Project Fletcher
Will Hetfield

Executive Summary

1. **Project is NLP for different countries constitutions**
2. **API of Comparative Constitutions Projects to get basic info**
3. **Web scrape and BeautifulSoup to get 206 different English Translation Constitutions**
4. **Analyze at the clause level (defined as new line break in HTML and at least 5 words)**
5. **NMF and LDA Topic Modeling**
6. **Subset Clauses by TextBlob subjectivity score and perform Topic Modeling based on subjectivity score**
7. **Function to take any constitution's clause and return the most similar clause from another country's constitution**
8. **Aggregate clauses back to the constitution level by:**
    - **I. Normalize the NMF clause - topic vectors**
    - **II. Sum each of 206 constitutions**
    - **III. Normalize again so each constitution is now a 10 feature topic vectors, with all 10 features summing to 1 for every constitution**
9. **Calculate most similar constitution to another constitution based on the 10-feature topic vectors**
10. **Map the constitution vectors to 2 features through PCA**
11. **Graph the two features in Tableau**

Background / Motivation

I wanted to see if there were universal concepts that could be captured in different countries' constitutional documents. I would expect to see a lot of similar ideas in Western Constitutions with individual rights and government structures, but can they be captured in NLP and do they extend to non-Western world.

Data_Acquisition

Most of the data acquisition and cleaning is done through fletcher.py module functions. I found a non-profit online which gathers and translates to English different countries constitutions.
https://www.constituteproject.org/
Provided API
https://docs.google.com/document/d/1wATS_IAcOpNZKzMrvO8SMmjCgOZfgH97gmPedVxpMfw/pub

They don't allow clean versions of the constitutions to be downloaded en masse but they have a Basic API to get a JSON with information for each constitution. The IDs in the JSON corresponded to the web address and then it was straightforward to scrape and clean with

BeautifulSoup. [Random note, in the code snippets, the years in the country IDs are when the constitutions were last amended]

I elected to analyze the constitutions at the clause level. I thought this made sense to have more shorter, observations as the themes would be more clearly separable based on the terms used in specific clauses in dealing with say, elections versus religious rights.

Vectorizing the Text

In order to analyze textual data, we need to vectorize it / make it numeric. The basic model is the bag of words model, where we turn each individual document (the term for an observation in the NLP context) into a vector based on word counts in the document. Each column can be a word in the document, and we simply register a 1 or 0 if the particular word is in the observation / document.

This results in a very large, sparse matrix of mostly zeroes as only very few of the words will be any given document / observation. I used a few techniques for what "counts" as a term to reduce the size of the document term matrix and give it more meaningful features

Stop-words: Very common words which (a,an,the) which can be eliminated from every document as they don't provide any meaning

Stemming: Transforming variations of the same word into a single root running, ran, run -> run

Minimum document frequency: Require a minimum number occurrences within all documents in order for a term to be included in all documents

Topic Modeling

I wanted to look if there were meaningful themes / topics that the clauses could be be grouped into. I reduced the dimensionality of the document-term matrix to look for meaningful topics through two methods, non-negative matrix factorization (NMF) and Latent Dirichlet Allocation (LDA).

I went through various iterations of looking 5,10,20 topics , with and without stemming, and adjusting the minimum required occurence of a term in the documents for inclusion in the document term matrix. NMF with ten topics ended up giving what I felt were the most interpretable topics.

```
tfidf = TfidfVectorizer(stop_words='english', min_df= .01, ngram_range=(1,3))
bag_tfidif = tfidf.fit_transform(np.array(df_cons['Stemmed_Clause']))
nmf_model = NMF(n_components=10, random_state=47,beta_loss='frobenius')
nmf_topics = nmf_model.fit_transform(bag_tfidif)
bag_tfidif.shape
```

```
(176787, 254)
```

