Capstone Project - Fall 2019
Data Science Institute, Columbia University

Progress Report 2

Editorial Classification - Bloomberg

November 25, 2019

Team Members
Aastha Joshi
Ameya Karnad
Nirali Shah
Sarang Gupta
Ujjwal Peshin

Industry Mentors
Dr. Daniel Preotiuc-Pietro, Senior Research Scientist, Bloomberg
Kai-Zhan Lee, Data Scientist, Bloomberg

Faculty Supervisor
Dr. Smaranda Muresan, Research Scientist, Data Science Institute, Columbia University

Contents

1 Feedback from Report 1

In this section, we look at how the feedback recieved from mentors was incorporated in the project.

1.1 Article text exploration (Named Entity and Parts of Speech)

Along with the visualization depicting the distribution of Regular and Opinion articles (Refer Report 1 and Appendix), we look at statistics around important NER/POS. We see that the statistics have distinct means in the context of regular and opinion articles, but we cannot claim that they have a different distribution due to high stddev values. Nonetheless, we can take them as features. Refer to the Appendix for the statistics pertaining to all Parts of Speech and Named entities.

POS/NER	Regular Articles		Opinion Articles	
	Mean	Stddev	Mean	Stddev
Date	0.0099	0.0065	0.015	0.0097
Time	0.0026	0.0029	0.0065	0.0089
Quantity	0.0028	0.0025	0.0052	0.0063
Adjective	0.0693	0.0170	0.0553	0.0187
Adverb	0.0518	0.0161	0.0372	0.0174
Number	0.0162	0.0123	0.0317	0.0293

1.2 Correlation Analysis among Article Features

We conducted a covariance analysis to examine how different features of the articles relate to each other and the target variable. Some significant correlations are visible in the matrix. For instance, the POS tag for proper nouns is significantly correlated with PERSON and ORG NER tag. This makes sense as person and org names are more likely to be proper nouns than common nouns.

There are some significant correlations between article features and the target variable (Figure 1). Opinion articles, in general, have a higher number of questions (normalized for article-length) as compared to regular articles. Similarly, regular articles are longer in length than opinion articles. For the analysis of other features, please refer to the Appendix.

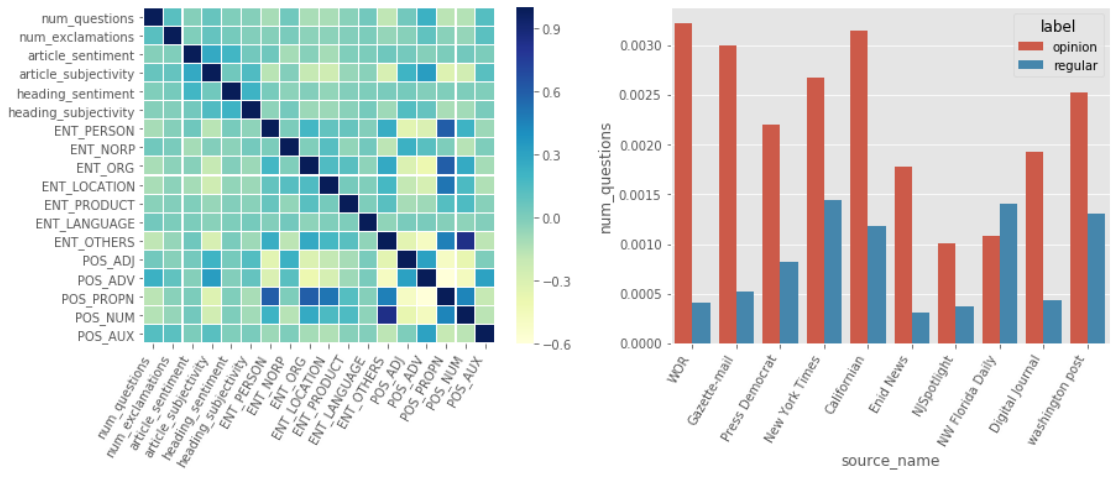


Figure 1: (a) Covariance between features (b) Normalised number of questions by source

2 Data Preparation

In this section, we look at the data preparation process that was employed. This was a subsequent step to the data collection step. This was a step that would prepare the data for input into the ensemble and deep learning models. The data preparation module involved a two-pronged approach:

2.1 Train-test split and handling class imbalance

2.1.1 Train-Test split

The train and testing splits were carried out in a temporal fashion to ensure that the testing would be on a future data set and the model would be robust to future datasets. Hence, the training data would encompass 90% of the original data set and the testing would be the remaining 10%. A date was selected such that 90% of the data set before that date would become our training dataset and after that would become our testing dataset.

2.1.2 Undersampling

Then, undersampling was carried out on the training data set to ensure that the balance of both the classes was equal. The undersampling would ensure that for every article of our 1 class on the same date, there would be an equivalent 0 class in the training set of the same day. A look back - look forward paradigm was implemented to pick an equivalent 0 class if it wasn't available on the same day.

2.2 Text Processing and Feature Creation

2.2.1 Text Preprocessing

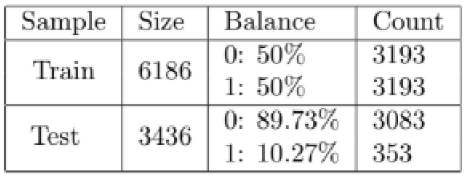
Additional preprocessed text was generated in order to aid the generalizability of the models. The various steps include converting numbers to strings, lowercase conversion, stopword removal, short word removal, stemming / lemmatization, expansion of contractions and punctuations removal.

2.2.2 Feature creation

Features were extracted from both raw articles and processed text and headlines to be fed into our machine learning models. General-purpose text features included average word length and sentiment (using VADER, AFINN, and TextBlob) of the article (both retrieved from the processed text). Other features retrieved from the raw articles include heading text, heading length, article length, number of punctuations (question marks and exclamations), subjectivity (using TextBlob) and normalized counts of Named Entity Recognition (NER) and Parts-of-Speech (POS) tags (using SpaCy). Data-specific text features included vectorized text (character and word-based) using Count and Tf-Idf vectorizers. N-gram ranges of (1, 2) and (1, 3) were used for both the vectorizers.

Refer to the Appendix for further details about each of the features.

The final test train split is as shown in the table below



Sample	Size	Balance	Count
Train	6186	0: 50%	3193
		1: 50%	3193
Test	3436	0: 89.73%	3083
		1: 10.27%	353

3 Models

In this section, we discuss the different models that were picked and trained on the data.

3.1 Baseline model

For the baseline, we used a simple logistic regression model with a CountVectorizer. CountVectorizer provides a simple way to not only tokenize a collection of text documents and build a vocabulary of known words but also to encode new documents using that vocabulary.

To display and compare the results of the baseline with all the other models, we run all the models on first 64, 128, 256 and 512 words.

3.2 Linear and Ensemble models

The following general classification models were used

- Logistic Regression (with TF-IDF)
- Decision Tree
- Random Forest
- Gradient Boosting

The models were run using python's *scikit-learn* package. The description of parameters is given in the Appendix.

To get the best out of the algorithms, we tuned parameters.

