Final Project: Proposal
Topic modeling and visualization of news comments

Andy Chen, Pushkar Sinha, Maria Babaeva

1 Motivation and Background

With the massive amount of text data world produces every day, the thought of knowing enough about it without having to read even a single paragraph is inherently amazing. Further, as our results will be used by the people who have provided us with this data - it was practically inviting. It's not an unknown fact that NLP can help us do some amazing stuffs, but the field is still far from being well defined in terms of problems and their solutions.This project starting from topic modeling has introduced us to certain depths of NLP like sentiment analysis and NER (Named Entity Recognition). As this area is hot and many related projects have been done in the near past, we were able to find good tools to quickly wade through the tough parts, making this project even more interesting.

Our data came from SFU linguistic department, the SFU Opinion and Comments Corpus data set. The corpus contains 10,339 opinion articles (editorials, columns, and op-eds) together with their 663,173 comments from 303,665 comment threads, from the main Canadian daily in English, *The Globe and Mail*, for a five-year period (from January 2012 to December 2016).

The data is divided into two main parts, with each part being, in turn, divided into three portions:

- Raw data: The articles corpus, the comments corpus, and comment-threads corpus.
- Annotated data: SFU constructiveness and toxicity corpus, SFU negation corpus, SFU Appraisal corpus

The corpus is larger than any other currently available comments corpora, and has been collected with attention to preserving reply structures and other metadata. In addition to the raw corpus, we will also have annotations for four different phenomena: constructiveness, toxicity, negation and its scope, and appraisal .
The corpus can be used to study, among other aspects, the connections between articles and comments; the connections of comments to each other; the types of topics discussed in comments; the nice (constructive) or mean (toxic) ways in which commenters respond to each other; and how language is used to convey very specific types of evaluation. Our current focus is the study of constructiveness and evaluation in the comments.

Once our project is complete it will help people:
- to best organize comments to encourage constructive and civil conversations online.
- to do the comments express varying views on issues and policies.
- to find out the most popular view on the articles using its comments and the overlapping topics of the two.
- to find out the most informative and constructive view supported by evidence.
- to create a succinct summary of different views on an article or issues or policies in the article

- to engage people more in emotionally-driven conversation or in discussion based on facts and evidence
- to understand how often do commenters target authors and other commenters, rather than the arguments or issues in the article
- to understand how common is sarcasm in online comments.
- to generate further summarized documents using the constructive comments over the articles.

2 Problem Statement

In our project we were able to answer the following questions:

- What are the top topics mentioned in the articles and comments? What is the overlap?
- What are the most mentioned people, places, organizations in the articles and comments? What is the overlap?
- What are the sentiments expressed in the comments for the articles? Time-series analysis of the sentiments expressed in articles and comments.
- Clustering similar authors based on the named entities recognized.

Those tasks were challenging because of the size of data which we had to analyse. It was too big and required different and tricky approaches and being flexible. Plus there are different methods and techniques existed now so we needed to choose one which would perform the best and would show the better accuracy.

3 Data Science Pipeline

- Data Collection: We are going to download all data from: https://researchdata.sfu.ca/islandora/object/islandora%3A9109
  And use information related to that data from:
  https://github.com/sfu-discourse-lab/SOCC where we are able to find the following files:
  - gnm_articles.csv which contains information about The Globe and Mail 10,339 articles in our dataset.
  - gnm_comments.csv which contains all unique comments (663,173 comments) in response to the articles in gnm_articles.csv after removing duplicates and comments with large overlap.
  - gnm_comment_threads.csv which contains all unique comment threads -- a total of 303,665 unique comment threads in response to the articles in the gnm_articles.csv.
  - SFU_constructiveness_toxicity_corpus.csv which is a subset of SOCC for constructiveness and toxicity.
  - The negation and appraisal annotations were performed using WebAnno software.