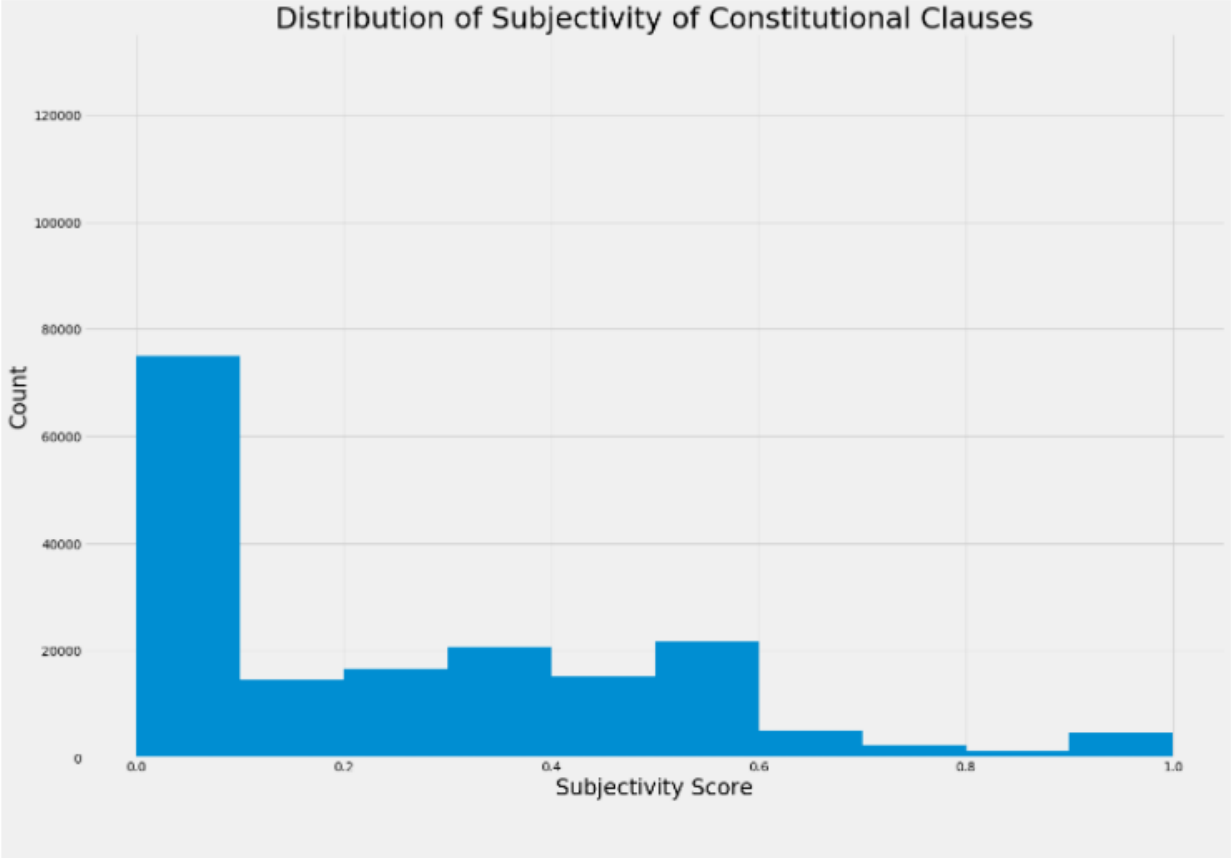
NMF is a linear algebra numeric method applied to a Term Frequency Inverse Document Frequency Matrix. It finds two matrices whose product is approximately equal to the original matrix, with the restriction that no terms in any of the three matrices can be negative. In the document - term matrix context, the matrix is factored into two matrices, a topics matrix of documents (rows) by topics (columns), where the number of topics is specified for the algorithm, and a topic by terms matrix, which can be viewed as the weights of each term in the topic space. There are different cost functions / optimization constraints which can be used to calculate the best matrix product. In this, case the Frobenius norm was used, which can be thought of as minimizing the squared difference in a matrix space (norm) between the original document-term matrix and the product of the two factors from NMF.

```
Topic 1:
member elect shall parliament hous vote repres year day council commiss speaker senat elector major hous repres assem
bl candid number appoint parti deputi term period committe follow meet ha session held
Topic 2:
offic public minist appoint servic hi gener commiss function prime prime minist hold remov act public servic servic c
ommiss governor perform term governor gener shall judg judici chief exercis power duti deputi advic govern
Topic 3:
law state shall govern establish organ regul state shall feder provid public accord law shall author power determin a
dministr local forc accord law protect territori condit intern develop social execut respect institut procedur
Topic 4:
court suprem judg suprem court appeal high high court justic jurisdict court shall constitut court decis proceed chie
f judici order shall case tribun befor constitut determin administr rule appoint crimin matter question refer ha
Topic 5:
articl thi articl paragraph thi claus provis refer appli govern freedom legisl feder provid term accord specifi forc
schedul power principl applic purpos concern matter amend relat effect pass council compet
Topic 6:
right freedom protect citizen human ha guarante person everi properti duti equal exercis polit public social hi shall
individu respect work educ secur chapter peopl inform legal cultur limit oblig
Topic 7:
nation assembl nation assembl council govern legisl peopl session approv secur territori deputi develop minist commit
te propos power meet polici report vote resolut senat execut submit independ budget countri pass major
Topic 8:
presid republ presid republ vice vice presid council minist presid shall constitut day senat shall deputi elect hi pr
ime prime minist term govern propos chamber case submit head declar appoint territori execut follow intern
Topic 9:
ani person ani person proceed hi ha author offenc ani law subsect reason time purpos relat person shall order act req
uir subject make includ case respect matter question befor crimin provis wa entitl
Topic 10:
constitut thi thi constitut section provis act thi section provis thi amend subject appli accord forc subsect constit
ut law refer power purpos make effect thi articl chapter parliament schedul constitut court relat exercis paragraph m
ean provid
```