- C, penalty (Logistic regression)
- n_estimators, max_depth (Random Forest)
- max_depth (Decision Tree)
- n_estimators, max_depth, learning_rate (Gradient Boosting)

3.2.1 LightGBM

Along with a simple Gradient Boosting Classifier, we also decided to try LightGBM (Ke et al., 2017) as it is frequently used in Kaggle competitions, and it trains fast and gives better performance than other ensemble models. LightGBM is a really fast and distributed ensemble model that is primarily based on decision trees. It splits the trees leaf-wise instead of tree-depth wise or level-wise (Figure 2). Hence, it leads to faster training time and more reduction in loss (Daoud, 2019).

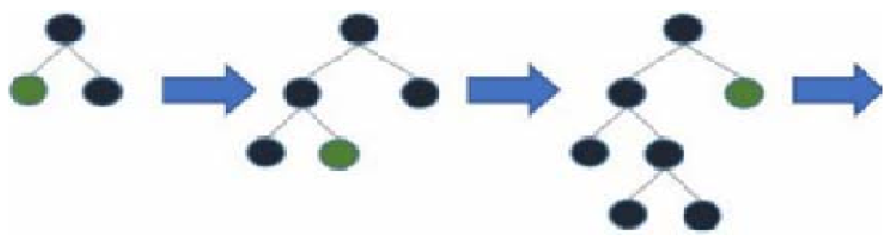


Figure 2: LightGBM Leaf-wise tree growth

LightGBM uses Gradient-based One-Side Sampling to find which value to split on, which is faster than a Histogram-based approach for finding splits. So, it keeps all instances which lead to higher gradients, and then randomly samples instances which have a lower gradient. The assumption here is that those samples with lower gradient have a smaller training error, and hence the model is already tuned to those samples.

The parameter tuned for LightGBM were max_depth, num_leaves, n_estimators, and learning_rate. The description of parameters is given in the Appendix.

3.3 Recurrent Neural Networks and LSTM

Recurrent Neural Networks (RNN) are neural networks where connections between nodes form a directed graph along the time dimension. This allows information to persist over time. Long Short Term Memory Networks (LSTM) (Hochreiter et al., 1997; Olah, 2015; Karpathy, 2015) are a special class of RNN capable of learning temporal dependencies. They are designed to ensure that the long term dependency problem of RNN is avoided using forget gate.

The input text is pre-processed & converted to the tokenised format. After pre-padding it for various article lengths like 64, 128, 256 & 512 words it is fed to an Embedding layer. The weights used for the embedding layer are Glove (Pennington et al., 2014) 300-dimensional word vectors. They have been pre-trained on Wikipedia 2014 text corpus.

3.3.1 Bidirectional LSTM

Bidirectional LSTM (BiLSTM) is type of LSTM model that connects 2 hidden layers of opposite directions to the same output. The neurons of a regular RNN are split in 2-time directions, one for forwarding states & other for backward states (Schuster et al., 1997). Outputs from both states are not connected. Instead, they are concatenated at each time step. This helps take into consideration the future input information for prediction. The input sequence is fed in normal time order for one network, & in reverse time order for another.

3.3.2 Model architecture

The general model architecture used is LSTM/BiLSTM layer with 64 hidden units. The LSTM/BiLSTM layer has been regularised by using dropout (p=0.2) & recurrent dropout (p=0.1) on hidden layers. The next layer used is a global Max pooling layer. It is followed by a dense layer of 64 units with 'relu' activation. Next, a dropout layer with dropout percentage p=0.5 is used. Dropout helps improve model generalizability & prevent model overfitting. The last layer is a Dense layer with softmax activation to give a probabilistic output score for article classification.

3.3.3 Bidirectional LSTM with Attention

For our dataset, we observed that training BILSTM led to a higher macro F1 score than the LSTM model. To further improve BiLSTM, word-level attention was incorporated. It is observed that different words & sentences in a document are differentially informative. Attention uses context to discover when a sequence of tokens is relevant rather than simply filtering for sequences of tokens taken out of context (Yang et al., 2016; Vaswani et al., 2017).

3.3.4 Multi-input Bidirectional LSTM

A Multi-input BiLSTM model was developed. The 2 inputs to the model are article heading & text. Both input sources were trained separately on the same model architecture (Refer to Section

3.3.2). The results were concatenated & the final vector passed through a dense layer with softmax activation function to predict the final class probabilities.

3.4 BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformer (BERT) is a deep language understanding model released by Google in late 2018. BERT uses a multi-layer bidirectional transformer encoder to learn contextual representation between words within a text. Unlike Bidirectional LSTM which is trained both from left-to-right and right-to-left to predict the next and the previous words independently, BERT looks at both the ways at the same time - the model is made to learn from the entire sentence.

3.4.1 Model Architecture

Instead of using Recurrent Neural Network Architecture mechanism to model sequences, BERT incorporated a Transformer in its architecture, which uses the attention mechanism over the input sequence (Singh, 2019). Attention allows the model to focus on relevant parts of the input sequence as needed (Alammar, 2018).

The input to the model is a combination of position embeddings (to represent the position of words in a sentence), segment embeddings (to represent the position of sentences in text comprising of multiple sentences) and token embeddings. BERT uses a WordPiece tokenizer which reduces the size of the vocabulary significantly and helps cover a wide spectrum of out-of-vocabulary words.

The original BERT paper presents two model sizes:

- BERT Base: A lighter version of the model with 110M parameters and 12 layers
- BERT Large: Full-size model with 340M parameters and 24 layers

3.4.2 Model Training

BERT is trained using 1) Masked language modeling where 15% of the tokens are masked at random and the model tries to predict the masked words based on the unmasked words. 2) Next Sentence Prediction where the model predicts where a second sentence is actually the next sentence. BERT is pre-trained on a corpus of BookCorpus (800M words) and English Wikipedia (2,500M words) (Devlin et al., 2018).

3.4.3 Model Fine-tuning

Hugging Face's implementation of the BERT model (Wolf et al., 2019) in PyTorch was imported and fine-tuned on our training data. An untrained layer of 2 neurons for the purpose of binary classification was added at the end of the network. The fine-tuning process (v/s training the model from scratch) allows easy training with less data by leveraging the already learned language representations from the large English corpus mentioned above.

Uncased and Cased versions were run for both BERT Base and BERT Large. The text is lowercased before the WordPiece tokenization in the uncased version. Both models allow for a maximum sequence length of 512 tokens. The model was run on Google Cloud's AI platform and the BERT base model for a run with sequence lengths of 64, 128, 256 and 512. Due to memory limitations of the available GPUs, BERT large models with sequence lengths greater than 128 could not be run.

The results of the model fine-tuning are reported in the Results section.

3.5 XLNet

XLNet (Yang et al.,2019) is a model that was released by CMU and Google Brain in conjunction in June 2019. It claimed to outperform BERT, which was the previous state-of-the-art in 20 language modeling tasks.

XLNet is a generalized Autoregressive model, that uses either forward or backward context in order to train the model. XLNet allowed Autoregressive models to learn from forward as well as backward context at the same time. XLNet defined a new training objective called Permutation Language Modelling, which was based on permutations.