4 Topic Modeling

4.1 Top Topics in Comments

We use **PySpark MLlib**, LDA in **gensim**, and NMF in **SciKit-Learn** to find the top topics in comments. The following steps were performed:

- Normalization, lemmatization, cleaning from stopwords, and tokenization using NLTK (Stopwords, WordNetLemmatizer) and PySpark.ML (StopWordsRemover, Tokenizer) libraries. After that using Pandas dataframe a cleaning.csv file was generated which could be used for any model.
- Term Frequency Vectorization (number of times that term (word) t occurs in a comment d). The two methods used here were HashingTF function which was faster than CountVectorizer (from PySparkML) being the other one. CountVectorizer was used over HashingTF as it could convert the vectors back to the real words.
- Term frequency–Inverse document frequency (whether the term is common or rare across all comments) using PySpark.ML (IDF).
- Used pyspark.mllib.clustering LDA, LDAModel on our data trying different number of topics(k) and analysing the result. To run 20 topics with 500 iteractions it took 2.5 hours on a SFU cluster. With describeTopics() function topics were extracted and then converted to words (singular terms). As a result a csv file is created with all topics and their most frequent terms (words) (spark_topics_650000_20t.xlsx).
- Performed a visualization with MatPlotlib.pyplot and seaborn libraries by creating sns.barplots showing topics and 15 most frequent terms(words) in each of them. (spark_charts_650000_20t.png). First 4 topics shown in Figure 1.

Evaluation of Results (Based on top 20 topics)

It was discovered that the most common topic in comments about the **canadian worldwide government** and **political elections**. Next important Topic 2 is about **home and work of canadian immigrants**, Topic 3 is about **languages spoken in Canada**, Topic 4 is about crime and etc.

4.2 Topic Modeling using Gensim

Next, we found the 50 most frequent topics in comments with 30 words each. Bearing in mind a big limitation working with SPARK in terms of visualization we have decided to perform our task in just python. However we have agreed to use PySpark to at least clean from stopwords, normalize, and lemmatize our data to accelerate the whole execution.
Therefore to start with, we took an already cleaned file created in the first section (see first step in section 1.) cleaning.csv. The size of the cleaning file appeared to be half (175MB) of the original one (gnm_comments.csv). Now it's supposed to take less time to run any models. The following steps were performed:

- Created a chart and a xlsx file showing the most frequent words and their number in comments. It helped us to go throw them and eliminate more irrelevant words from our data (Figure 2)
- Created the term dictionary of our corpus using corpora.Dictionary(data) function from Gensim library, where every unique term is assigned an index.
- Converted list of documents (corpus) into Document Term Matrix using dictionary and doc2bow(data) function from Gensim library.
- Created object for LDA model using Gensim gensim.models.ldamodel LdaModel as Lda.
- Created the object for LDA model using Gensim & trained LDA model on the document term matrix. Parameters were 50 topics and 500 iteractions. As the result 50t.csv report was generated showing 50 topics with 30 most frequent words(terms)
- Visualized a result by using special pyLDAvis package for topic modeling in Gensim, It's an interactive presentation of topics and their more frequent 30 words(terms).

- Every circle represents a topic. The bigger the topic the more important it is. By navigating into a circle to right it's possible to see words which are in this specific topic. (Figure 2).

Evaluation and Result of Comments

Most common words in the whole comments are: Canada, government, Harper, work, vote, party, tax, pay, world, change, and etc. As the main result of our program it was discovered that the most common topic in comments expresses canadian and world culture and values. Next important Topic 2 is about legal issues and its prosecution and government. Followed by Topic 3 about jobs, salary and government. Topic 4 about governmental expenditures. Topic 5 tells us about the canadian government and its economy, Topic 6 tells us about the canadian worldwide government and political elections and etc.

4.3 Top Topics in Articles and Overlap with Comments

We worked on a gnm_articles.csv file which contains ~1800 authors and their 10,339 articles articles. We used the similar approach, methods, models, and libraries as mentioned above in a section 2 to generate 50 topics on articles. However in addition to that we had to find an overlap between authors who wrote those articles and topics associated with them. To do that with get_document_topics(data) function from Gensim we were able to extract and create a list of all topics associated to their articles. One by one. And merged a list of topics together main dataset with authors and their articles. As a result we were able to cluster (group) our 50 topics based on number of authors whose articles were related to those topics. The bigger a circle the more authors who wrote their articles on that topic. See Figure 4.
Also another picture was created which shows a number of authors who wrote on that article, most popular topics in the given data. Figure 5.

