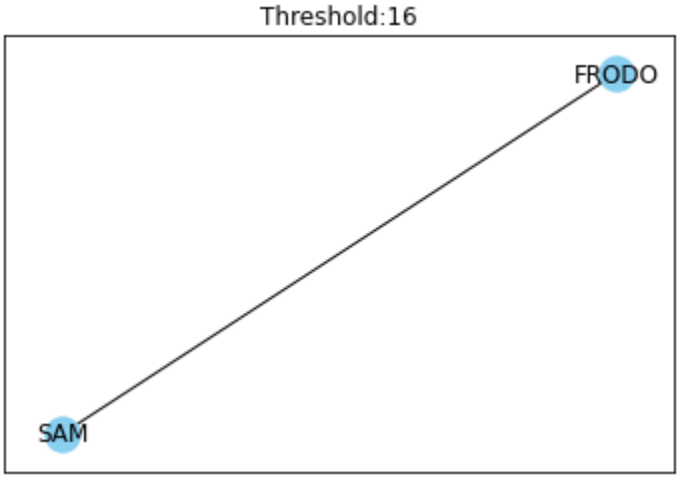

Threshold:11
SAM
FRODO

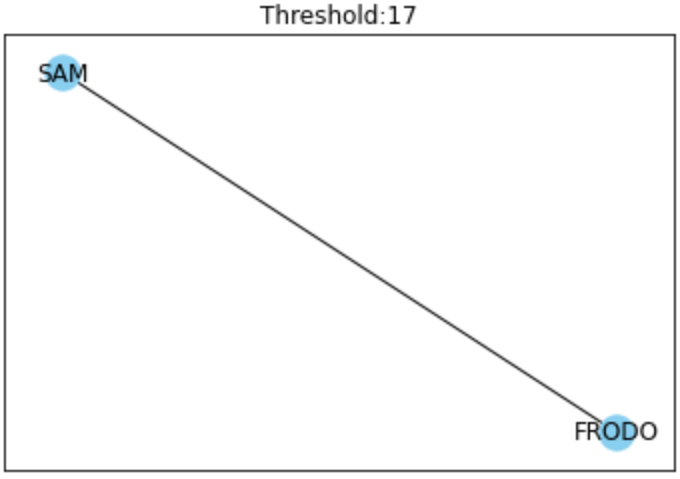

Threshold:12
SAM
FRODO

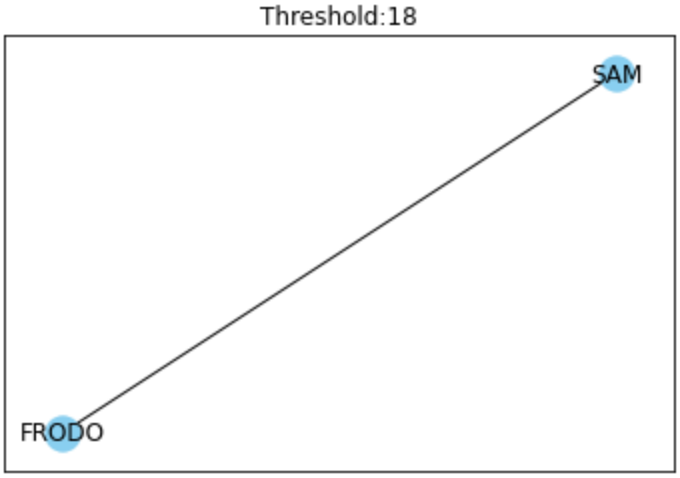

Threshold:13
SAM
FRODO

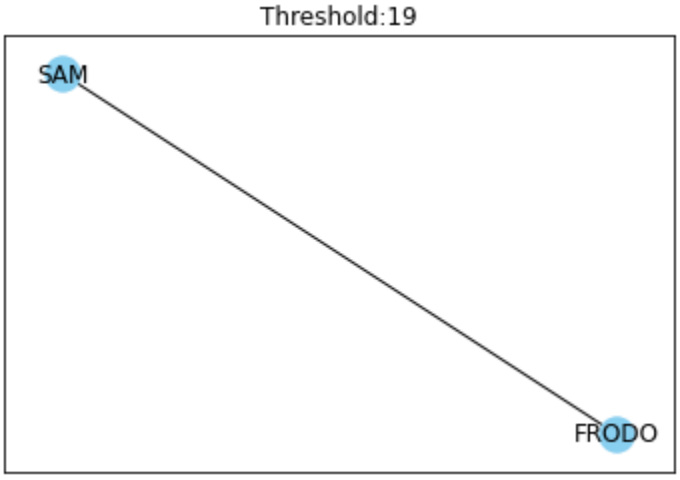

Threshold:14
SAM
FRODO

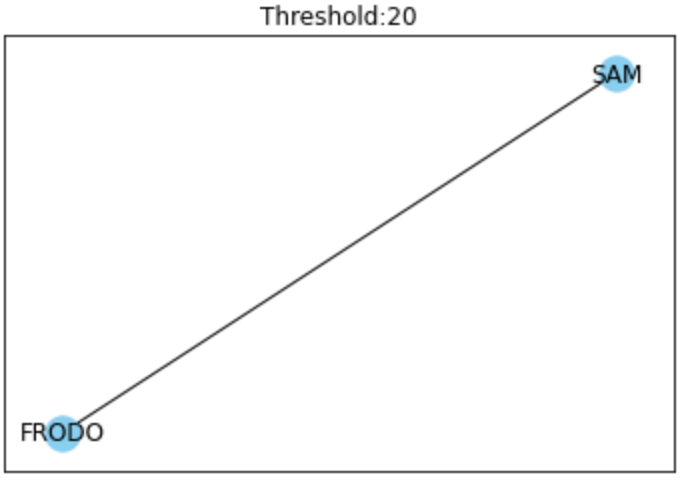

Threshold:15
SAM
FRODO

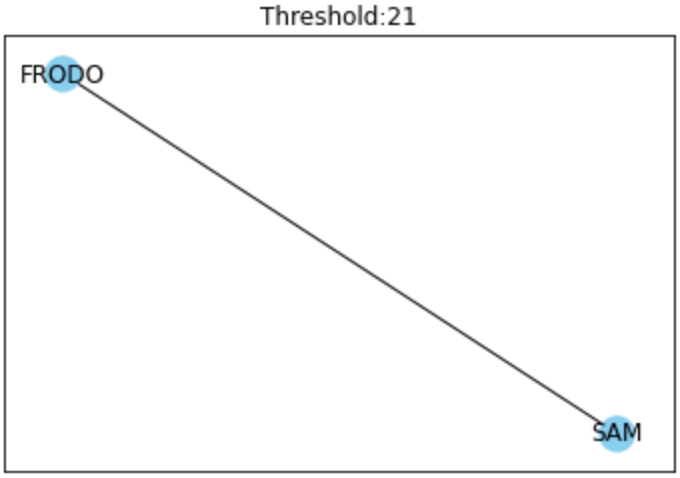

Threshold:16
FRODO
SAM

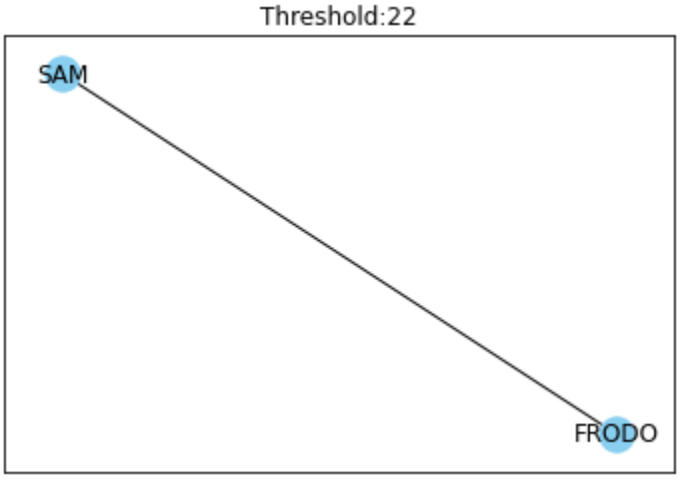

Threshold:17
SAM
FRODO

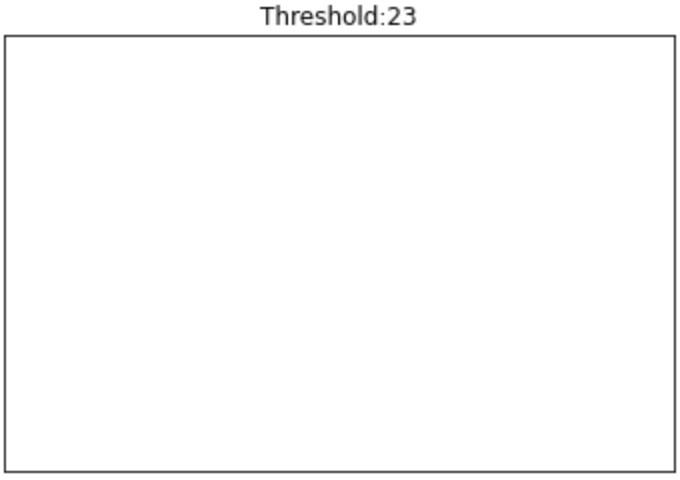

Threshold:18
SAM
FRODO


Threshold:19
SAM
FRODO


Threshold:20
SAM
FRODO


Threshold:21
FRODO
SAM