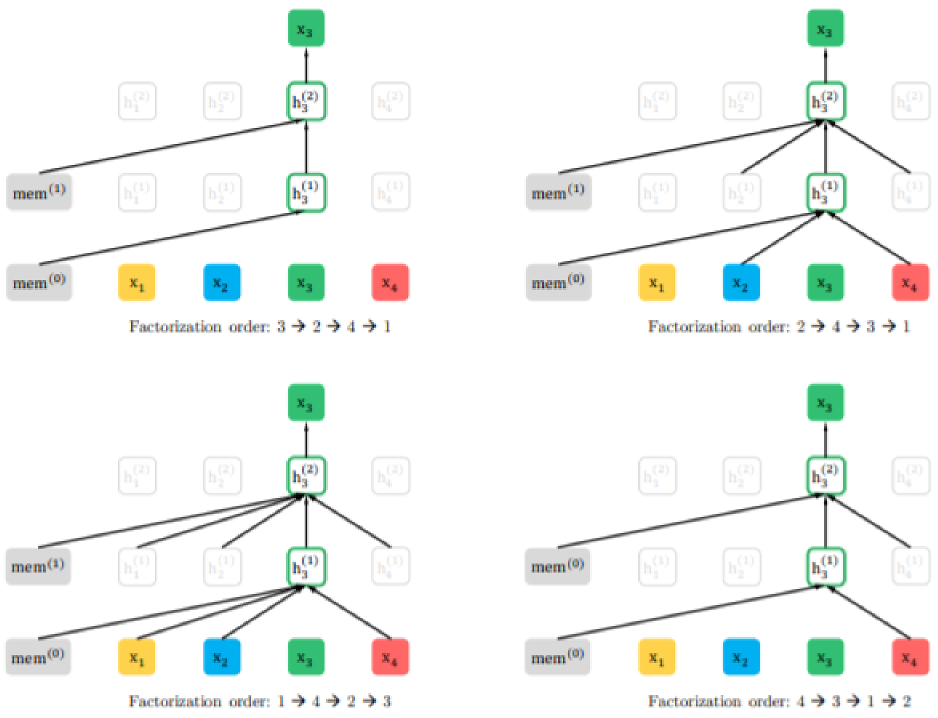

Factorization order: 3 → 2 → 4 → 1
Factorization order: 2 → 4 → 3 → 1
Factorization order: 1 → 4 → 2 → 3
Factorization order: 4 → 3 → 1 → 2
mem(0)
mem(1)
x1
x2
x3
x4

Figure 3: Different Permutations generated by the model to predict 3

Let us say that we have the following token sequence, 1 2 3 4, and we want to predict 3. XLNet would generate a random sequence of this combination, and would then ask the model to predict 3, given everything before it in that permutation. Hence, as we can see above, the model has generated the following combinations, $3 > 2 > 4 > 1$, $2 > 4 > 3 > 1$, $1 > 4 > 2 > 3$, $4 > 3 > 1 > 2$. Hence, the first permutation is asking, given that there is no information about the content of the sentence, what would be the third token. The second permutation is asking the model, given that we know that the 2nd token is 2, and the 4th token is 4, what would be the 3rd token, and so on. Hence, the model has information about context from both sides.

The implementation of all these models can be viewed on our GitHub page:

https://github.com/ujjwal95/bloomberg_editorial_classifier

4 Results

In this section, we talk about the various metrics we used to evaluate our models. We also discuss important findings from the various algorithm runs.

4.1 Evaluation Metrics

We evaluated our model using numerous evaluation metrics. A complete list of metrics is provided in the Appendix. In the report, we will particularly be focusing on 3 of them.

4.1.1 Matthews Correlation Coefficient (MCC)

Introduced by Matthews (1975), this takes all the 4 parameters of the 2x2 confusion matrix into account (unlike precision and recall). It is a balanced measure and can be used in case of imbalanced classes. The MCC is a correlation coefficient between the observed and the predicted class values.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}$$

4.1.2 Macro Avg F1 Score

Macro Avg F1 score is a metric that can be used when the classes are equally important. It is given by

$$F1(Macro) = 2 * \frac{Precision(Macro) * Recall(Macro)}{Precision(Macro) + Recall(Macro)}$$

$$Precision(Macro) = \frac{(\frac{TP}{TP+FP}) + (\frac{TN}{TN+FN})}{2}$$

$$Recall(Macro) = \frac{(\frac{TP}{TP+FN}) + (\frac{TN}{TN+FP})}{2}$$

So, this gives a balanced measure which takes into account both the classes. The Macro Average F1 score is also an industry-standard when it comes to classification. We also explored the possibility of taking the Weighted Average F1 score into account (The weights are given based on an imbalance in the class). But ultimately, we went with the Macro Average F1 score as it treats both classes equally.

4.2 Main Results

Now, let's look at how our models work with regards to the metrics mentioned above.

For the graphs, we selected the best out of each type of model. The models and their configurations are

- Baseline : Logistic regression with Countvectorizer
- Logistic regression with tf-idf
- Decision Trees
- Random Forests
- Gradient Boosting
- LightGBM

- Combined BiLSTM (Text + Title)
- *bert-base-cased* with a batch size of 24
- *xlnet-base-cased* with a batch size of 8

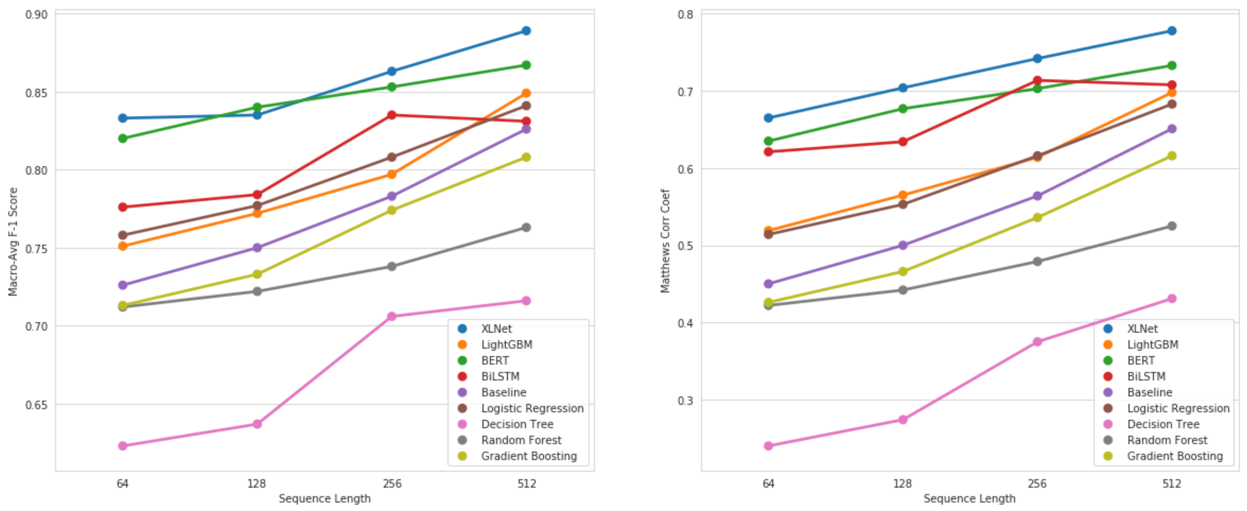


Figure 4: (a) Performance of different models with their Macro Avg F1 Scores. (b) Performance of different models with their Matthews Corelation Coefficients.

We found out that XLNet is the best performing model on both the Macro Average F-1 Score metric (Figure 4(a)) and Matthew's Correlation Coefficient metric (Figure 4(b)) followed by BERT. This is most likely due to the fact that XLNet and BERT are pre-trained on a large corpus of English text, which enables it to understand the semantics of the language. Moreover, they have complex architectures that enable them to learn the underlying structure of the data as compared to the linear models. All the other models, on the other hand, have been trained from scratch and do not provide the benefits associated with transfer learning.