Evaluation and Result of Articles

The most common Topic 3 (appeared in ~1200 articles) talks about women, time, work, men etc. expresses working women and feminism. Next topic is "number 8" (appeared in ~980 articles) which talks about party, election, liberals, government and etc. expressing the election time and its important components. Further "Topic 12" (appeared in ~960 articles) talks about world politics. Followed by "Topic 0" (appeared in ~780 articles) which talks about war, terrorism and defense, "Topic 27" (appeared in ~610 articles) talking about policy makers and their major projects and "Topic 37" (appeared in ~600 articles) mentioning the tax and economy.

4.4 Lessons Learnt

We have run our programs on different number of topics. The total time of execution was highly correlated with number of topics. For example to run 15 topics it took about 2.5 hours, but to run 50 topics it took 24 hours. Also an interesting thing was discovered that running LDA model on comments with 20 topics in Spark and just python code took more or less the same time. Also when we run a code several times with 50 topics on articles we found out that the final result is always different meaning that the importance and relevance of topics are changing all the time we run a program.

5 Sentiment Analysis

We conduct sentiment analysis on the articles and comments using VADER (Valence Aware Dictionary and sEntiment Reasoner), which is based on a dictionary of a set of words with positive or negative sentiment scores. VADER calculates the sentiment score of a document by summing up the sentiment scores of each word in the dictionary.

We found that Between 2013 and 2016, the trend of sentiment scores is rather stable (between -0.25 and 0.25). There are some sharp peaks and troughs reach 0.5 and -0.5. We also found that the sentiments of the comments are correlated with the sentiments of the articles. The peaks and troughs in the sentiment scores correspond to some positive or negative articles. For example, On 2014-09-06, the sentiment score reached a peak of 0.034, and the articles released that day contained phrases such as 'I love it'. On 2014-06-01, the sentiments reached a low of -0.45, and the articles contained phrases such as 'misogyny', 'terrorism', and 'abortion'. We also found that the sentiments of the articles are average 0 and follows a normal distribution. This is expected as reporters are often expected to remain neutral when reporting. The sentiment of comments is also centered around 0, with approximately 10% of commenters consistently making negative comments and 15% consistently making positive ones.

Evaluation of Results

The stable pattern of the sentiments over time is reasonable as the media is expected to remain objective, and the number of articles that induce positive and negative sentiments should balance out. The peaks and troughs of sentiments is due to unusual news. For example, On 2014-09-06, the sentiment score reached a peak of 0.034, and the articles released that day contained phrases such as 'I love it'. On 2014-06-01, the sentiments reached a low of -0.45, and the articles contained phrases such as 'misogyny', 'terrorism', and 'abortion'. The normal distribution of article sentiments is expected as reporters are often expected to remain neutral when reporting.

6 Named Entity Recognition

- Files *'gnm_articles.csv'* and *'gnm_comments.csv'* were the 2 files used.
- '*cleaning_articles_ner.py'* and *'cleaning_comments.py*' were used to clean both the articles and the comments data respectively. These two use ***pyspark***.
- After cleaning, *'ner.py'* can be used to tag the words in the cleaned data. On articles it takes ~2 days and on comments its has not been checked.
- Alternatively there exists an NER project at 'https://github.com/glample/tagger' which can be used with the existing 'english' corpora to tag both articles' and comments' data. But before using it, make sure a C++ compiler exists in your machine without which the performance will be **severely degraded.**
- Using the above project we get tagged data as *'atags.txt'* (article tags) and *'ctags.txt'* (comments tag). Further, using *'ner_clean_tab.py'* the generated tagged data is cleaned and kept in the form of *csv* files *'atags.csv'*(article tags) and *'ctags.csv'*(comment tags).
- For doing the first visualization *(Figure 9, Figure 10, Figure 11)* to get the top named entities *'ner_final.py'* is used with *'atags.csv'*(article tags) and *'ctags.csv'*(comment tags) as input files.The top entities in articles(*Figure 9*), comments(*Figure 10*), as well as the overlapping entities(*Figure 11*), in these two are displayed. The python file generates an output file (*'ne_overlap.csv*') which contains the overlapping entities in both articles and comments.
- For the second visualization (*Figure 12)* use *'common_author.py'* with 'ne_overlap.csv' as input. This visualization uses plotly as well as PCA, KMeans from SciKit learn. It displays the similar authors who mostly use the common entities in their texts. Any number / any type of entity can be given for any desired visualizations.

