Building NLP Models to Classify News Articles

ReThink Media Data Science Discovery Spring 2022

Researchers -

Atmika Sarukkai, Daniel Chung, Grace Chen, Tiffany Liang

Advisors -

Laura Nixon, Lana Elauria, ReThink Media, Alex Bruefach, UC Berkeley

Introduction -

Our project aimed to build machine learning models to analyze the content of news articles. The technical goals of this project were to develop classifiers that predict the topic of article quotes and the type of speakers quoted in the articles.

Data -

Our data comes from a dataset of over 72,047 hand-annotated articles about nuclear weapons issues published in U.S. news outlets from 2011 - 2021, which were collected from web scraping and from the Factiva news database. Data attributes include publication date, author name, publisher, article body, and hand-annotated labels for topic, speaker type, and other variables.

Topic Classification

One goal for this project was to classify the topic of a news article, specifically based on the countries and treaties mentioned in the article. For feature extraction, we looked for regex

matches on country and treaty names within articles, taking into account the variations, stemming, and abbreviations of the names. We used these matches to calculate the frequency of countries mentioned in different parts of the article. The features used in the final model are:

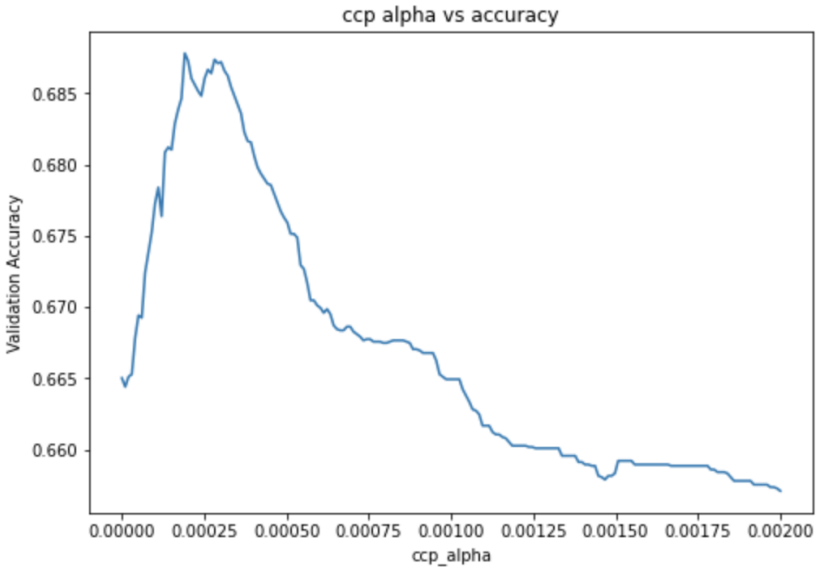
Feature	Description
most_frequent_country_3	Countries where topic message equals the most frequent country for 3 sentences surrounding country instance
most_frequent_country_full	Countries where topic message equals the most frequent country in the full text
most_frequent_headline	Countries where topic message equals the most frequent country in the headline
max_frequency_context	Frequency of country with the maximum frequency in in the content
max_frequency_full	Frequency of country with the maximum frequency in in the full article content
max_frequency_headline	Frequency of country with the maximum frequency in in the headline
majority	Country with the maximum frequency among all countries mentioned in the article
Treaties	Treaty mentioned in the entire article body
Prop Overlapped (Messages, Headline)	Proportion of words in the topic message column and headline that are the same
Prop Overlapped (Messages, 10% Content)	Proportion of words in the topic message column and the first 10% of article content that are the same

After extracting the features that were able to predict topics at a fairly high accuracy individually (>50% accuracy), we tested three classification models, specifically Logistic Regression, Decision Tree, and Random Forest.

A logistic regression model achieved 0.685 accuracy without any parameter tuning, the lowest score of the three models, so we focused on tuning the decision tree-based models for our final model.

Decision Tree

A decision tree model with all features and default hyperparameters achieved an accuracy of 0.703. We then introduced pruning to the model by adjusting the ccp_alpha parameter. The optimal value for this parameter was found to be 0.00019 after a grid search.

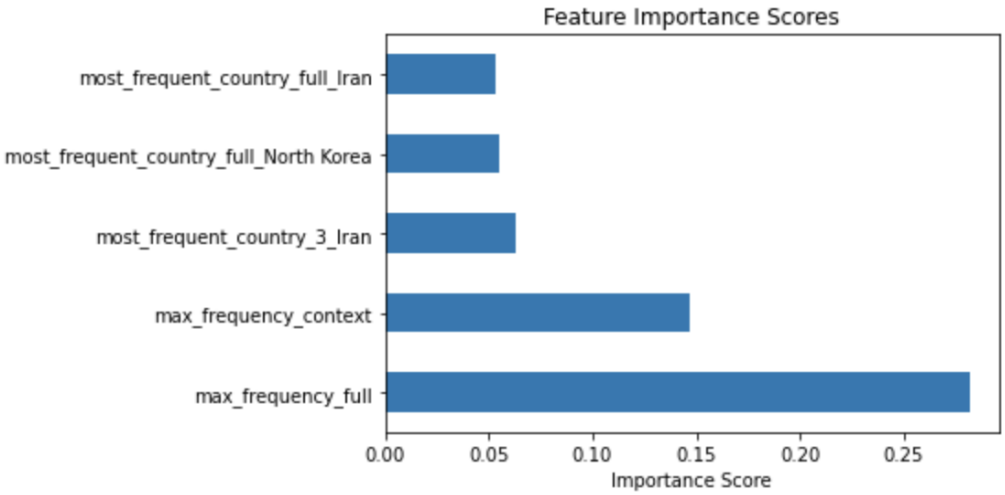


After parameter tuning, the accuracy of the decision tree model increased to 0.715.