Furthermore, Logistic Regression models fitted with count vectorizer (baseline) and Tf-IDF vectorizer performed better than the three tree-based models.

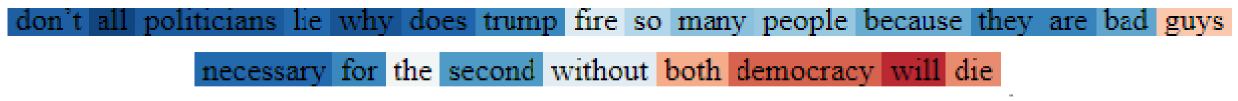
4.3 Feature Importance - Linear Models

The feature importances generated by the linear model helped us determine the most important features for this classification task. While the most important features varied across different models, the most persistent ones included 'ENT_OTHERS' (Other entities), POS_PROPN (Proper nouns), POS_ADV (Adverbs), NEU_VADER (Sentiment). Specific words like 'said' and other characters identified by the tf-idf vectorizer also had high feature importances.

4.4 BiLSTM Attention

In this subsection, we continue our discussion on the results of the attention for BiLSTM Adding attention to our BiLSTM model did show a marginal macro F1 score improvement for word length 128 and 256. However, a slight decrease in the Macro Average F1 score was observed for the word length of 64 and 512. The visualization for the attention for an opinion article with article length as 256 and 64 hidden units of the BiLSTM model is as in figure 5.

It can be seen from Figure 5 that the model pays more attention to words like why, don't, lie, all,


don't all politicians lie why does trump fire so many people because they are bad guys

necessary for the second without both democracy will die

Figure 5: BiLSTM attention based word score

etc. which convey strong emotions of the author. Words like both, guys, democracy, die, etc. are given a lower score as they are more generic and don't give any strong opinions of the author.

The entirety of results can be accessed from the below google sheets link

https://drive.google.com/file/d/1TIc3fmMXsHR-jOEJ-4UvcmN7hc8x88RB/view?usp=sharing

5 Other Progress and Next Steps

In this section, we look at the next steps and future scope of the projects.

5.1 Piecewise sequence Analysis

We want to analyze if certain parts of the news article give more information about whether the article is a regular article or a non-regular (opinion) article. One important analysis that we plan to do involves running all our algorithms on a combined text containing the Article title, the first 150-200 words of the article and the final 150-200 words of the article. The hypothesis is that the introduction and the conclusion to the article may give information about the article type.

Current Status : We have already started working on this part

5.2 Fresh Test Set

To verify whether our models generalize to new data, we decided to evaluate them on a new set of news articles. To do this, we collect data from **Metro Winnipeg**. Metro Winnipeg is a news website, based out of Winnipeg, Manitoba, Canada. This news site was specifically chosen to ensure that the model is evaluated on a new, hidden test set. Evaluating on Canadian news can determine if any of our models have trained on features which are news-resource or location (United States) specific. For this task, we plan to train the linear and Deep learning models on the original training data and then evaluated on the test data created.

Current Status : Data collection for Metro Winnipeg is completed. Testing in Process

5.3 Topic Modeling

We plan to identify the latent topics in the articles and explore month-wise patterns using Topic Modeling. For this, we are currently exploring Dynamic Topic Models (DTM) proposed by Blei et al.(2006). Dynamic Topic Models are built on top of static Latent Dirichlet allocation(LDA) (Blei et al., 2003) by extending the topics along the time axis. This enables us to explore the evolution of trends through time. We plan to run the Topic-over-time DTM algorithm month-wise on articles published in 2019. Subsequently, we plan to look at trends of topics along the Regular - Non-Regular lines and look for distinctions between them.

Current Status : DTM exploration completed, Implementation in progress.

6 References

Alammar, J. (2018, May 9). Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention). Retrieved November 23, 2019, from https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.

Blei, D. M., & Lafferty, J. D. (2006, June). Dynamic topic models. In Proceedings of the 23rd international conference on Machine learning (pp. 113-120). ACM.

Daoud, E. (2019). 'Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset'. World Academy of Science, Engineering and Technology, Open Science Index 145, International Journal of Computer and Information Engineering, 13(1), 6 - 10.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Karpathy, A. (2015, May 21). The Unreasonable Effectiveness of Recurrent Neural Networks. Retrieved November 23, 2019, from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems (pp. 3146-3154).

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA)-Protein Structure, 405(2), 442-451.

Olah, C. (2015, August 27). Understanding LSTM Networks. Retrieved November 23, 2019, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pennington, J., Socher, R., & Manning, C. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11), 2673-2681.

Singh, A. (2019, June 26). Building State-of-the-Art Language Models with BERT. Retrieved November 23, 2019, from Building State-of-the-art Language Models with Bert Ankit Singh - https://medium.com/saarthi-ai/bert-how-to-build-state-of-the-art-language-models-59dddfa9ac5d.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. and Brew, J., (2019). Huggingface's transformers: State-of-the-art natural language processing. ArXiv, abs/1910.03771

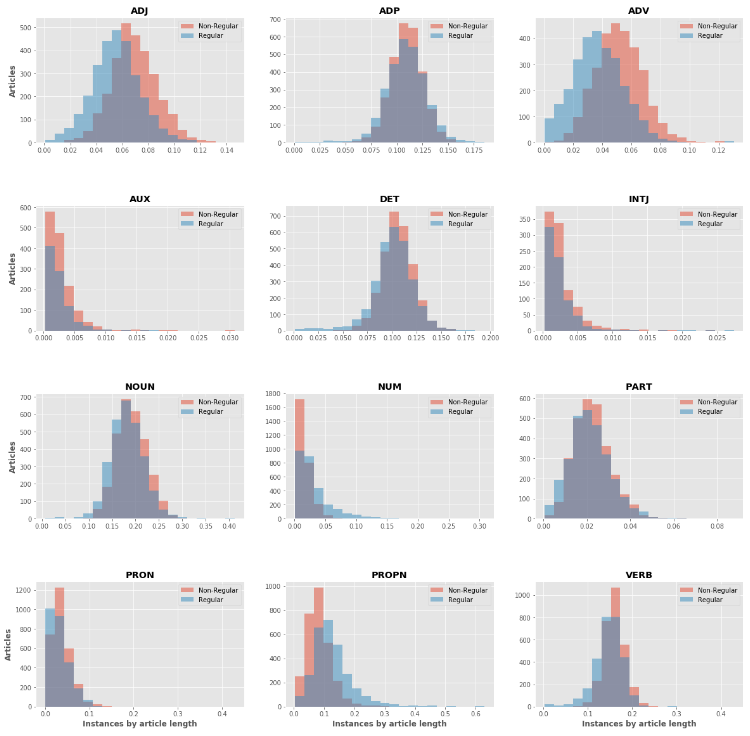
Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv preprint arXiv:1906.08237.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016, June). Hierarchical attention networks for document classification. In Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies (pp. 1480-1489).