Threshold:22
SAM
FRODO


Threshold:23

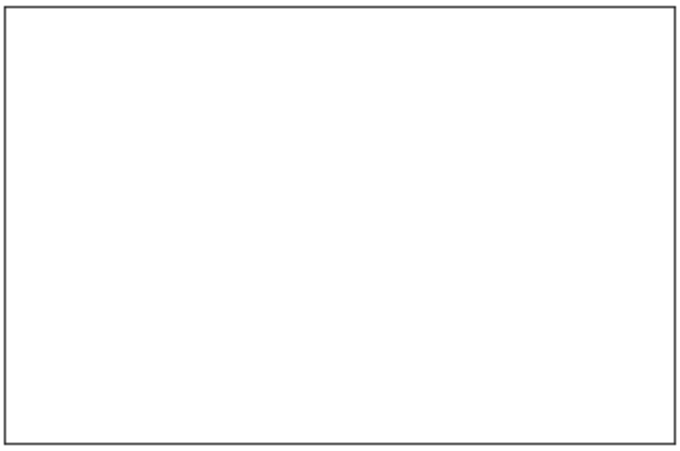
Threshold:24

Threshold:25

Threshold:26

Threshold:27

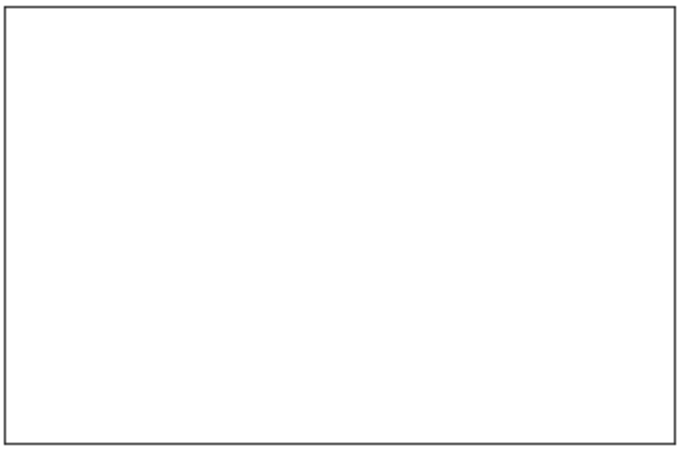
Threshold:28



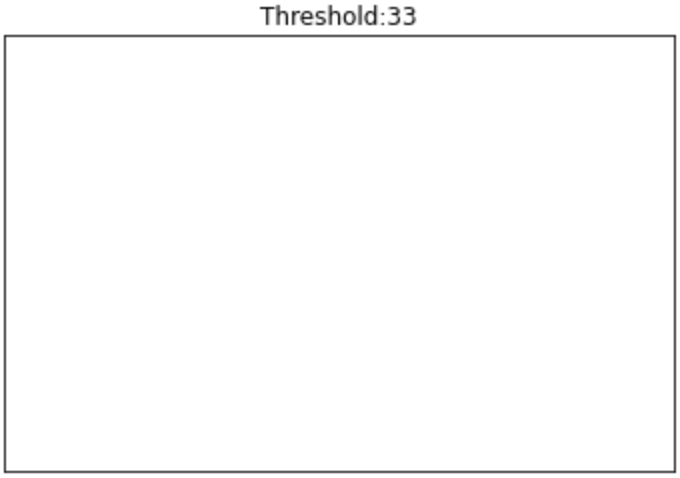
Threshold:29

Threshold:30

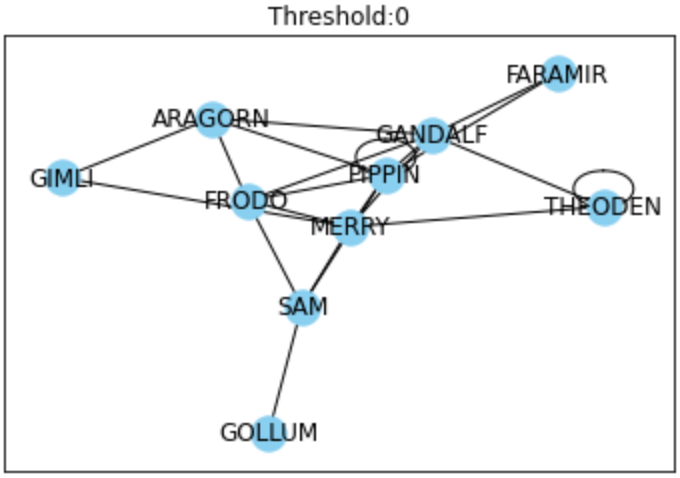
Threshold:31

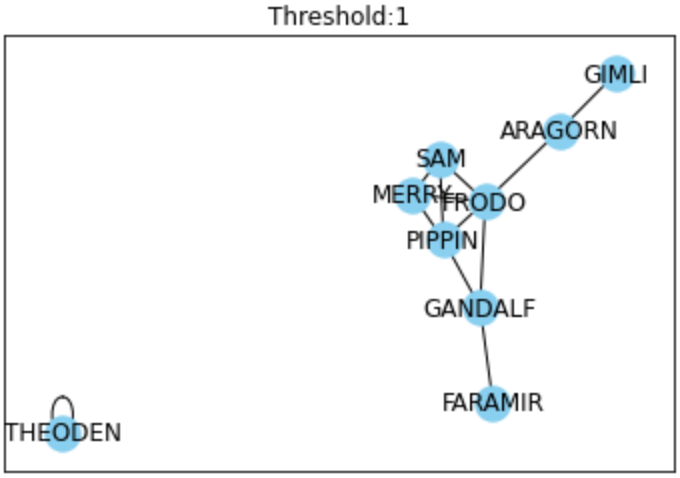


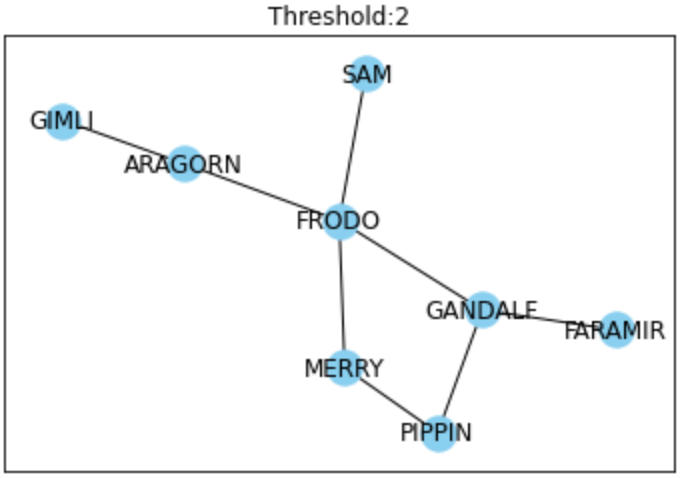
Threshold:32

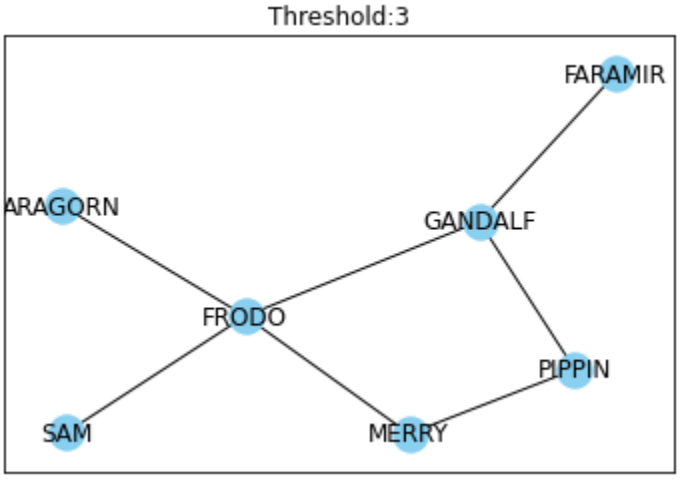


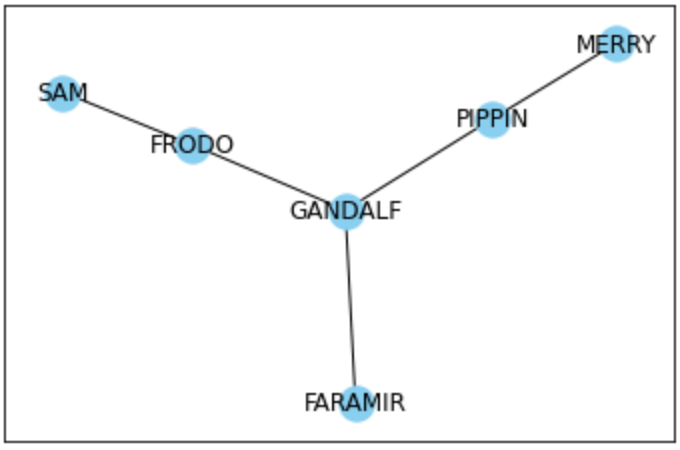
In [42]:
```
for i in range(34):
    get_relationships(return_king_df, return_king_df.char.unique().tolist(),
```