All the python files as well as the intermediate data (*not the big ones*) are available on github.
https://github.com/pushsinha/TmNerSa

Summary : Text analysis with NLP concepts (NER, Sentiment analysis) and topic modeling with visualizations of the above approaches and creating a base with intermediate data and code for further much complex visualizations.

7 APPENDIX

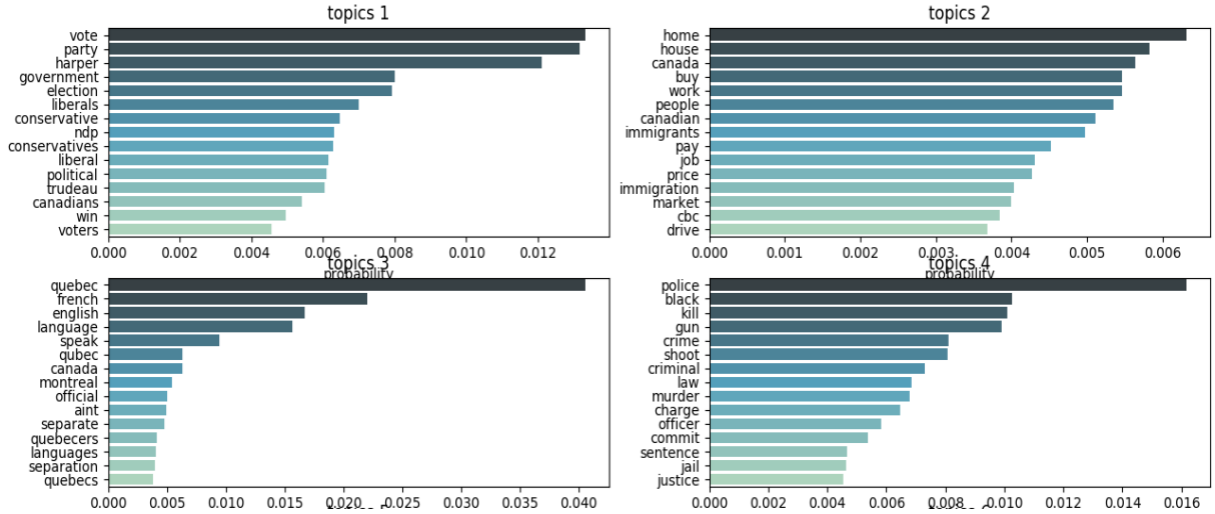

Figure 1

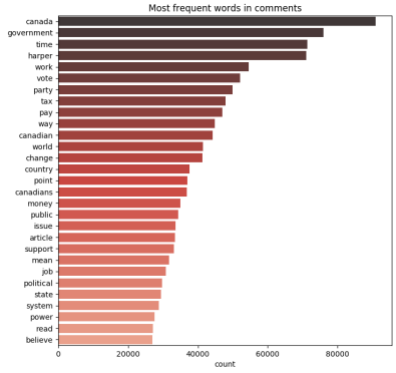

Figure 2

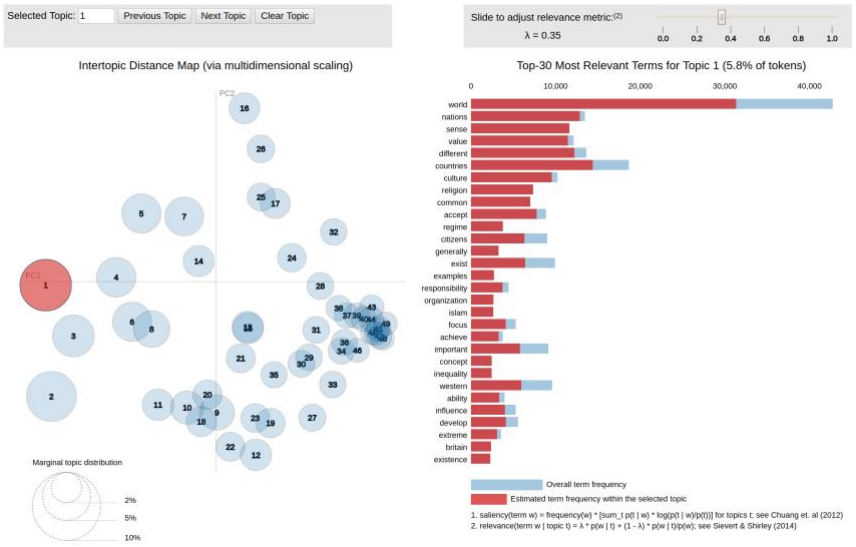

Figure 3