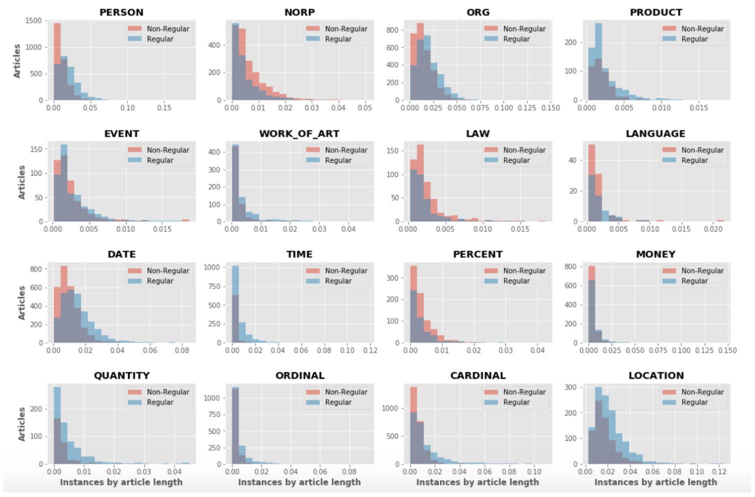
7 Appendix

7.1 Named Entities and Parts of Speech Distribution Graph

Please note that these graphs are reproduced from Report 1. Please refer the report for detailed explaination



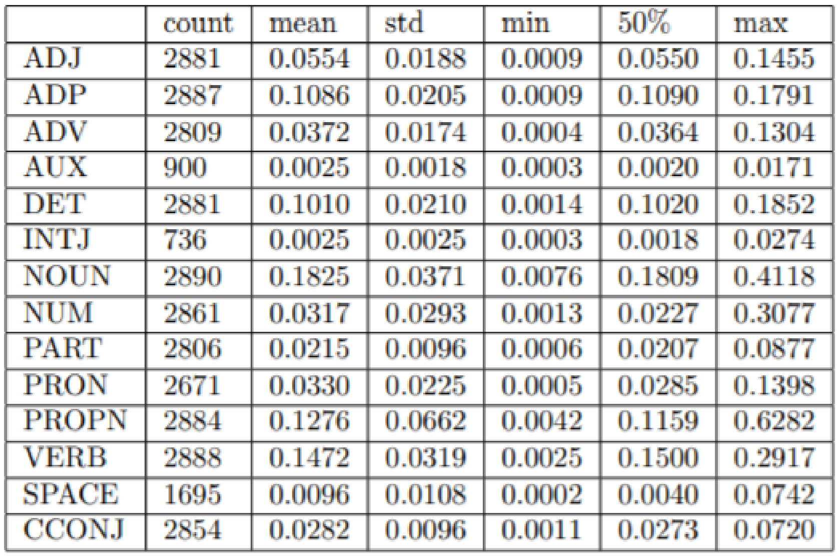
The above figure is the distribution of Part of Speech. Please refer Figure 4 of Report 1 for more details



The above figure is the distribution of Named Entities. Please refer Figure 5 of Report 1 for more details

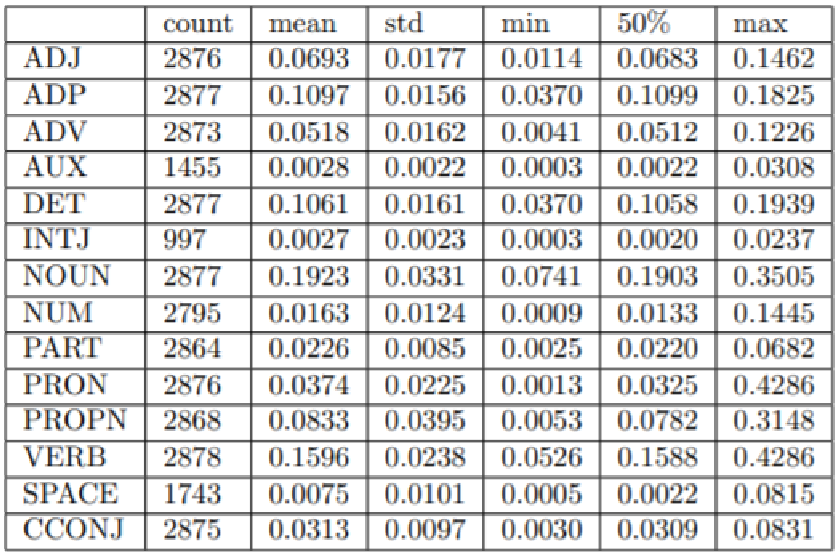
7.2 Statistics of Parts of Speech and Named Entities

7.2.1 Parts of Speech



	count	mean	std	min	50%	max
ADJ	2881	0.0554	0.0188	0.0009	0.0550	0.1455
ADP	2887	0.1086	0.0205	0.0009	0.1090	0.1791
ADV	2809	0.0372	0.0174	0.0004	0.0364	0.1304
AUX	900	0.0025	0.0018	0.0003	0.0020	0.0171
DET	2881	0.1010	0.0210	0.0014	0.1020	0.1852
INTJ	736	0.0025	0.0025	0.0003	0.0018	0.0274
NOUN	2890	0.1825	0.0371	0.0076	0.1809	0.4118
NUM	2861	0.0317	0.0293	0.0013	0.0227	0.3077
PART	2806	0.0215	0.0096	0.0006	0.0207	0.0877
PRON	2671	0.0330	0.0225	0.0005	0.0285	0.1398
PROPN	2884	0.1276	0.0662	0.0042	0.1159	0.6282
VERB	2888	0.1472	0.0319	0.0025	0.1500	0.2917
SPACE	1695	0.0096	0.0108	0.0002	0.0040	0.0742
CCONJ	2854	0.0282	0.0096	0.0011	0.0273	0.0720

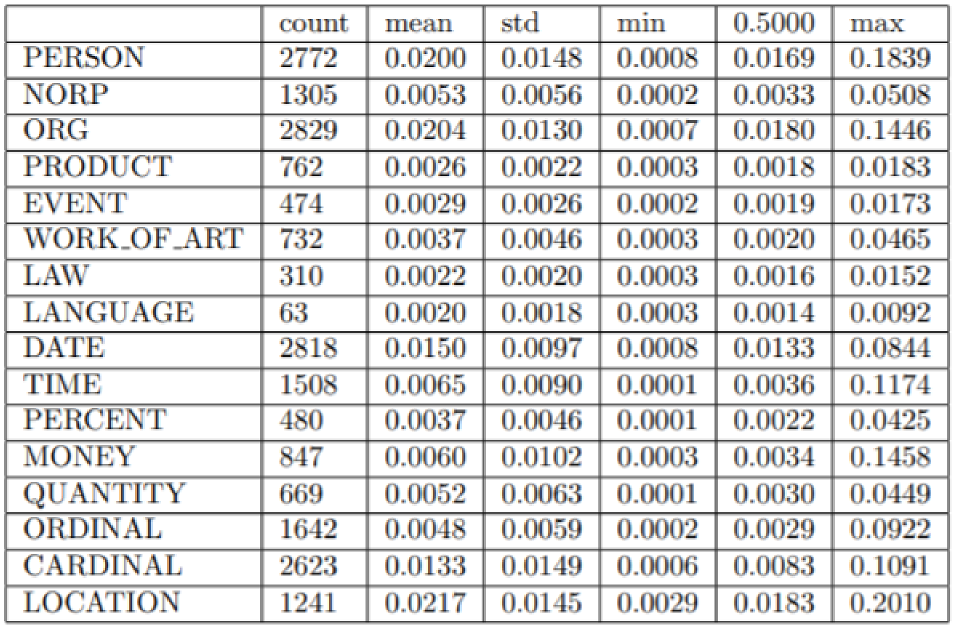
The above Table has the Statistics for Parts of Speech for Regular Ariticles.