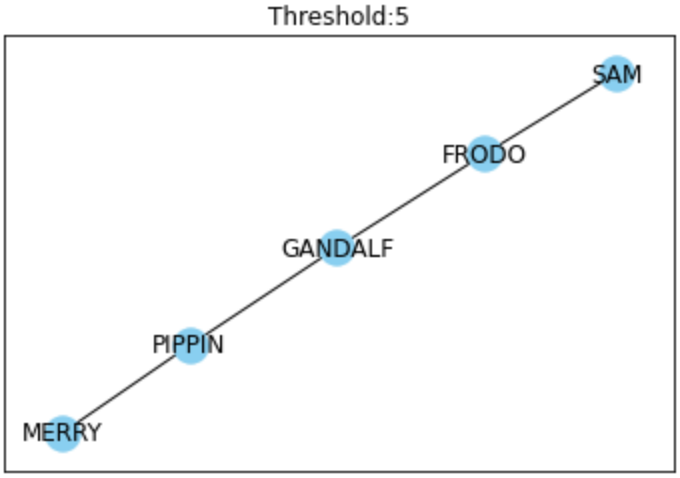


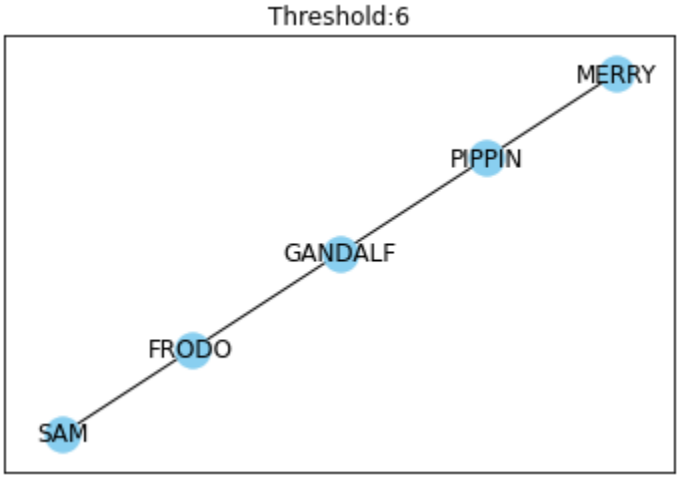


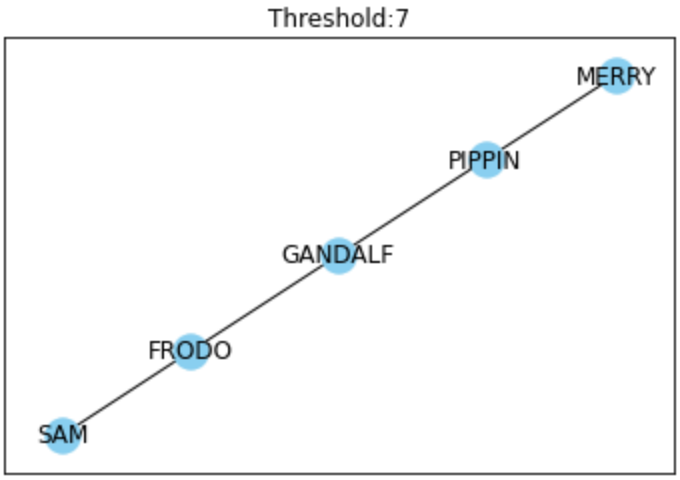

Threshold:2
SAM
GIMLI
ARAGORN
FRODO
GANDALF
FARAMIR
MERRY
PIPPIN

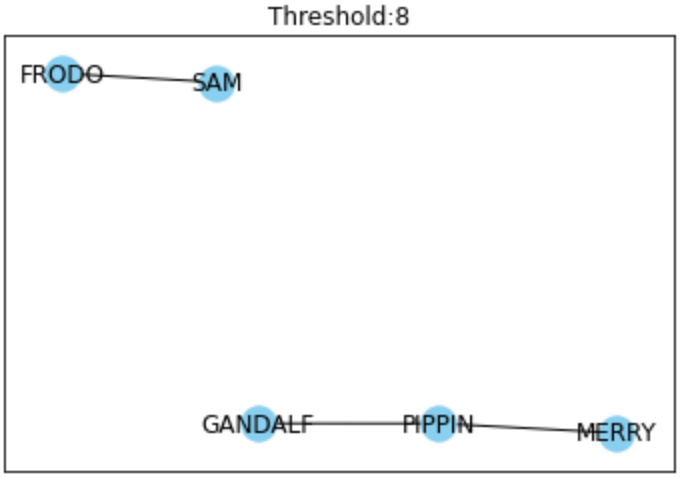

Threshold:3
FARAMIR
ARAGORN
GANDALF
FRODO
PIPPIN
SAM
MERRY

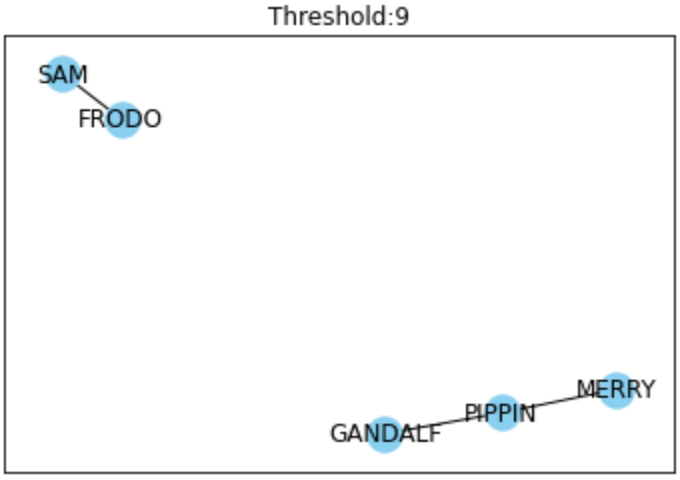

Threshold:4
MERRY
SAM
PIPPIN
FRODO
GANDALF
FARAMIR

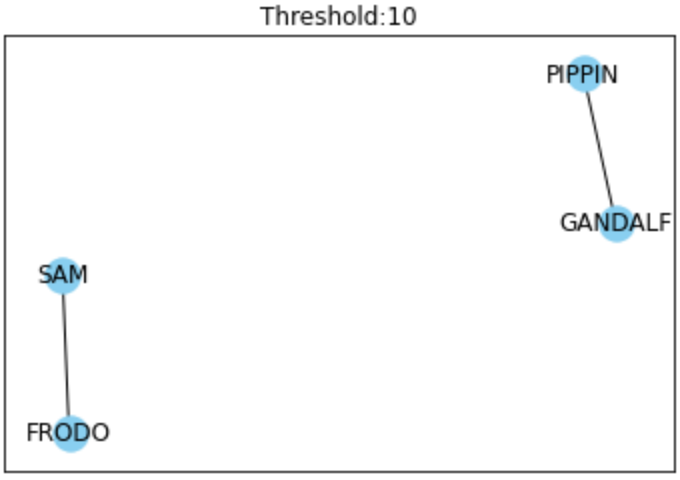

Threshold:5
SAM
FRODO
GANDALF
PIPPIN
MERRY

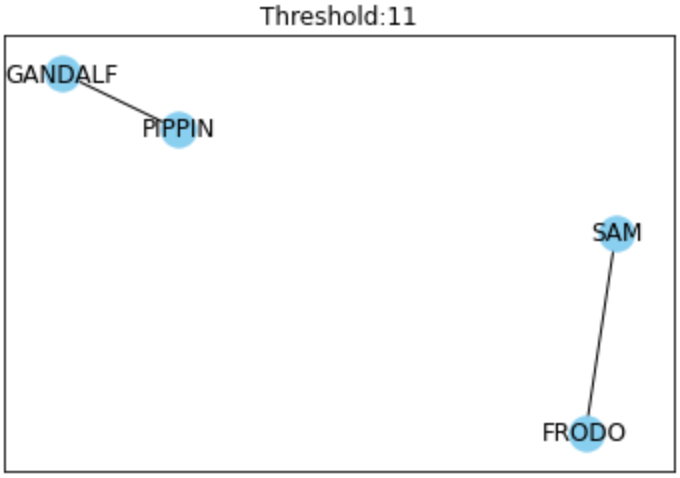

Threshold:6
MERRY
PIPPIN
GANDALF
FRODO
SAM

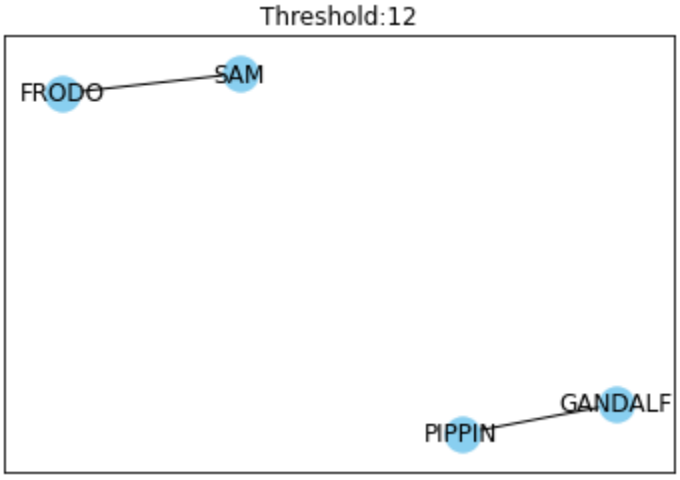

Threshold:7
MERRY
PIPPIN
GANDALF
FRODO
SAM

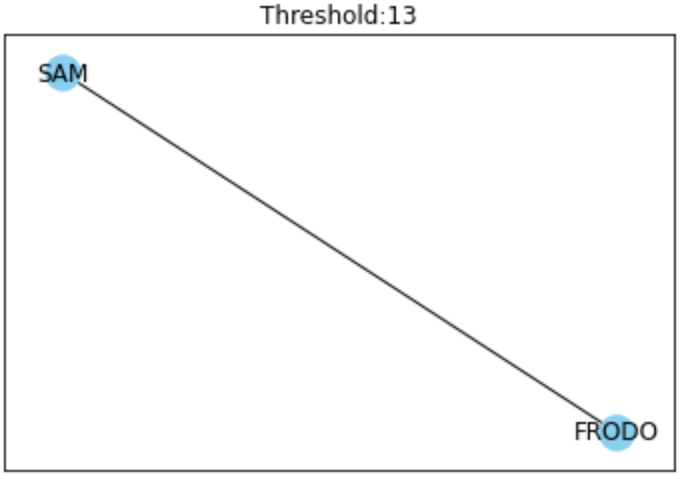

Threshold:8
FRODO
SAM
GANDALF
PIPPIN
MERRY

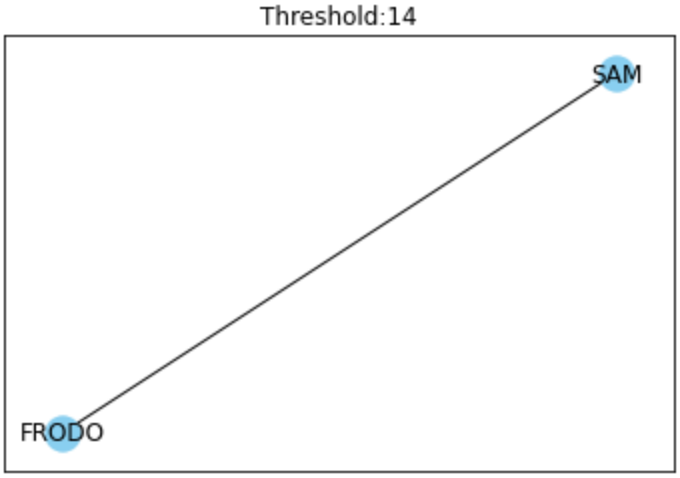

Threshold:9
SAM
FRODO
MERRY
PIPPIN
GANDALF

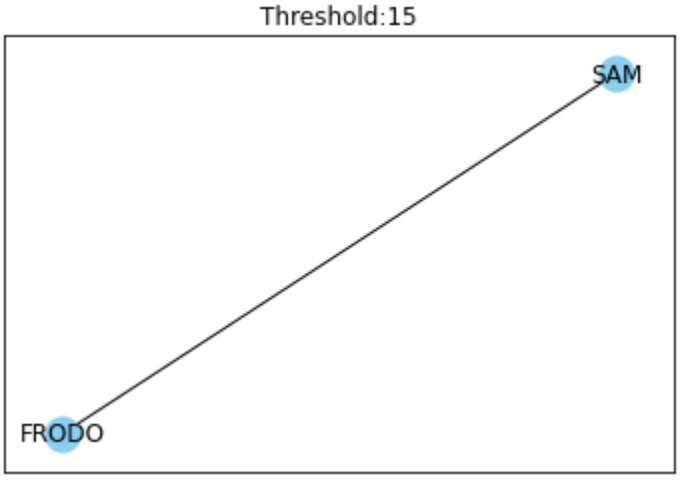

Threshold:10
