Topic Modeling and Sentiment Analysis on Canadian News Articles and Comments

04.16.2018

Chithra Bhat, Ruoting Liang, Tianpei Shen
Professional Master's Program in Big Data
Simon Fraser University

Table of Content

Motivation and Background

The biggest motivation for doing this project is to create a succinct summary of different views or opinion between Canada citizens on articles, issues or policies for different time periods, so that to help Canadian Government or other Organizations to observe the 'Topic Trends' within Canada and make better decisions. This idea was inspired by *Google Trends* [1]. Unlike *Google Trends* compares the search volume over different periods of time, our team wants to compare the discussion volume of different topics for different time periods and finds out the most frequently discussed topic in Canada. In addition, our team wants to discover the attitude and sentiment along with a topic as well.

Problem Statement

This project is answering the following questions:

- Observe the change in hot topic:
  The data we got include over ten thousand articles from 2012 to 2016 publish in Canadian newspaper. We want to observe the 'Topic Trends' within Canada by counting the number of articles under a given topic for different years. The challenge is the existing popular topic modeling algorithm would not give us a high-level phase that represents the topic. Also, it is difficult to create a highly interactive visualization to demonstrate the 'Topic Trends'.
- Discover the surrounding topics on comments under a given article:
  For a given article, people not only talked the article, they also care about what the impact of the article may have. So, this motivates us to do topic modeling on comments under the articles. However, unlike article text, the comment text is usually short and is very challenging to generate a topic for short text. Moreover, the comment file is about 400MB, it takes long time to clean and process the data.
- Discover the constructiveness/toxicity/sentiment for the comments under a given topic:
  In order to clean up the comment environment, we want to remove nonconstructive and toxic comments. We also do sentiment analysis of comments to get an overview of the public opinion. The challenge here is to find labeled constructiveness, toxicity, and sentiment data that are very similar to our SOCC dataset. Another challenge is to evaluate the accuracy of the sentiment model with unlabeled comment data.

Data Science Pipeline

We divided our project into 6 phases: data collection, data cleaning & integration, exploratory data analysis, topic modeling, sentiment analysis, and visualization.

➢ Data Collection

This project uses *the SFU Opinion and Comments Corpus* (SOCC)[2] to do text mining because of it is a collection of opinion articles and the comments published in the Canadian newspaper The Globe and Main in the five-year period between January 2012 and December 2016.

All comments in the SOCC are unlabeled, as a result of that, our team needs to collect labeled training data to training sentiment analysis model. Thus, we collected positive/negative sentiment data[3] and toxic comment data[4] from *Kaggle* and obtained constructiveness data from *the Yahoo News Annotated Comments Corpus* (YNACC)[5].

➢ Data Cleaning & Integration

Cleaning is an important step before any text mining task. In order to make sure that the topic model is identifying interesting or important patterns instead of noise, we had to accomplish the following pre-processing or "cleaning" steps:

1. Escaping HTML characters: We used regular expressions to directly remove HTML entities (e.g. &lt, &gt, &amp) and then used *OpenRefine* and *BeautifulSoup* to change everything to lowercase, unescaped HTML characters, and removed extra white spaces between words.
2. Decoding data: We maintained the complete data in standard UTF-8 encoding format.
3. Removal of Expressions/Punctuations/ Stopwords: Textual data may contain human expressions like laughing, crying, and punctuations like "?!:;" etc. These expressions are usually non-relevant to topic modeling and hence needs to be removed. NLTK(Natural Language Toolkit) is used to clean the text.
4. Stemming/Lemmatization: Removing all the inflection from words. We used Porter Stemming Algorithm.

After cleaning the text and integrated few of the datasets to prepare for training the model.