	count	mean	std	min	50%	max
ADJ	2876	0.0693	0.0177	0.0114	0.0683	0.1462
ADP	2877	0.1097	0.0156	0.0370	0.1099	0.1825
ADV	2873	0.0518	0.0162	0.0041	0.0512	0.1226
AUX	1455	0.0028	0.0022	0.0003	0.0022	0.0308
DET	2877	0.1061	0.0161	0.0370	0.1058	0.1939
INTJ	997	0.0027	0.0023	0.0003	0.0020	0.0237
NOUN	2877	0.1923	0.0331	0.0741	0.1903	0.3505
NUM	2795	0.0163	0.0124	0.0009	0.0133	0.1445
PART	2864	0.0226	0.0085	0.0025	0.0220	0.0682
PRON	2876	0.0374	0.0225	0.0013	0.0325	0.4286
PROPN	2868	0.0833	0.0395	0.0053	0.0782	0.3148
VERB	2878	0.1596	0.0238	0.0526	0.1588	0.4286
SPACE	1743	0.0075	0.0101	0.0005	0.0022	0.0815
CCONJ	2875	0.0313	0.0097	0.0030	0.0309	0.0831

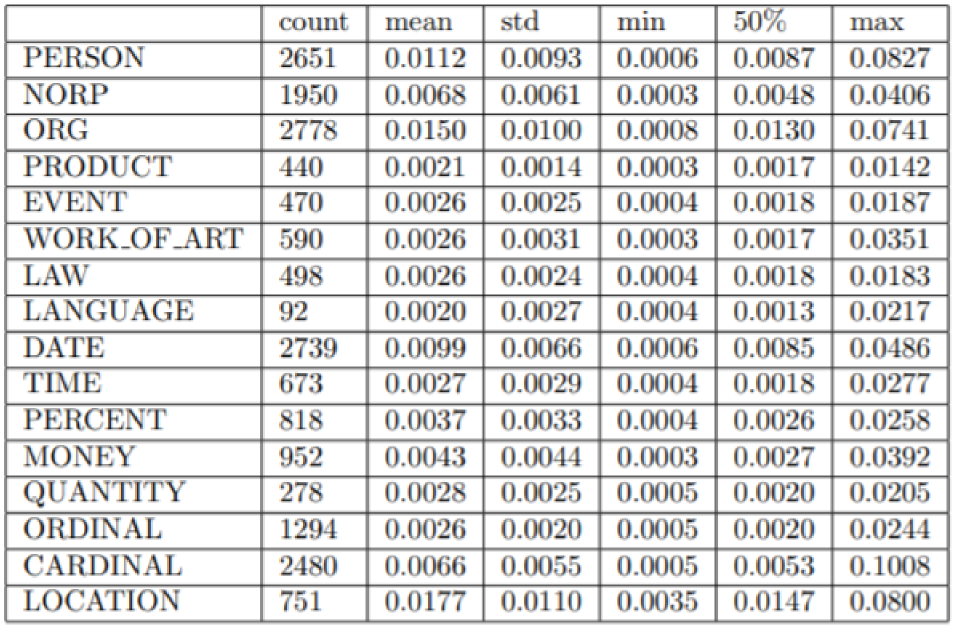
The above Table has the Statistics for Parts of Speech for Non-regular (Opinion) Ariticles.

7.2.2 Named Entities



	count	mean	std	min	0.5000	max
PERSON	2772	0.0200	0.0148	0.0008	0.0169	0.1839
NORP	1305	0.0053	0.0056	0.0002	0.0033	0.0508
ORG	2829	0.0204	0.0130	0.0007	0.0180	0.1446
PRODUCT	762	0.0026	0.0022	0.0003	0.0018	0.0183
EVENT	474	0.0029	0.0026	0.0002	0.0019	0.0173
WORK_OF_ART	732	0.0037	0.0046	0.0003	0.0020	0.0465
LAW	310	0.0022	0.0020	0.0003	0.0016	0.0152
LANGUAGE	63	0.0020	0.0018	0.0003	0.0014	0.0092
DATE	2818	0.0150	0.0097	0.0008	0.0133	0.0844
TIME	1508	0.0065	0.0090	0.0001	0.0036	0.1174
PERCENT	480	0.0037	0.0046	0.0001	0.0022	0.0425
MONEY	847	0.0060	0.0102	0.0003	0.0034	0.1458
QUANTITY	669	0.0052	0.0063	0.0001	0.0030	0.0449
ORDINAL	1642	0.0048	0.0059	0.0002	0.0029	0.0922
CARDINAL	2623	0.0133	0.0149	0.0006	0.0083	0.1091
LOCATION	1241	0.0217	0.0145	0.0029	0.0183	0.2010

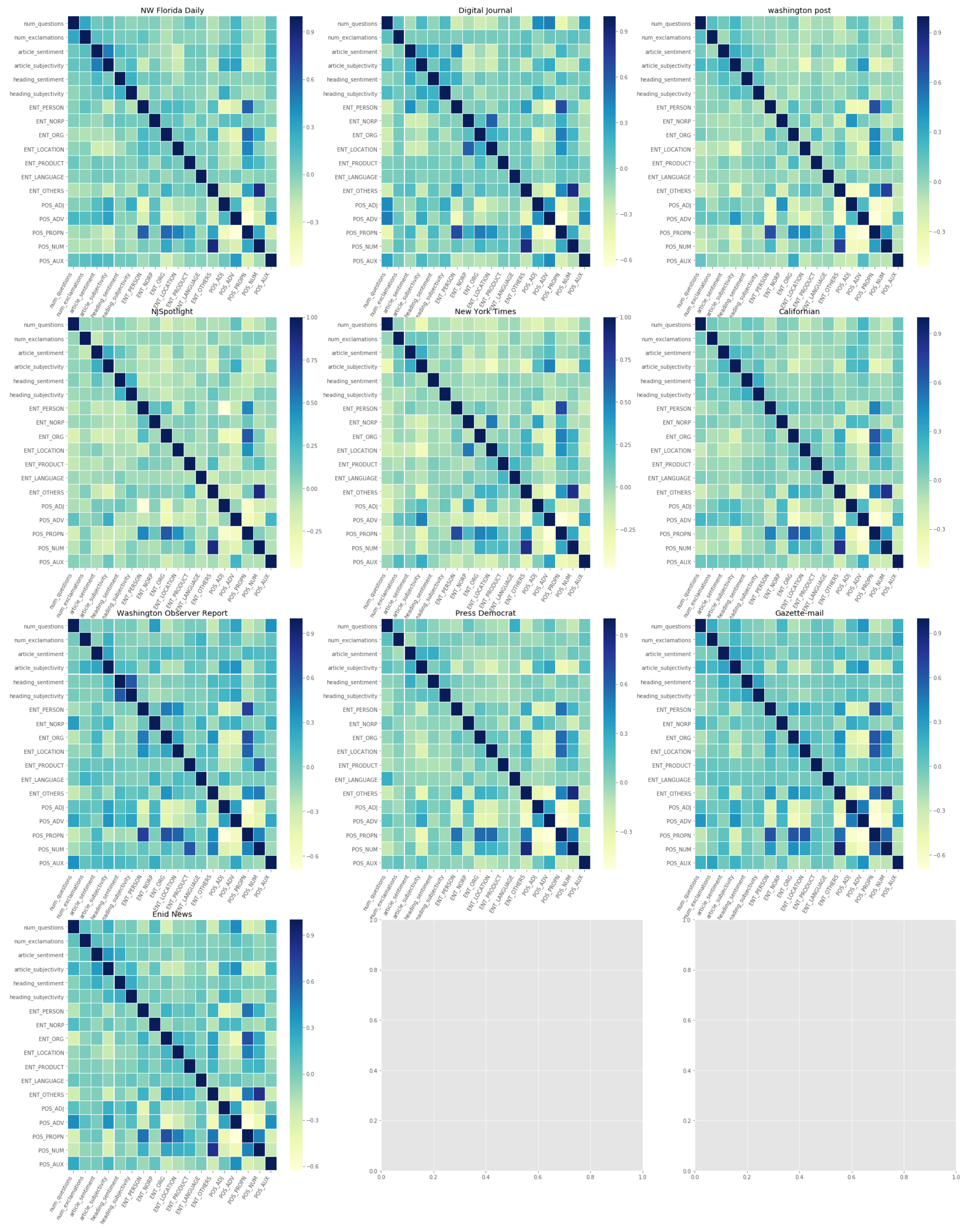
The above Table has the Statistics for the Named Entities for Regular Ariticles.



