This is a report of a mini-project for visualizing wikipedia conversations on summer 2022. The data was gathered by Christine De Kock and Andreas Vlachos. The code was written by me (Zahra Arjmandi Lari). Part of the credit for the ideas on "how to visualize" goes back to Tom Stafford.

The Story In Short

Wikipedia pages have a talk/discussion page where editors discuss how to improve a page. In some occasions, the editors have different opinions on some topics and the talk page will be tagged as "dispute" by editors. In some cases the editors resolve the disputes on their own; in other cases the disputes are "escalated" and needs mediation. Wikipedia suggests its users to use hierarchy of disagreement, proposed by Paul Graham 1 to resolve disputes constructively. Christine De Kock and Andreas Vlachos have annotated ~200 conversation from wikipedia talk pages categorized as "dispute". They have labeled each part of the conversation (utterance) accordingly which consist of ~4000 utterance 2. Part of the labels are on the basis of Graham's proposed hierarchy of disagreement 1 which addresses the "rebuttal tactics". Graham suggest 7 levels for disagreements which (starts from name-calling at the bottom or DH0 to refuting the central point or DH7). The other part of labels are called "resolution tactics" and attempts to promote understanding and consensus. See table 1 [ADD TABLE]. For more information see 2.

Our Question

What we wanted to know was "what is the difference in how conversation flows in"escalated" versus "non-escalated" discussions?"

What We Did In Summary

I divided data into escalated and non-escalated conversations. For each utterance, picked one or multiple label(s). Then, built the transition matrix and visualized the graphs; each label served as a node and the transition probability between each two nodes, became the strength of the edges. First, we wanted to know whether escalated conversation lack higher order DHs or not. I drew the graph such that in case of multi-label utterance, the higher label is shown (for more information see 1). Our graphs showed that escalated discussion do have at least the same amount of higher order DHs.[reference to related cells] Then we thought that maybe the higher order DHs are accompanied with lower order DHs in escalated conversations. Potentially, this could neutralize the effect of using higher order resolution techniques. [reference to related cells] Based on the graphs, it seems like in 'escalated' conversations lower order DHs were used more! In the end, we had the idea of dividing transitions such that each label has the chance to be part of the graph. It shows that less resolutional tactics were used in 'escalated' conversations, compared to 'non-escalated' ones. It also shows that 'rebuttal' tactics did not encourage 'resolutional' tactics (such as 'Asking questions' "providing a clarification", or "suggesting a compromise") in 'escalated' conversations versus 'non-escalated' ones.

##The Code

Import libraries. Numpy for handling matrices and json for handling the raw data file (opening and reading .json format). Pyvis and networkx are the two libraries for visualizing graphs. I have used IPython to help in displaying the graphs (.html format) inside this notebook.

```
library(reticulate)
py_install("pyvis")
```

```
## Using virtual environment '/cloud/project/r-reticulate' ...
```

```
## + '/cloud/project/r-reticulate/bin/python' -m pip install --upgrade --no-user 'pyvis'
```

```
#import needed libraries
import numpy as np
import json

from collections import Counter
```

```
# import pyvis
# install pyvis
# pyvis is a network visualization library based on Networkx
#which allow graphs to be interactive. You can also save them as html
#using pip in case the code is directly run under python
# !pip install pyvis
from pyvis.network import Network

# # to be able to display output graph which is in html format
# from IPython.core.display import display, HTML
```

```
# Opening JSON file
raw_data = open('frequency_data.json')

# returns JSON object as
# a dictionary
raw_data = json.load(raw_data)

raw_data[0:5]
```

```
## [{'utt_labels': [['Coordinating edits'], ['Contextualisation'], ['Providing clarification'], ['DH5: C
```

```
#count frequency of each label in a dictionary


super_dict_labels = {}
for i in range(len(raw_data)):


  dict_list = ([dict(Counter(x)) for x in raw_data[i]['utt_labels']])
  for item in dict_list:
    for k, v in item.items():
      if k in super_dict_labels.keys():
        super_dict_labels[k] += v
      else:
        super_dict_labels[k] = v
super_dict_labels
```

```
## {'Coordinating edits': 972, 'Contextualisation': 208, 'Providing clarification': 144, 'DH5: Counterar
```

```
# change label "DH-1: Bailing out" to formal format "DH1: Bailing out"
for conversation in raw_data:
  for labels in conversation['utt_labels']:
    #check if DH- is there
    if "DH-1: Bailing out" in labels:
      idx = labels.index("DH-1: Bailing out")
      labels[idx] = "DH1: Bailing out"


#count frequency of each label in a dictionary after changing DH-1 to DH1

super_dict_labels = {}
for i in range(len(raw_data)):
```

```
  dict_list = ([dict(Counter(x)) for x in raw_data[i]['utt_labels']])
  for item in dict_list:
    for k, v in item.items():
      if k in super_dict_labels.keys():
        super_dict_labels[k] += v
      else:
        super_dict_labels[k] = v
super_dict_labels
```

```
## {'Coordinating edits': 972, 'Contextualisation': 208, 'Providing clarification': 144, 'DH5: Countera
```