PIPPIN
GANDALF
SAM
FRODO

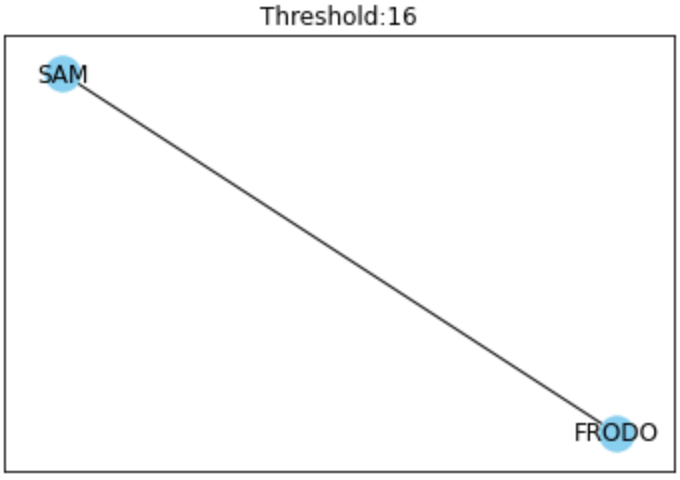

Threshold:11
GANDALF
PIPPIN
SAM
FRODO

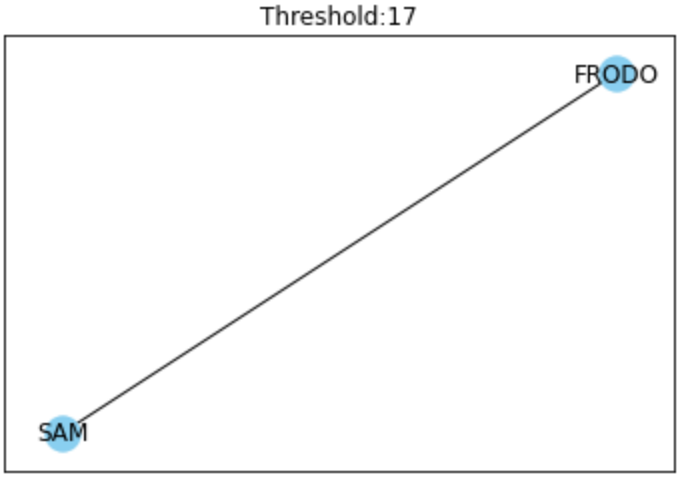

Threshold:12
FRODO
SAM
GANDALF
PIPPIN

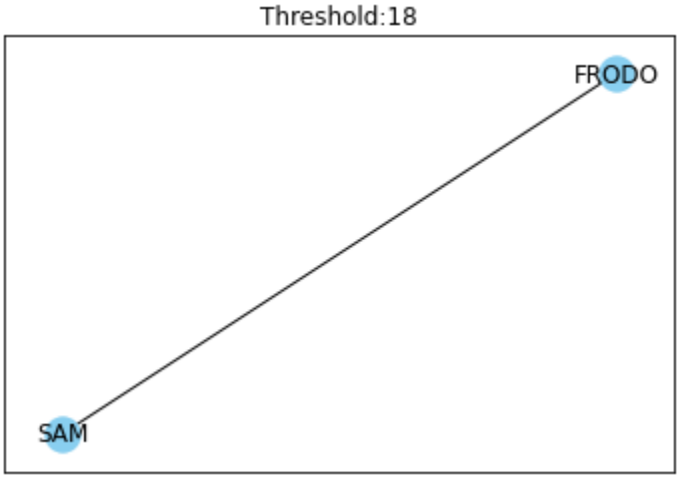

Threshold:13
SAM
FRODO

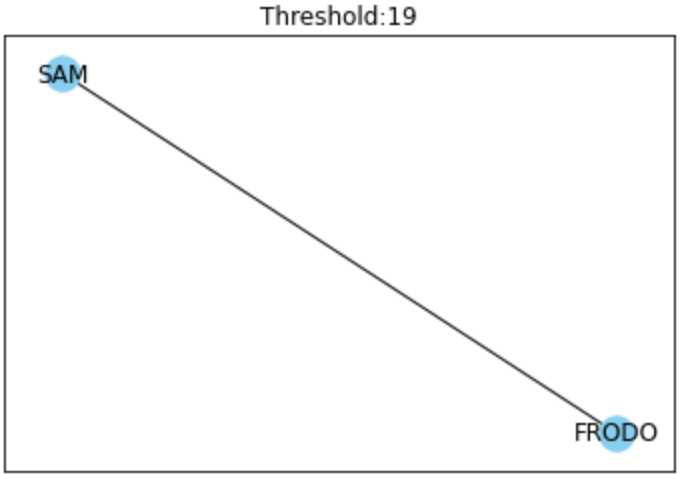

Threshold:14
SAM
FRODO

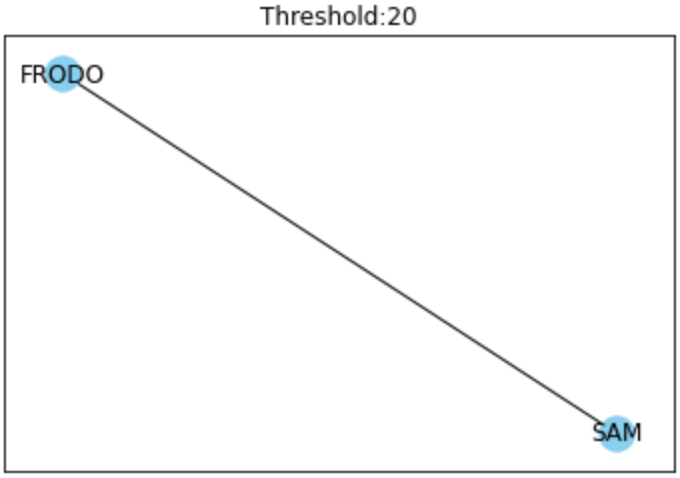

Threshold:15
SAM
FRODO

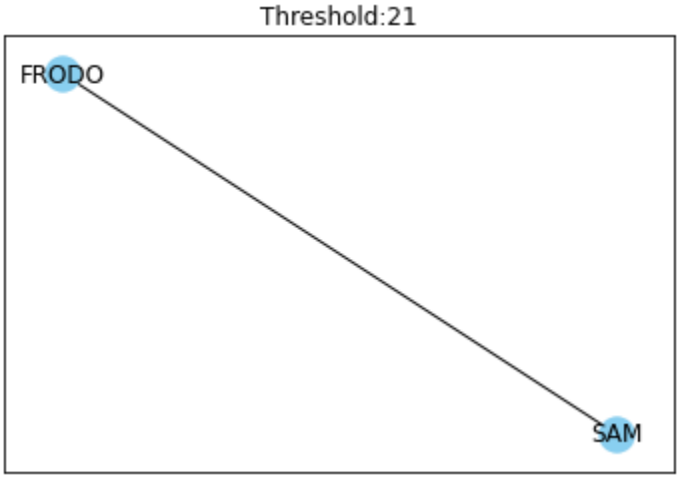

Threshold:16
SAM
FRODO

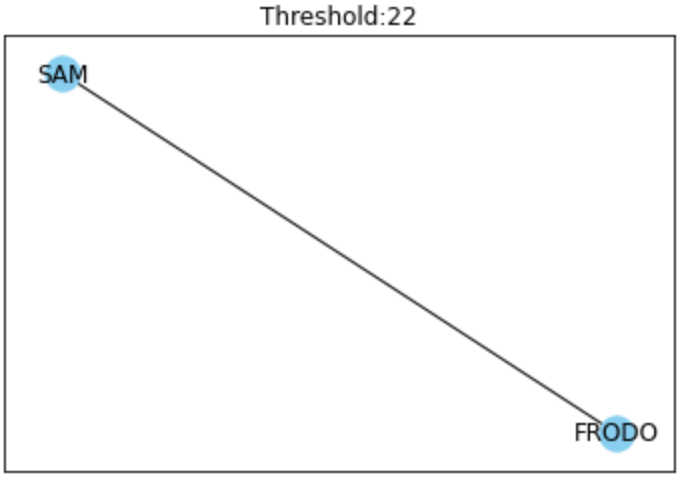

Threshold:17
FRODO
SAM

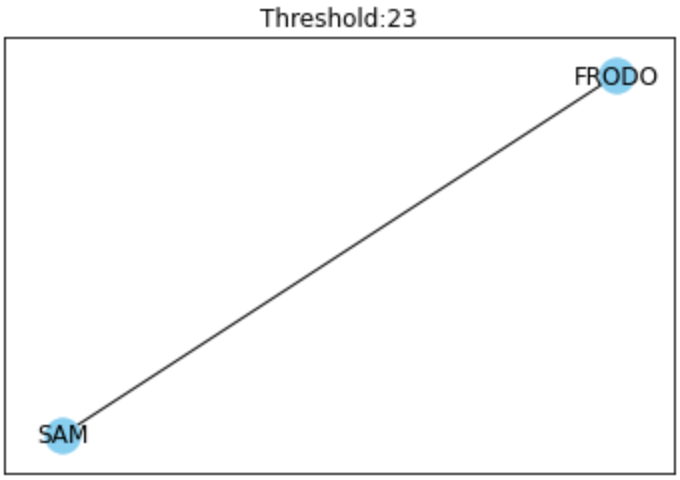

Threshold:18
FRODO
SAM

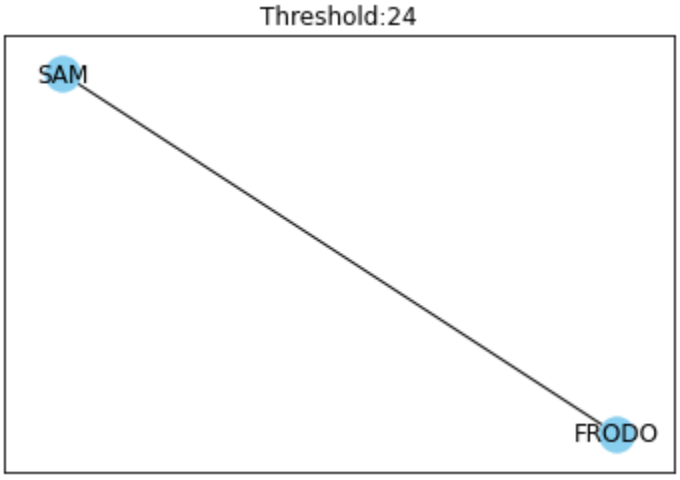

Threshold:19
SAM
FRODO

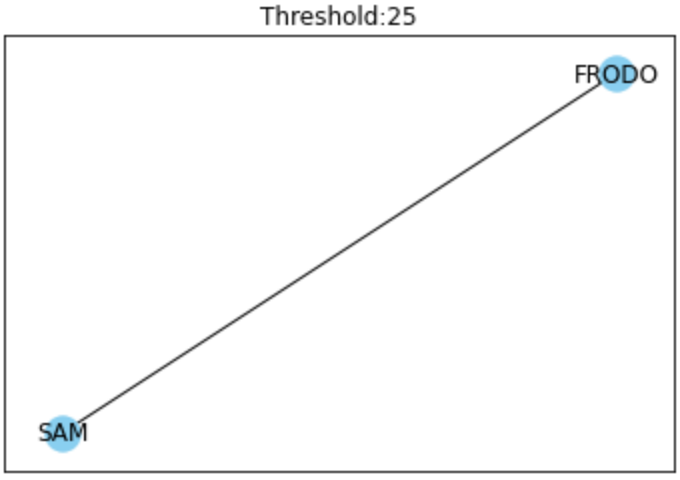

Threshold:20
FRODO
SAM


Threshold:21
FRODO
SAM

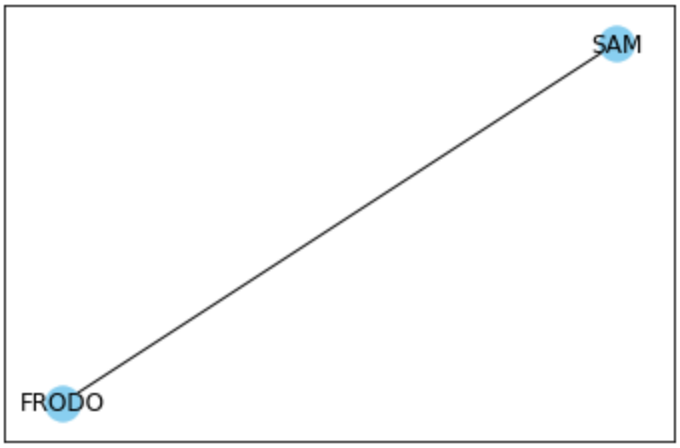

Threshold:22
SAM
FRODO


Threshold:23
FRODO
SAM

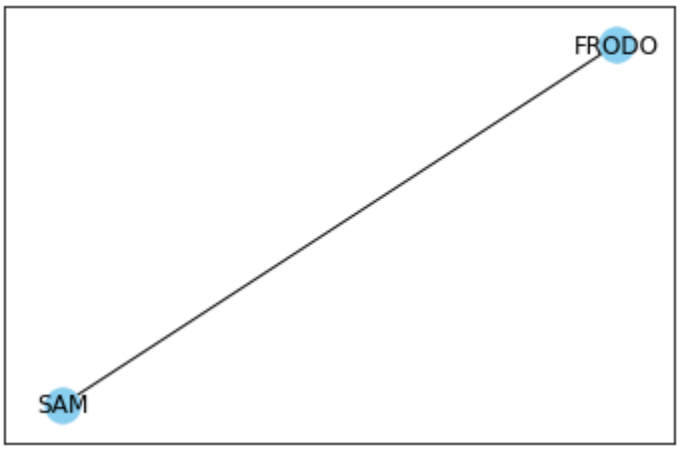