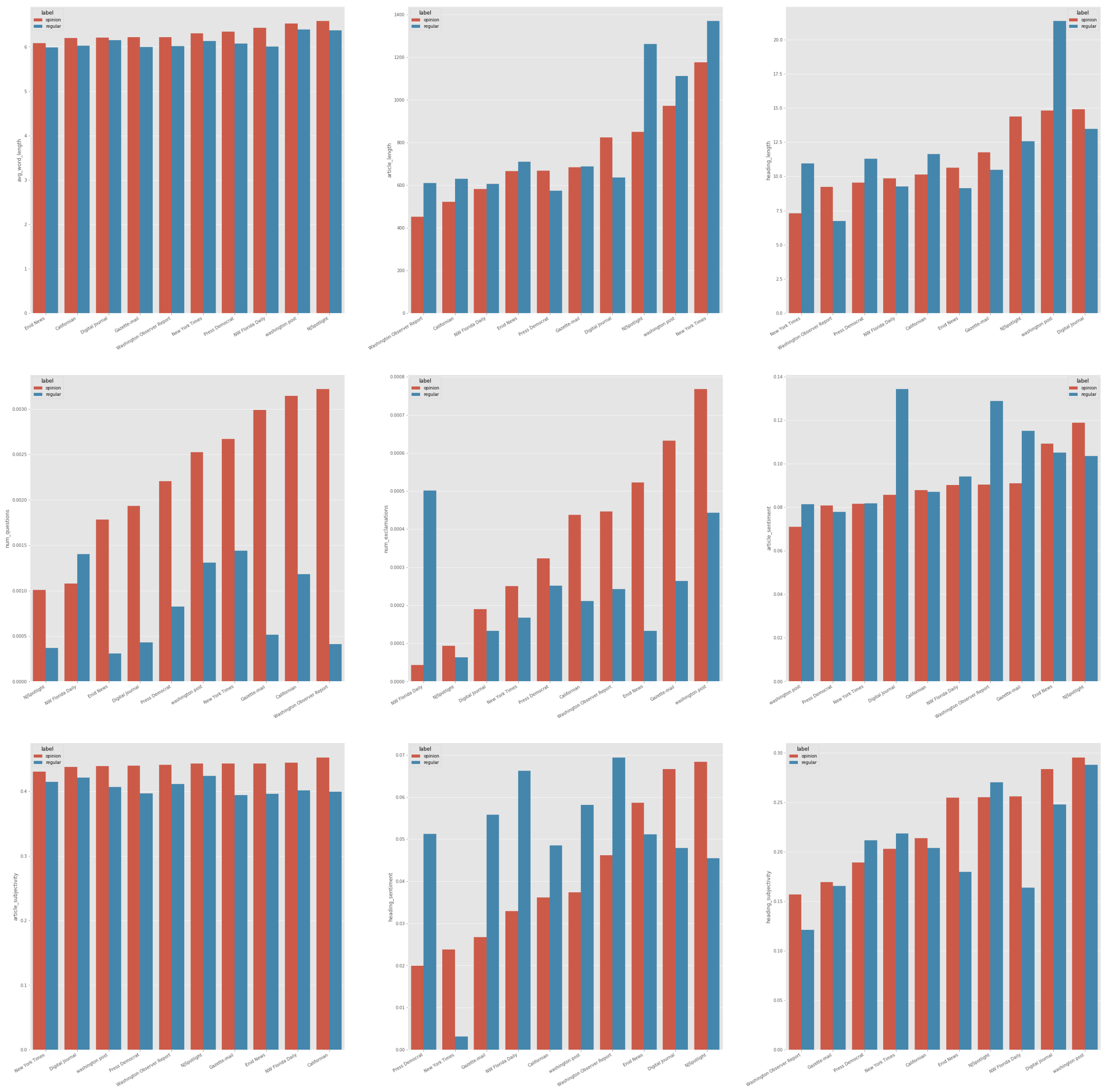
	count	mean	std	min	50%	max
PERSON	2651	0.0112	0.0093	0.0006	0.0087	0.0827
NORP	1950	0.0068	0.0061	0.0003	0.0048	0.0406
ORG	2778	0.0150	0.0100	0.0008	0.0130	0.0741
PRODUCT	440	0.0021	0.0014	0.0003	0.0017	0.0142
EVENT	470	0.0026	0.0025	0.0004	0.0018	0.0187
WORK_OF_ART	590	0.0026	0.0031	0.0003	0.0017	0.0351
LAW	498	0.0026	0.0024	0.0004	0.0018	0.0183
LANGUAGE	92	0.0020	0.0027	0.0004	0.0013	0.0217
DATE	2739	0.0099	0.0066	0.0006	0.0085	0.0486
TIME	673	0.0027	0.0029	0.0004	0.0018	0.0277
PERCENT	818	0.0037	0.0033	0.0004	0.0026	0.0258
MONEY	952	0.0043	0.0044	0.0003	0.0027	0.0392
QUANTITY	278	0.0028	0.0025	0.0005	0.0020	0.0205
ORDINAL	1294	0.0026	0.0020	0.0005	0.0020	0.0244
CARDINAL	2480	0.0066	0.0055	0.0005	0.0053	0.1008
LOCATION	751	0.0177	0.0110	0.0035	0.0147	0.0800

The above Table has the Statistics for the Named Entities for Non-regular (Opinion) Ariticles.