➢ Exploratory Data Analysis

In order to analyze data sets and summarize their main characteristics, we did Exploratory data analysis (EDA) to better knowing the dataset. Below are two examples:

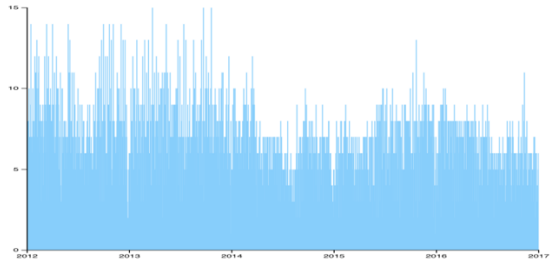

Figure 1: Number of opinion articles published from 2012 Jan-2016 Dec

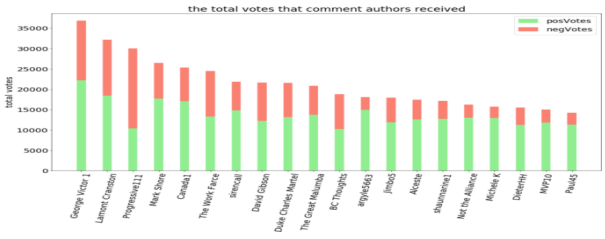

Figure 2: Top-20 comment authors who got the most number of votes

➢ Topic Modeling

Topic Modeling is an unsupervised learning approach to clustering documents, to discover topics based on their contents. We have tried two approaches in our project to do topic modeling on both articles and comments, Latent Dirichlet Allocation (LDA) and Non-Negative Matrix Factorization (NMF). We then used WikiAPI combined with human labeling to label the topics. For more details, refer to Methodology section.

➢ Sentiment Analysis

The sentiment analysis includes constructiveness, toxicity, and sentiment analysis. Firstly, we need to convert text comments into some numeric representations so that we could training on them. Two approaches have tried, bag of words model and word embedding model.

- For bag of words, we used the CountVectorizer from *scikit-learn*.
- For word embedding, we used Word2vec in *genism*. And then do a vector quantization by using k-means clustering to assign centroid to each word. We have set k=20 and then we create a 1-by-20 vector representation to each comment basic on how close the comment word vectors to each centroid.

Finally, we use the vector representations above to train a Random Forest Classifier.

➢ Visualization

Our team built a web application to create an interactive visualization of the learning results. We used Flask as the back-end servicer because it is easy to implement and it is Python based and so we could easily processing data in Pandas and PySpark. Then we used a Bootstrap template to build the web layout. All the visualizations are highly interactive and they mainly rely on D3.js and Dc.js. We also used ZingChart and plotly library to draw nice pie chart and word cloud.

Methodology

➢ Bag of word or Word Embedding

We found that word embedding works silty better than bag of words model to create numeric representations to do sentiment analysis. This is because word embedding methods (e.g. word2vec) are able to capture the relation between words and measure how similar/dissimilar words are. See Figure 1&2 for an example from Andrew Ng[6] to illustrate this concept.

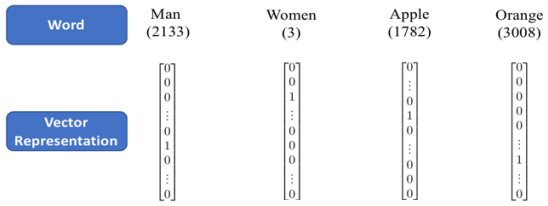

Figure 3

	Man (2133)	Women (3)	Apple (1782)	Orange (3008)
Gender	-1	1	0.00	0.01
Royal	0.01	0.02	-0.01	0.00
Age	0.03	0.02	0.03	-0.02
Food	0.04	0.01	0.95	0.97
...	...	...	...	...

Figure 4

As we see Figure 3, Bag of word uses one-hot-encoding to represent a word. For example, assume word 'Man' appears in the 2133th positive in a word dictionary, then uses vector with one at position 2133 and zero at other all position. Word Embedding (Figure 4) uses dense vector to represents a word. For example, the blue box shows a vector representation of 'Man'. By looking at the vectors, we may know that 'Man' and 'Women' are similar words, while 'Apple' and 'Orange' are similar words.