Threshold:24
SAM
FRODO


Threshold:25
FRODO
SAM

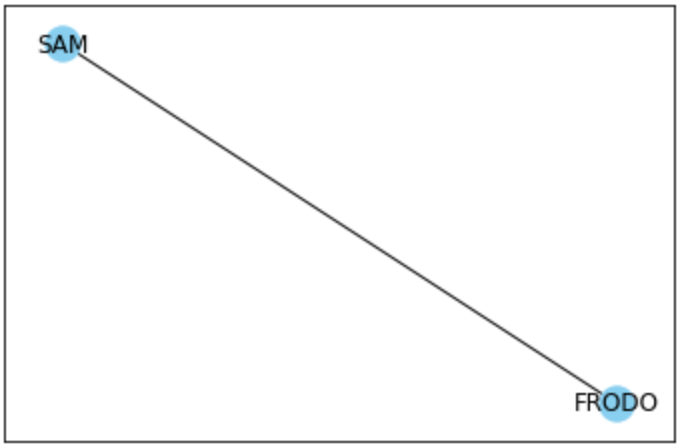
Threshold:26



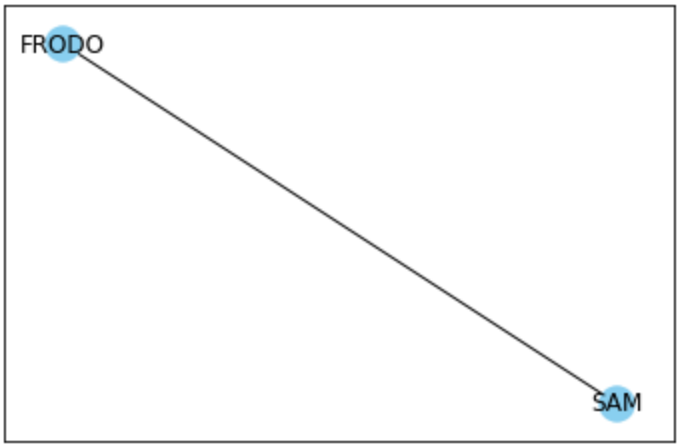
Threshold:27



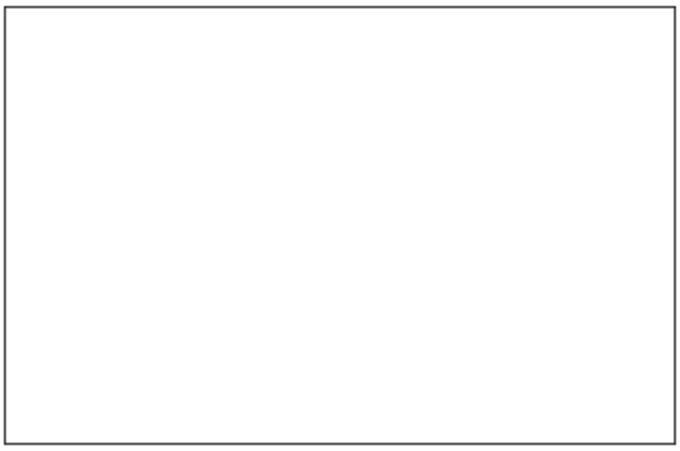
Threshold:28



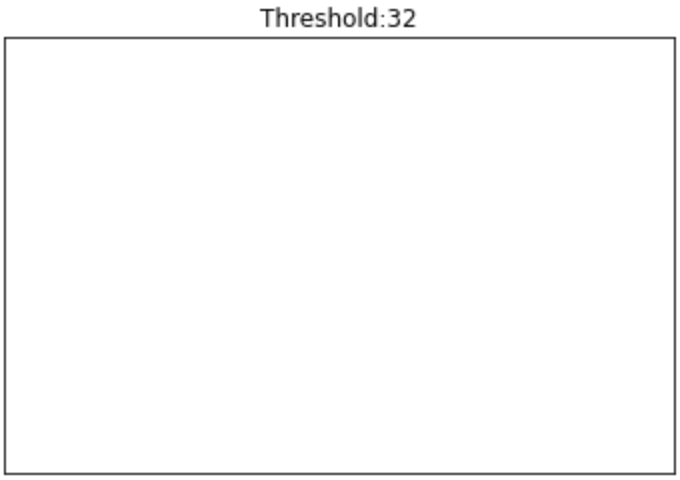
Threshold:29

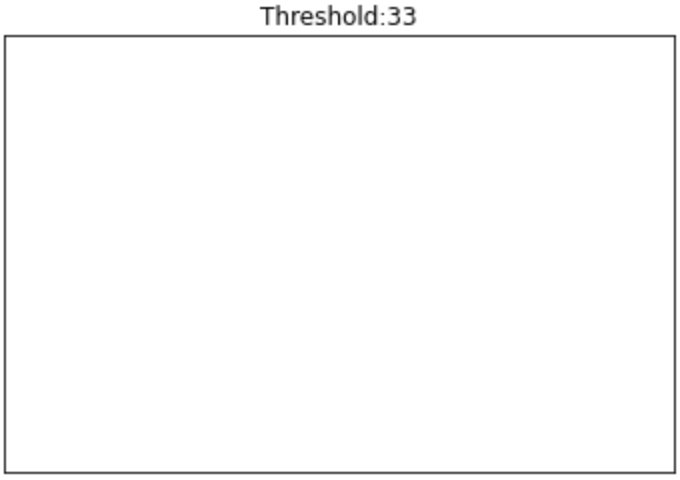


Threshold:30



Threshold:31





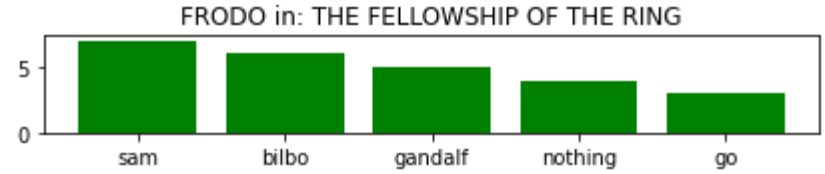
Now let us try to see if there was any character growth for the top 10 characters

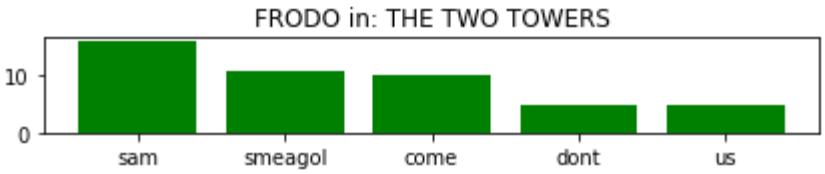
In [43]:
```
top10_chars_allmovies
```

Out[43]:
```
['FRODO',
 'SAM',
 'GANDALF',
 'ARAGORN',
 'PIPPIN',
 'MERRY',
 'GOLLUM',
 'GIMLI',
 'THEODEN',
 'FARAMIR']
```

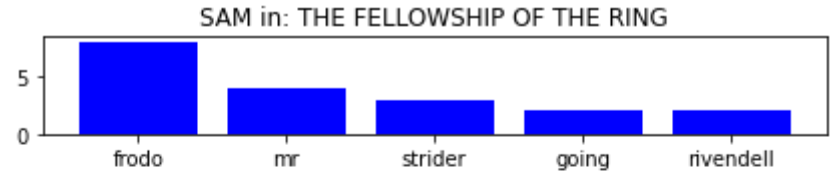
In [44]:
```
def get_char_growth(df_list, char, i):

    colors = ['green', 'blue', 'teal', 'indigo', 'limegreen',
          'mediumblue', 'fuchsia', 'cyan', 'coral', 'seagreen']

