**# Document Retrieval (Clustering)**

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

**# Load dataset**

import graphlab as gl

gl.canvas.set\_target('ipynb')

people = gl.SFrame('people\_wiki.gl')

people.head()

**# Take a look at a few examples**

obama = people[people['name'] == 'Barack Obama']

obama['text']

clooney = people[people['name'] == 'George Clooney']

**# Let's get some basic word counts**

obama['word\_count'] = gl.text\_analytics.count\_words(obama['text'])

**# Create a word count table for Obama article**

obama\_word\_count\_table = obama[['word\_count']].stack('word\_count',new\_column\_name =['word', 'count'])

obama\_word\_count\_table.sort('count', ascending = 0) **# There's a lot of un-interesting words**

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

**# Compute TF-IDF**

people['word\_count'] = gl.text\_analytics.count\_words(people['text']) **# Create a dictionary of word counts for all a people**

tfidf = gl.text\_analytics.tf\_idf(people['word\_count']) **# Run TF IDF built in func over all WIKI articles**

people['tfidf'] = tfidf **# Store tfidf dict array in a column called TFIDF**

obama = people[people['name'] == 'Barack Obama'] **# Refresh the obama table**

**# Pulls the most interesting words from the Obama article**

**# Python - chain command of stack and sort**

**obama[['tfidf']].stack('tfidf', new\_column\_name =['word','tfidf']).sort('tfidf', ascending =0)**

**# Manually compute distances between a few people**

**# Let's take three people - Obama, Clinton, Beckham**

clinton = people[people['name']== 'Bill Clinton']

clinton[['tfidf']].stack('tfidf', new\_column\_name=['word','tfidf']).sort('tfidf',ascending=0)

beckham = people[people['name']== 'David Beckham']

beckham[['tfidf']].stack('tfidf', new\_column\_name=['word','tfidf']).sort('tfidf',ascending=0)

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

**# Similarity measurement using distances -Is Obama closer to Clinton (or) Beckham?**

# Using cosine distance - smaller = more similar. Euclidean distance is another option

# Obama vs Clinton = 0.83

**gl.distances.cosine(obama['tfidf'][0], clinton['tfidf'][0]) # Calling built in similarity distance function**

# Obama vs Beckham = 0.979

gl.distances.cosine(obama['tfidf'][0], beckham['tfidf'][0])

# Beckham vs Clinton = 0.969

gl.distances.cosine(beckham['tfidf'][0], clinton['tfidf'][0])

# Obama vs Nelson = 0.99

gl.distances.cosine(nelson['tfidf'][0], obama['tfidf'][0])

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

**# Nearest neighbor model - relative distances for all articles in the Corpus**

# Use built in func "nearest\_neighbors"

**knn\_model = gl.nearest\_neighbors.create(people, features = ['tfidf'],label = 'name')**

# Label allows the results to be listed by name instead of a reference label number for each person

**# Apply the model for retrieval**

# Who is closest to Obama?

**# Command gives us the closest neighbors of query name**

knn\_model.query(obama)

knn\_model.query(beckham)

knn\_model.query(clinton)

**# Playing around with other queries**

swift = people[people['name'] == 'Taylor Swift']

knn\_model.query(swift) # Carrier Underwood is her closest neighbor!!!

jolie = people[people['name'] == 'Angelina Jolie']

knn\_model.query(jolie) # Brad Pitt and then actors who have been in movies with Angelina Jolie

arnold = people[people['name']== 'Arnold Schwarzenegger']

knn\_model.query(arnold) # Combo of wrestlers, body builders and politicians due to the dual careers of Arnold

**# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**

**# End of module assignment**

**# 1) Compare top words according to word counts to TF-IDF for Elton John**

elton = people[people['name']== 'Elton John']

elton\_word\_count\_table = elton[['word\_count']].stack('word\_count',new\_column\_name = ['word', 'count'])

elton\_word\_count\_table.sort('count', ascending =0)

# Top 3 words for WORD\_COUNT is 'the', 'in, 'and'

elton\_tfidf\_table = elton[['tfidf']].stack('tfidf', new\_column\_name = ['tfidf','count'])

elton\_tfidf\_table.sort('count', ascending = 0)

# Top 3 words for TFIDF is 'furnish', 'elton', 'bilboard'

**# 2) What's the distance between Elton John and Victoria Beckham and Paul McCartney**

victoria = people[people['name'] == 'Victoria Beckham']

gl.distances.cosine(elton['tfidf'][0], victoria ['tfidf'][0]) # 0.95 - Elton is farther from Victoria B than McCartney

mccartney = people[people['name'] == 'Paul McCartney']

gl.distances.cosine(elton['tfidf'][0], mccartney['tfidf'][0]) # 0.82 - Elton is closer to McCartney than Victoria B

**# 3) Build NN models with diff input features (word count and TFIDF) and set distance metric**

**knn\_model\_word\_count = gl.nearest\_neighbors.create(people, features = ['word\_count'],label = 'name',distance='cosine')**

knn\_model\_word\_count.query(elton)

**knn\_model\_tfidf = gl.nearest\_neighbors.create(people, features = ['tfidf'],label = 'name',distance='cosine')**

knn\_model\_tfidf.query(elton)

**# Closest neighbor to Victoria Beckham using word count and TFIDF with cosine distance metric**

knn\_model\_word\_count.query(victoria) # Mary Fitzgerald

knn\_model\_tfidf.query(victoria) # David Beckham