➢ Topic Modeling: LDA or NMF

The goal of topic modeling is to discover hidden semantic structures of text in a collection of documents. In this project, we will walk through two major topic modeling techniques, our findings as list below:

- Non-negative Matrix factorization works better for long text (e.g. article):
  Non-negative Matrix Factorization (NMF) and it relies on linear algebra that factors high-dimensional vectors into a low-dimensionality representation. Given the original matrix A, we can obtain two matrices W and H, such that A = WH, where A is Document-word matrix, W is Basis vectors and H is Coefficient matrix.
  We chose NMF to do article topic model because it is a very efficient algorithm and it takes the advantage that matrix values are non-negative and NMF provides better results. The words under one topic are more related and make more sense.

- Latent Dirichlet Allocation works better for short text (e.g. comments):
  Latent Dirichlet Allocation (LDA) is a probabilistic graphical modeling of a corpus and represents topic by word probabilities. Given a document d that contains n words, we calculate two probability values: P (word | topics) and P (topics | documents). The probabilities are calculated multiple times, until the convergence of the algorithm.

  Unlike article text consists of multiple long paragraphs, comments are usually short. The traditional topic modeling algorithm, such as LDA and NMF that we discussed above, made an assumption that one article belongs

to multiple topics. Therefore, we grouped all comments under one article and treated them as a single document. In this way, the document is long enough, and it may belong to multiple topics.

Compare to NMF model, LDA is more human-understandable because the algorithms found a group of words that are highly co-occurrence. Another reason we chose LDA uses the bag-of-words approach which counts the frequency of each word regardless of their positions. For a group of comments, there is a great possibility that multiple comments mention the same words.

- **WikiAPI: gives a high-level phase to represent a topic.**

  A basic python library enabling access to Wikipedia.org's search results and article content. This library searches Wikipedia articles to get article summaries. We used this library in our project to get possible article topic names for a list of keywords generated under a topic by the topic model algorithm and before human labeling.

- **D3.js: builds interactive and beautiful visualizations.**

  Our team spent about two weeks on building an interactive visualization to demonstrate our learning results. We have chosen D3.js because it is as powerful as you want to make it. And it is written in JavaScript and uses a functional style which means you can reuse code and add specific functions to your heart's content.

Evaluation

Since the *SOCC* data are all unlabeled, it is not easy to evaluate the accuracy for topic modeling and sentiment analysis. To evaluate our topic models, we first evaluated how related the keywords of topic model generates used Word2vec. Second, we referred back to the actually articles to see if the keywords can represent the article's topic. For sentiment analysis evaluation, we tried two approaches to create a test dataset for sentiment. One uses expression to label the sentiment along with a text. For example, labeled positive if searched simile faces like ":)" in comment text, and labeled negative if a crying or upset faces like ": (". The other approach is to randomly select 100 samples from the dataset and manfully labeled these sub-samples of data. The accuracy we got for positive vs negative sentiment bellows 50%, which is very low. Topic models works pretty well, most of the generated article topics can represent the article contents.

Data Product

We have built a web application to visualize all learning results. The image in the cover page is the front page of the web application. In the discover hot topics page, we listed top 20 discussed topics of articles and rank them by frequency (Figure 5). In the surrounding topic page, we evaluated 10 topics of comments and their proportions of a given article used a donut chart (Figure 6). When you click on an article node, it will show the actually article content with a tree demonstrate the sentiment associates with the clicked article (Figure 7).

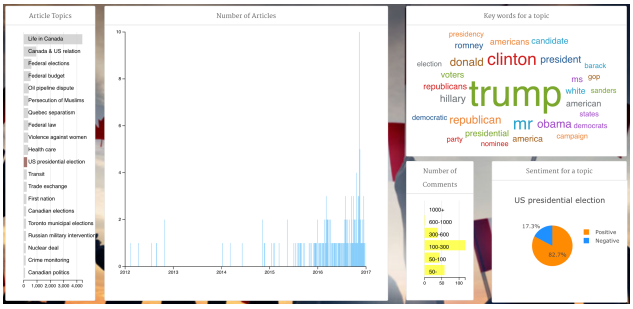

Figure 5: The number of articles under topic US presidential election got increased dramatically in late 2016 because of Trump was elected as US president. Wordcloud shows the keywords and pie chart shows the sentiment of a topic.