            # Get char vocabulary

    char_vocab_1 = consolidate_char_vocab(df_list[0], char)
    char_vocab_2 = consolidate_char_vocab(df_list[1], char)
    char_vocab_3 = consolidate_char_vocab(df_list[2], char)

    # Get top 5 words for character in each movie
    top5_words_1 = sorted(char_vocab_1, key=char_vocab_1.get, reverse=True)[
    top5_words_2 = sorted(char_vocab_2, key=char_vocab_2.get, reverse=True)[
    top5_words_3 = sorted(char_vocab_3, key=char_vocab_3.get, reverse=True)[

    # Get subset dictionary for the character based on top 5 words only
    subset_1 = {key: char_vocab_1[key] for key in top5_words_1}
    subset_2 = {key: char_vocab_2[key] for key in top5_words_2}
    subset_3 = {key: char_vocab_3[key] for key in top5_words_3}

    names_1 = list(subset_1.keys())
    names_2 = list(subset_2.keys())
    names_3 = list(subset_3.keys())

    values_1 = list(subset_1.values())
    values_2 = list(subset_2.values())
    values_3 = list(subset_3.values())

    figure, (ax1,ax2,ax3) = plt.subplots(3)

    # For first movie
    ax1.bar(range(len(subset_1)), values_1, tick_label=names_1, color=colors
    ax1.set_title(f"{char} in: {df_list[0].movie.unique()[0]}")


     # For second movie
    ax2.bar(range(len(subset_2)), values_2, tick_label=names_2, color=colors
    ax2.set_title(f"{char} in: {df_list[1].movie.unique()[0]}")

     # For third movie
    ax3.bar(range(len(subset_3)), values_3, tick_label=names_3, color=colors
    ax3.set_title(f"{char} in: {df_list[2].movie.unique()[0]}")
    figure.tight_layout()

    plt.show()
```

In [45]:
```
for i in range(len(top10_chars_allmovies)):
    get_char_growth([fellowship_df,two_towers_df, return_king_df], top10_cha
```