Particularly interpretable topics include Topic 1 (Legislative Branch / Elections), Topic 4 (Judiciary) , Topic 6 (Individual / Personal Rights).

Potential for better topic models through Subjectivity Scores

I thought perhaps Subjectivity Scores from TextBlob may be useful, as clauses with zero subjectivity perhaps deal with certain topics and more subjective statements with other topics. However, this didn't improve the interpretability of the topics produced from topic modeling



Function for Most Similar Clause from a different country's constitution

To explore to what extent topics were being captured in the clauses versus the writing style of each constitution, I wrote a function which took my data frame, a clause-topic matrix and the index of a specific clause within the dataframe to return the most similar clause (based on cosine similarity) from a different country's constitution. Below is a clause from Article 3 of Mexico's constitution

```
similar_clause(df_cons,nmf_topics,84194)
```

```
Mexico_2015
Education provided by the State shall develop harmoniously all human abil
ities and will stimulate in pupils the love for the country, respect for
human rights and the principles of international solidarity, independence
and justice.


Most Similar Clause from Another Country
Eritrea_1997
1. The State shall strive to create opportunities to ensure the fulfillme
nt of citizens' rights to social justice and economic development and to
fulfill their material and spiritual needs.
```

Aggregate Clause-Topic Vectors to the Constitutional Level

Next I aggregated back up to the 206 constitutions from the 175,000 clauses. First I normalized each clause topic - vector by summing all the elements of the vector and dividing by 1, then I summed all the vectors of each constitution to give 206 constitution vectors. Last, there were normalized as well so all 10-features summed to 1. I only explored a bit, and obviously with enough pairs, you would expect to have find some interesting matches just by chance, but this was pretty cool:

```
In [269]: topic_weights_df.loc['Cuba_2002',
                               'Nearest_Constitution']
Out[269]: 'China_2004'
```

I wanted to be able to explore the constitutions more for clustering, so projected it into two dimensions through PCA. Based on just scrolling around on Tableau, there do appear to be some geographic clusters. It would be interesting to shade by continent.

Next Steps

A less rigid definition of clause would very likely help and allow for perhaps more interpretable topics to be achieved. Ideally you would look at each constitution and see its structure as far as clauses/sub clauses and write custom function in how it was split.

Incorporating a time component would also be interesting to see if there is a hierarchical structure as far as influence over time. This would be tricky as “old” original constitutions have “newer” clauses through amendments.