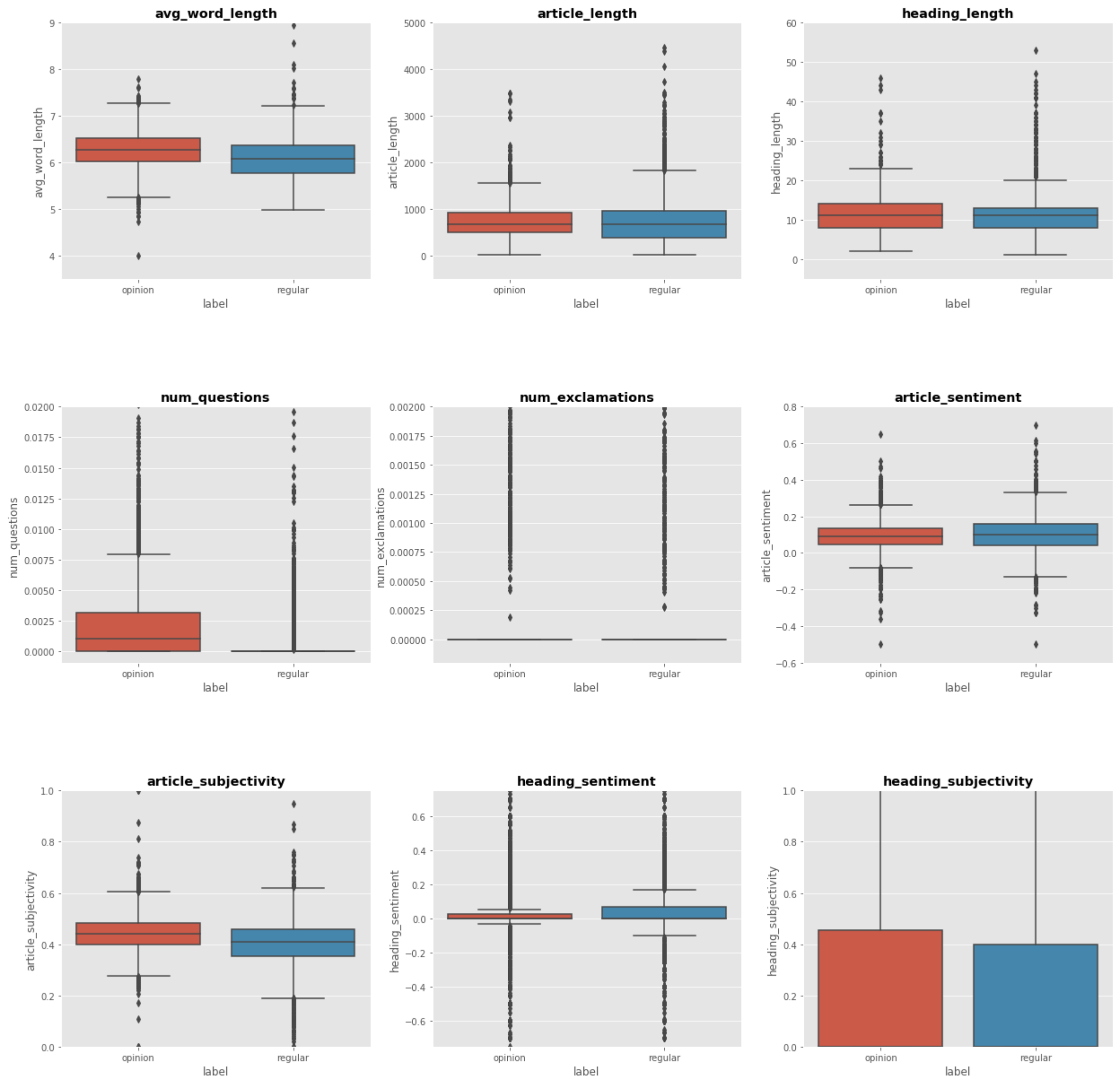
7.3 Covariance of features



The above figure is the covariance among features for different news sources



The above figure is the corelation between features and targets for different news sources



The above figure is the boxplot distribution for various features for regular and non-regular (opinion) articles

7.4 List of features

7.4.1 General-purpose text feature extraction:

From processed text

- avg_word_length - Average length of a word in a string.
- Sentiment analysis
  - VADER - find sentiments using VADER, which is a lexicon-based sentiment analysis extracted using social media.
  - AFINN - find sentiment using AFINN, with an integer between -5 (negative) and +5 (positive).
  - TextBlob - Sentiment of the article using TextBlob. Range is [-1.00, 1.00] with -1.00 implying negative and 1.00 implying positive sentiment.

From raw article text

- heading_length - Length of the title
- article_length - Length of the article
- num_questions - Number of questions marks in the article
- num exclamations - Number of exclamation marks in the article
- article_sentiment - Sentiment of the article using TextBlob. Range is [-1.00, 1.00] with -1.00 implying negative and 1.00 implying positive sentiment.
- article_subjectivity - Subjectivity of the article using TextBlob. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.
- heading_sentiment - Sentiment of the heading
- heading_subjectivity - Subjectivity of the heading
- ENT_PERSON, ENT_NORP, ENT_ORG, ENT_LOCATION, ENT_PRODUCT, ENT_LANGUAGE, ENT_OTHERS - Number of different named entities in the article. Refer to the EDA section for the description of the entities.
- POS_ADJ, POS_ADV, POS_PROPN, POS_NUM, POS_AUX - Number of different parts-of-speech tags in the article. Refer to the EDA section in Report 1 for the description of the POS tags.

7.4.2 Data-specific text feature extraction (from processed text)

- Count Vectorizer
  - Character-based - Used a ngram range of (1, 2), removed english stopwords and only used top 1000 frequent features.
  - Word-based - Used a ngram range of (1, 3), removed english stopwords and only used top 1000 frequent features.
- TF-IDF Vectorizer
  - Character-based - Used a ngram range of (1, 2), removed english stopwords and only used top 1000 frequent features.
  - Word-based - Used a ngram range of (1, 3), removed english stopwords and only used top 1000 frequent features.

7.5 List of tuning parameters for algorithms

- n_estimators, max_depth (Random Forest)
- max_depth (Decision Tree)
- n_estimators, max_depth, learning rate (Gradient Boosting)

7.5.1 Logistic Regression

- C : Inverse of regularization strength
- penalty ('l1', 'l2', 'elasticnet', 'none') : Penalty norm term for Logsitic regression

7.5.2 Random Forest

- n_estimators : The number of trees in the forest
- max_depth : The Maximum depth of the tree

7.5.3 Gradient Boosting

- n_estimators : The number of boosting stages to perform
- max_depth : Maximum depth of the individual regression estimators
- learning_rate : Shrink in the contribution of each tree

7.5.4 LightGBM

- max_depth - Maximum depth to which the trees should be grown
- num_leaves - Mumber of leaves in one tree
- n_estimators - Number of trees to fit
- learning_rate - Boosting learning rate

7.6 List of Evaluation Metrics

- Precision
- Recall
- TNR
- Accuracy
- Balanced Accuracy
- F1 Score
- Matthews correlation coefficient
- Macro average precision
- Macro average recall
- Macro average F1 Score
- Weighted average precision
- Weighted average recall
- Weighted average F1 Score

8 Contributions (Alphabetical order)

8.1 Work

- Corelation Analysis - Sarang
- Data Preparation - Sarang, Ujjwal
- Baseline and all linear models - Nirali
- LightGBM - Ujjwal
- XLNet - Ujjwal
- RNN and all LSTM Models - Aastha
- BERT - Ameya, Sarang
- Metro Winnipeg data collection and DTM - Ameya

8.2 Report

- Feedback from Report 1 - Ameya, Sarang
- Data Preparation - Sarang, Ujjwal
- Models - Aastha, Ameya, Nirali, Sarang, Ujjwal
- Results - Aastha, Ameya, Nirali, Sarang
- Other Progress and Next Steps - Ameya
- Appendix - Ameya, Sarang