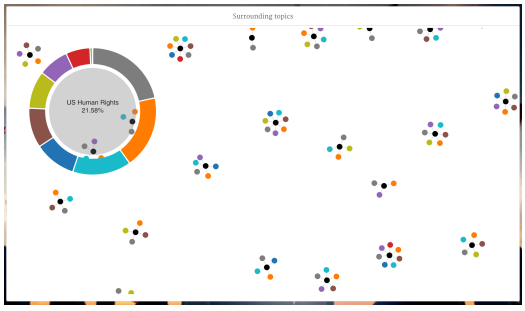

Figure 6: the black node represents article, while the color node represents comment topics. There are 21.58% of comment under US presidential election talks about US human rights.

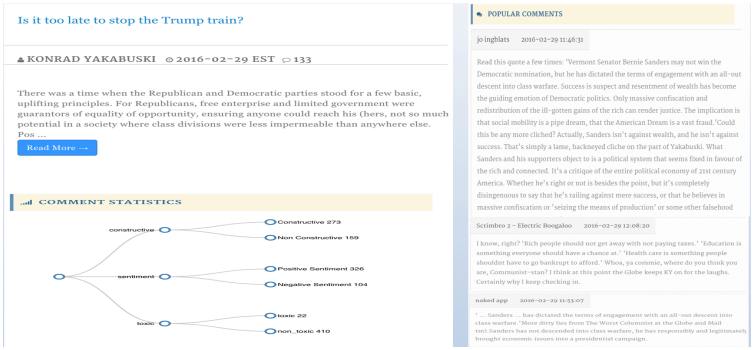

Figure 5: The tree shows there are 273 constructive vs 159 non-constructive comments, 326 positive sentiment vs 104 negative sentiment, and 22 toxic comment vs 410 non-toxic comment under article 'Is it too late to stop the Trump train'

Lessons Learnt

Key takeaways from this project includes:

- Stemming is not always good: Stemming results in a list of word-vector, which is not human-readable. When we saw the generated topics and the key terms under the topics, we noticed that stemming sometimes give us strange word such as 'parti', 'conser'. So, in our project instead of using stemming, we used lemmatization i.e., WordNetLemmatizer to improve the readability of topic terms.
- How to deal with large data set: The article data set is 44MB and the comment data set is 350MB. So, we used Spark to process the data and created UDF for code implementation. Also, for visualization, in order to improve the performance, we retrieved required data based on ID instead of reading the whole document.
- How to do topic modeling on short-text: LDA models a document as a mixture of topics and works better in short text topic modeling.
- Carefully sentiment analysis needed: For positive/negative sentiment analysis, most of the comments are classified as 'positive' sentiment. In fact, this may not the case; for example, reviewer A comments "I hate Trump.", follows with reviewer B replies A says, "I agree with you". In this case, we should have two 'negative' sentiment. However, our sentiment classifier will output one 'negative' and one 'positive'.

Summary

In summary, our project focused on topic modeling and sentiment analysis on Canadian news articles and comments. We used Latent Dirichlet Allocation and Non-negative Matrix Factorization to build topic models on new articles to observe the 'Topic Trends' in Canada over last 5 years and discover the surrounding topics of comments under a given article. Additionally, we cleaned up the comment environment by removing nonconstructive and toxic comments. We also did positive/negative sentiment analysis of comments to get an overview of the public opinions. Finally, we built the web application with interactive graphs to visualization all the learning results.

Reference

[1] Google Trends. https://trends.google.com/trends/

[2] The SFU Opinion and Comments Corpus. https://github.com/sfu-discourse-lab/SOCC

[3] Bag of Words Meets Bags of Popcorn. https://www.kaggle.com/c/word2vec-nlp-tutorial#part-2-word-vectors

[4] Toxic Comment Classification Challenge. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[5] C., J., E., B., & A. (2017). Finding Good Conversations Online: The Yahoo News Annotated Comments Corpus. Valencia, in: Association for Computational Linguistics. Page 13-23

[6] Word representation presented by Andrw Ng. https://www.coursera.org/learn/nlp-sequence-models/lecture/6Oq70/word-representation

[7] Vivek Kumar Rangarajan Sridhar. Unsupervised topic modeling for short texts using distributed representations of words.