FRODO in: THE FELLOWSHIP OF THE RING
5
0
sam
bilbo
gandalf
nothing
go


FRODO in: THE TWO TOWERS
10
0
sam
smeagol
come
dont
us


FRODO in: THE RETURN OF THE KING
20
0
sam
im
cant
ring
go

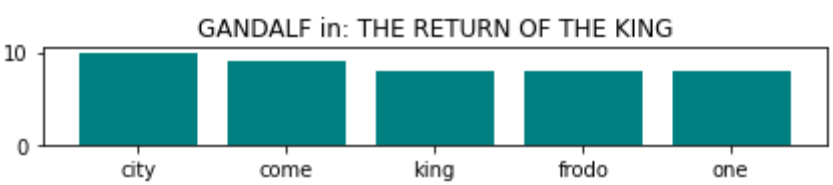

SAM in: THE FELLOWSHIP OF THE RING
5
0
frodo
mr
strider
going
rivendell

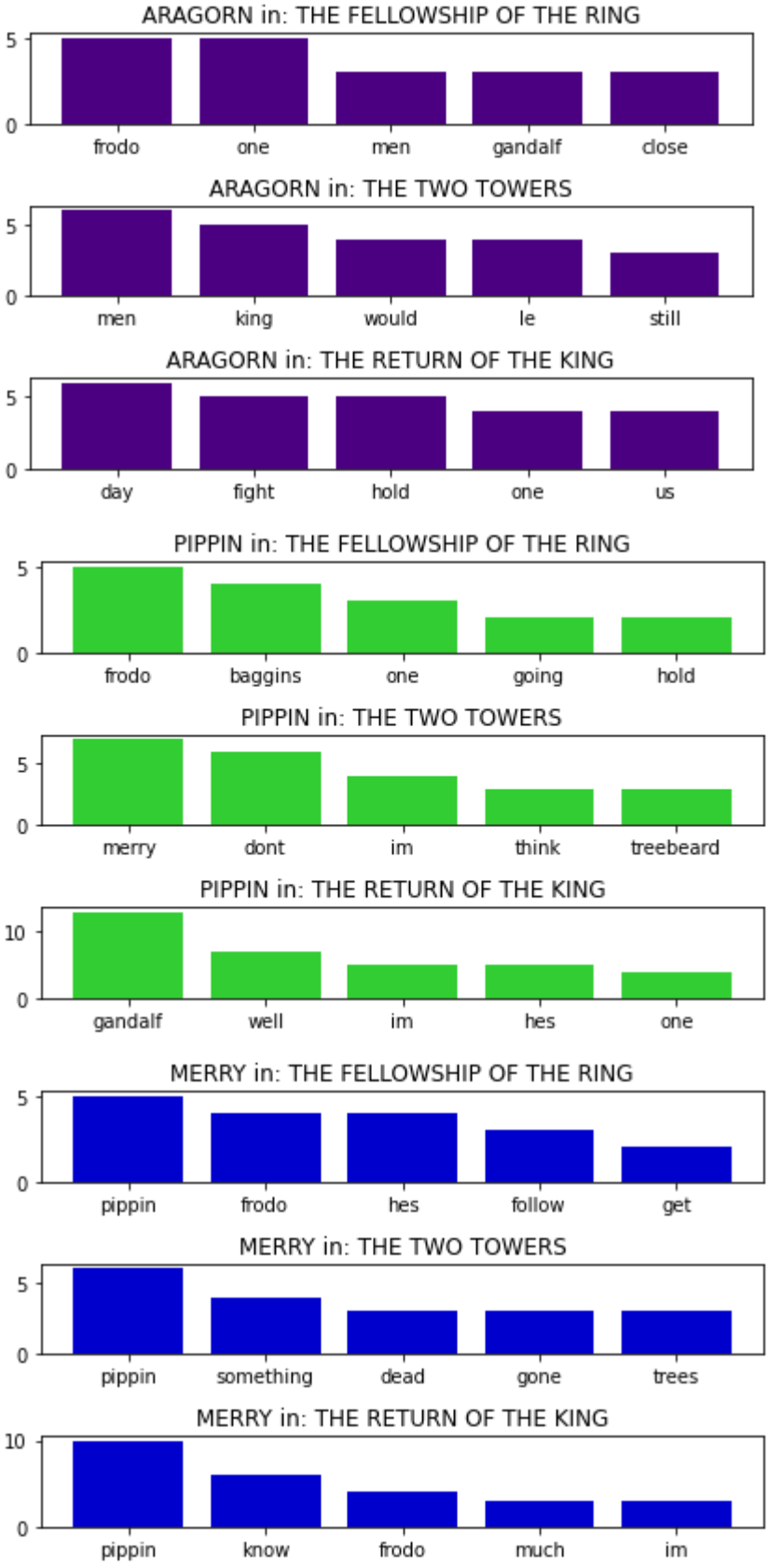

SAM in: THE TWO TOWERS
20
0
mr
frodo
know
come
dont

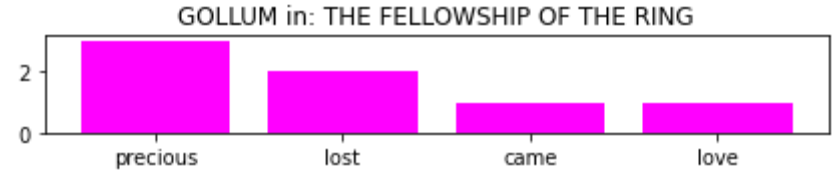

SAM in: THE RETURN OF THE KING
20
0
frodo
mr
dont
go
im


GANDALF in: THE FELLOWSHIP OF THE RING
5
0
hmm
must
frodo
ring
cannot


GANDALF in: THE TWO TOWERS
5
0
theoden
king
must
back
look


GANDALF in: THE RETURN OF THE KING
10
0
city
come
king
frodo
one


ARAGORN in: THE FELLOWSHIP OF THE RING
5
0
frodo
one
men
gandalf
close
ARAGORN in: THE TWO TOWERS
5
0
men
king
would
le
still
ARAGORN in: THE RETURN OF THE KING
5
0
day
fight
hold
one
us
PIPPIN in: THE FELLOWSHIP OF THE RING
5
0
frodo
baggins
one
going
hold
PIPPIN in: THE TWO TOWERS
5
0
merry
dont
im
think
treebeard
PIPPIN in: THE RETURN OF THE KING
10
0
gandalf
well
im
hes
one
MERRY in: THE FELLOWSHIP OF THE RING
5
0
pippin
frodo
hes
follow
get
MERRY in: THE TWO TOWERS
5
0
pippin
something
dead
gone
trees
MERRY in: THE RETURN OF THE KING
10
0
pippin
know
frodo
much
im


GOLLUM in: THE FELLOWSHIP OF THE RING
2
0
precious
lost
came
love

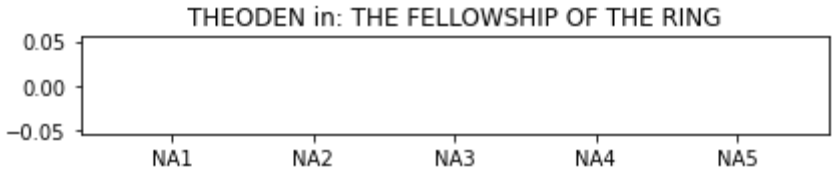

GOLLUM in: THE TWO TOWERS
10
0
master
us
precious
go
yes

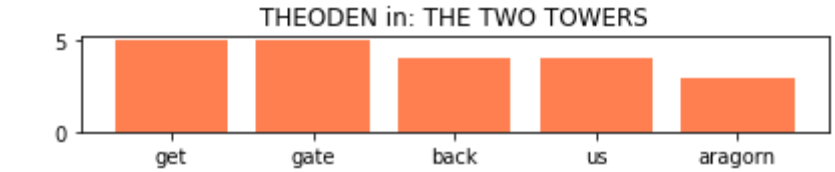

GOLLUM in: THE RETURN OF THE KING
10
0
smeagol
go
come
precious
must


GIMLI in: THE FELLOWSHIP OF THE RING
2.5
0.0
moria
one
dwarf
us
let

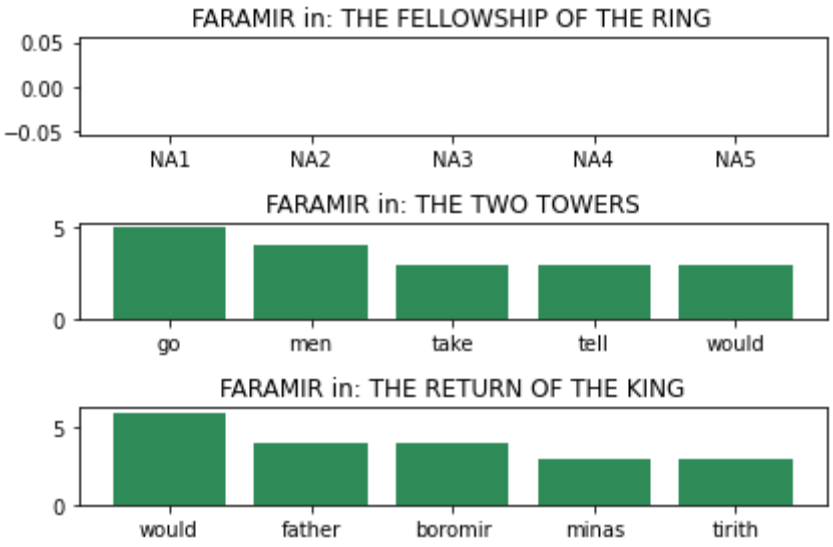

GIMLI in: THE TWO TOWERS
2.5
0.0
thats
come
tell
two
im


GIMLI in: THE RETURN OF THE KING
10
0
heh
well
oh
aragorn
us


THEODEN in: THE FELLOWSHIP OF THE RING
0.05
0.00
−0.05
NA1
NA2
NA3
NA4
NA5


THEODEN in: THE TWO TOWERS
5
0
get
gate
back
us
aragorn


THEODEN in: THE RETURN OF THE KING
10
0
ride
shall
take
peace
man


FARAMIR in: THE FELLOWSHIP OF THE RING
0.05
0.00
−0.05
NA1
NA2
NA3
NA4
NA5
FARAMIR in: THE TWO TOWERS
5
0
go
men
take
tell
would
FARAMIR in: THE RETURN OF THE KING
5
0
would
father
boromir
minas
tirith