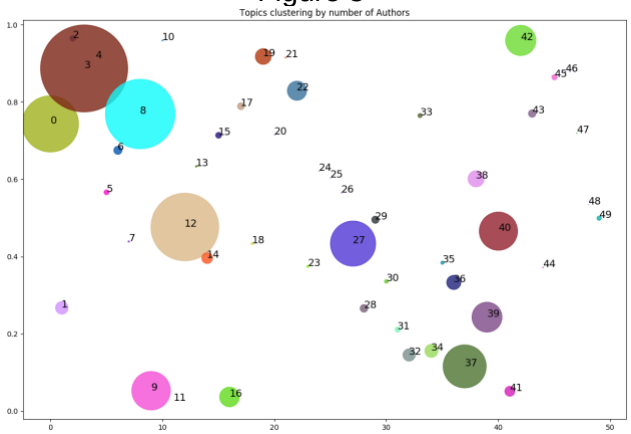

Figure 4

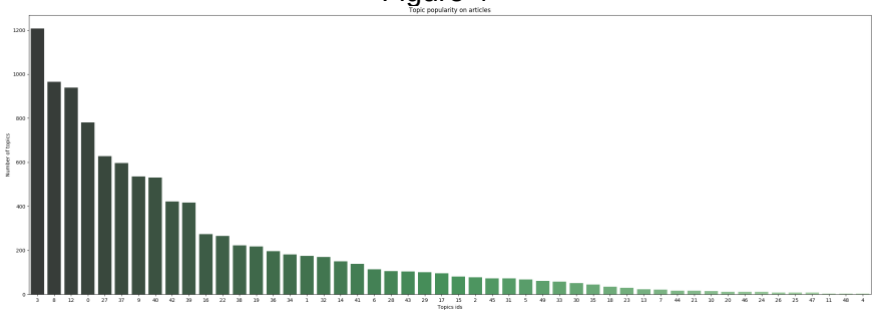

Figure 5

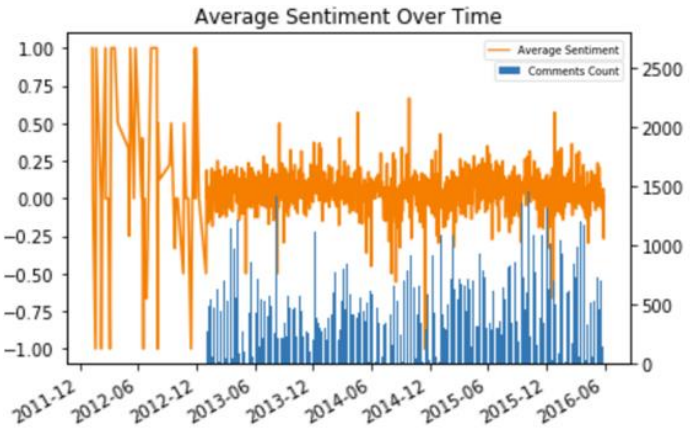

Figure 6

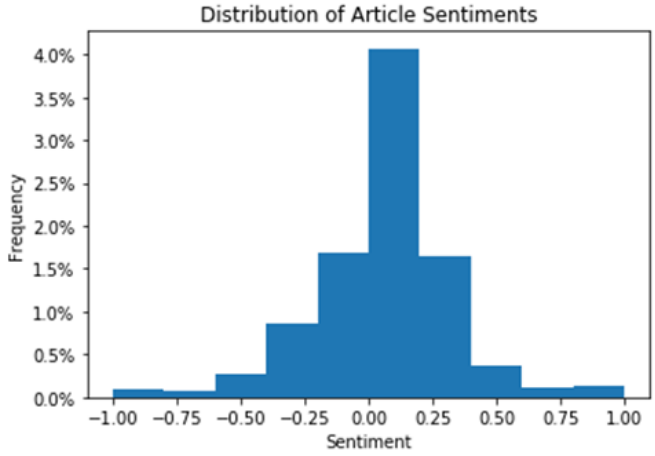

Figure 7

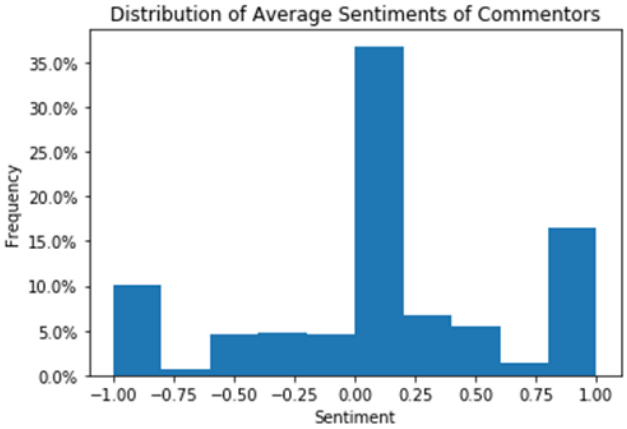

Figure 8

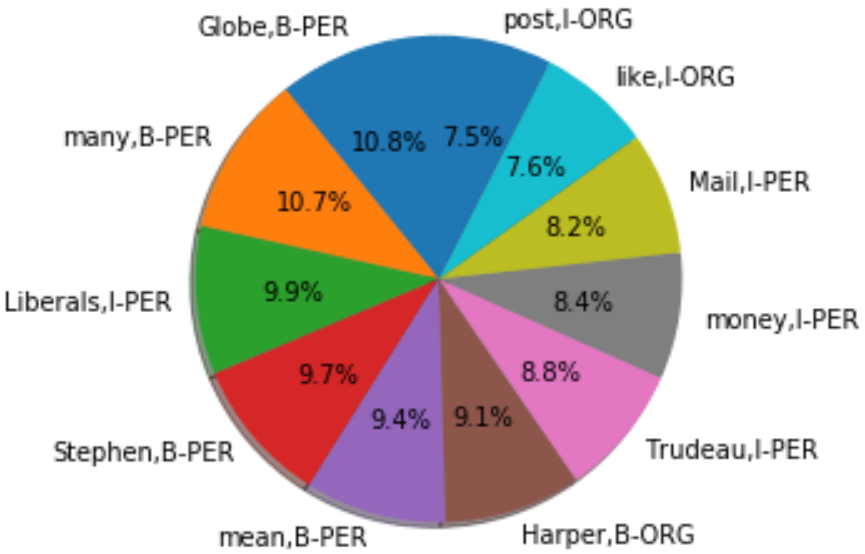

Figure 9

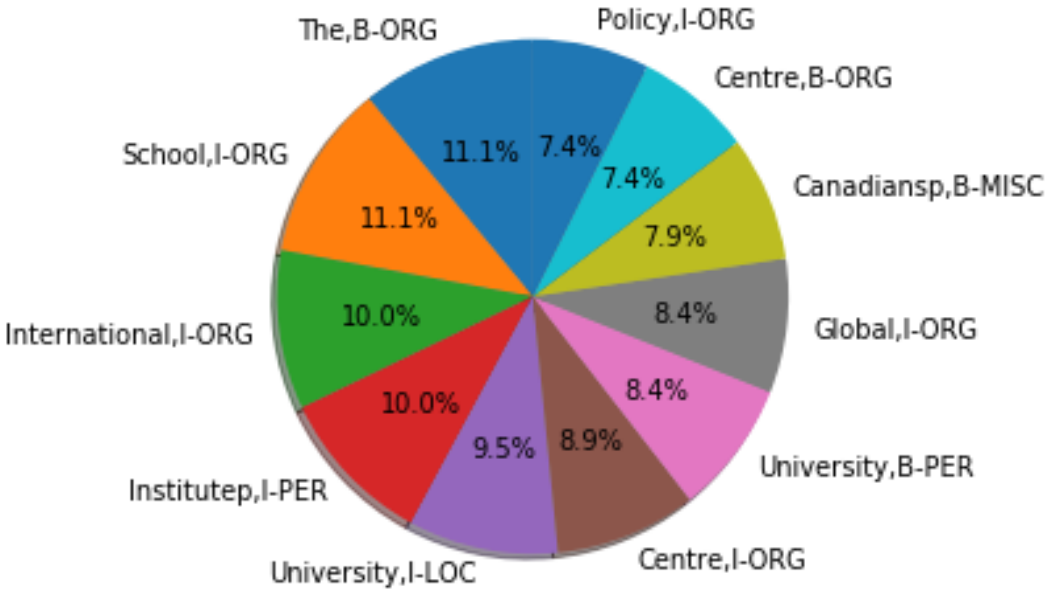

Figure 10

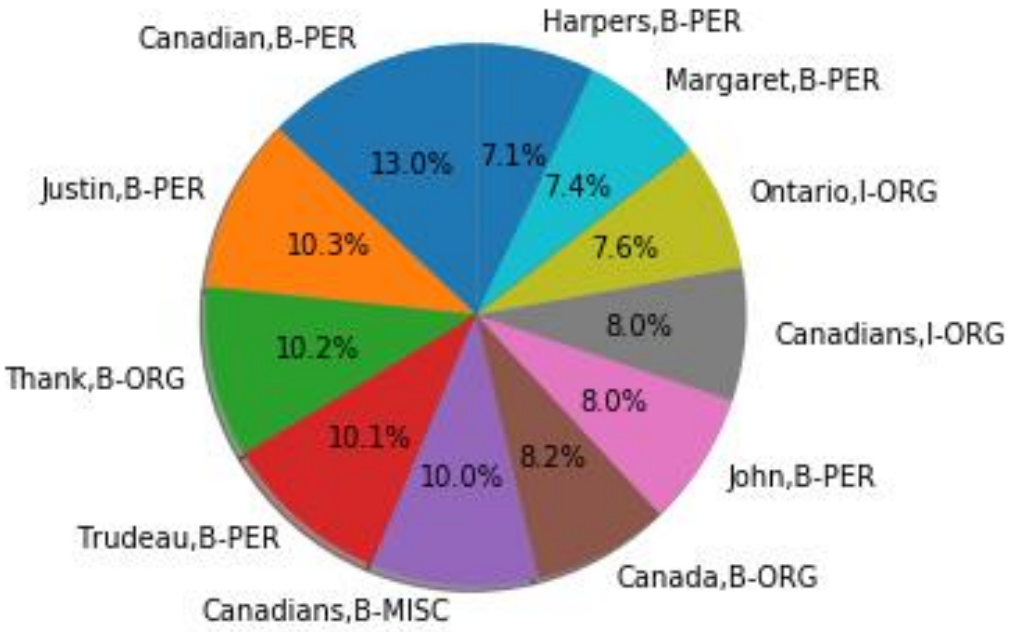

Figure 11

Clustered(similar) Authors for the top entities (Trudeau, Justin, Harper, Margaret, John, Ontario etc.) they use most in their text !!