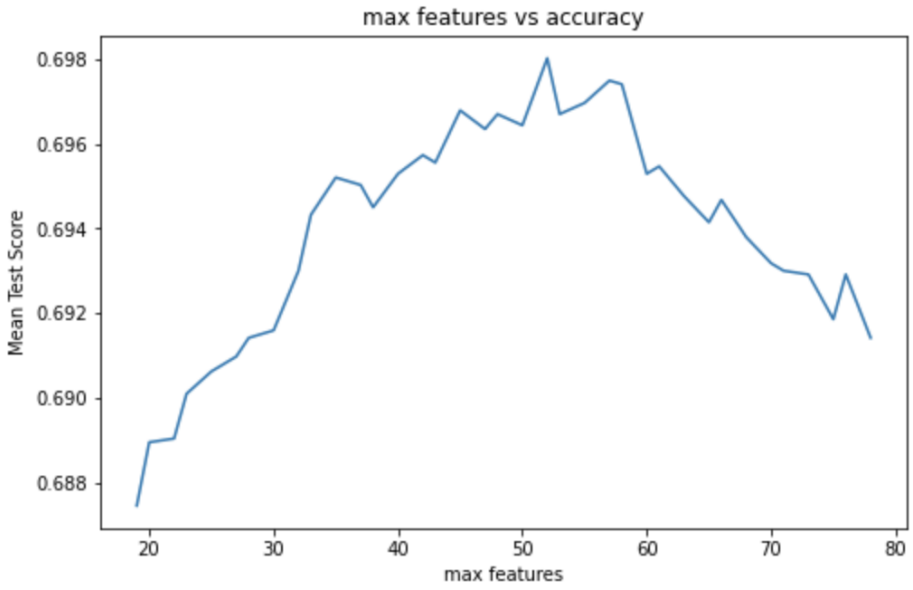
Random Forest

A default random forest model, with all features included, achieved an accuracy 0.711. We also evaluated feature importances within the random forest, and the five most important features are shown below:

From the importance scores, we can see that the countries frequently mentioned within an article provide the most useful information about the overall topic.



To further finetune the Random Forest model, we also experimented with dimensionality reduction. Below is a plot of the number of features against model accuracy:



The highest-performing random forest model had a maximum of 52 features, and the resulting accuracy was 0.717.

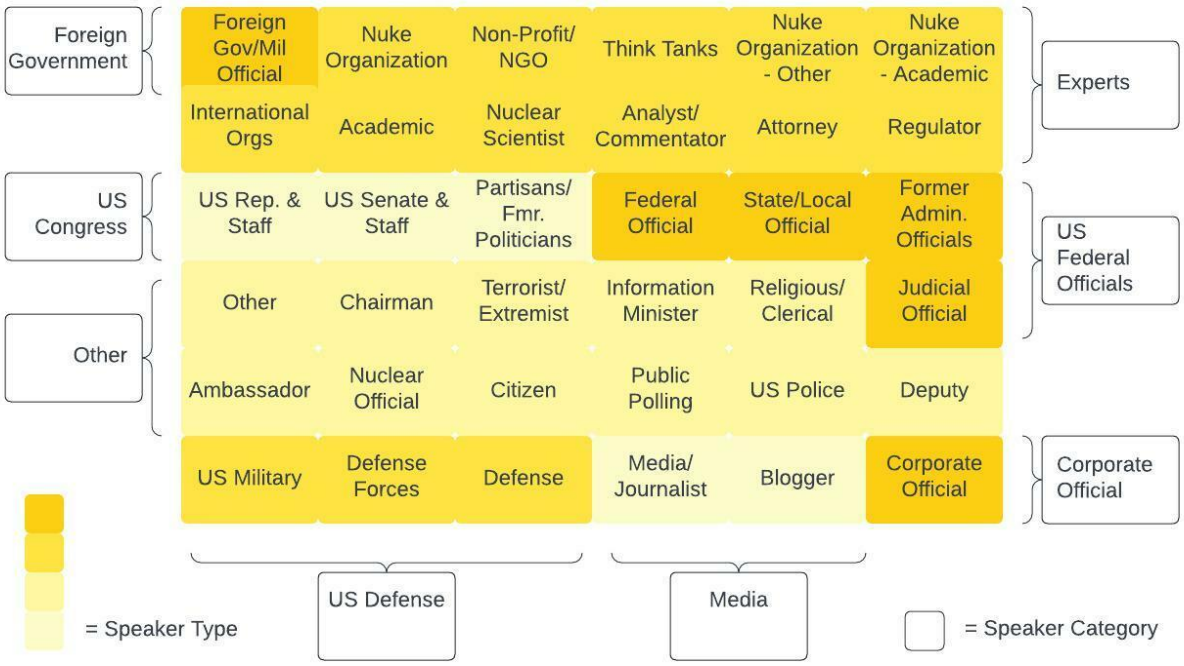
Summary Table of Model Accuracies:

Classification Model	**Test Set Accuracy**
Logistic Regression (all features)	0.685
Random Forest	0.711
Random Forest with Parameter Tuning (Maximum # Features)	0.717
CART	0.700
CART (with ccp_alpha tuning)	0.715
Random Forest 95% CI	[0.645, 0.679]
CART 95% CI	[0.699, 0.734]

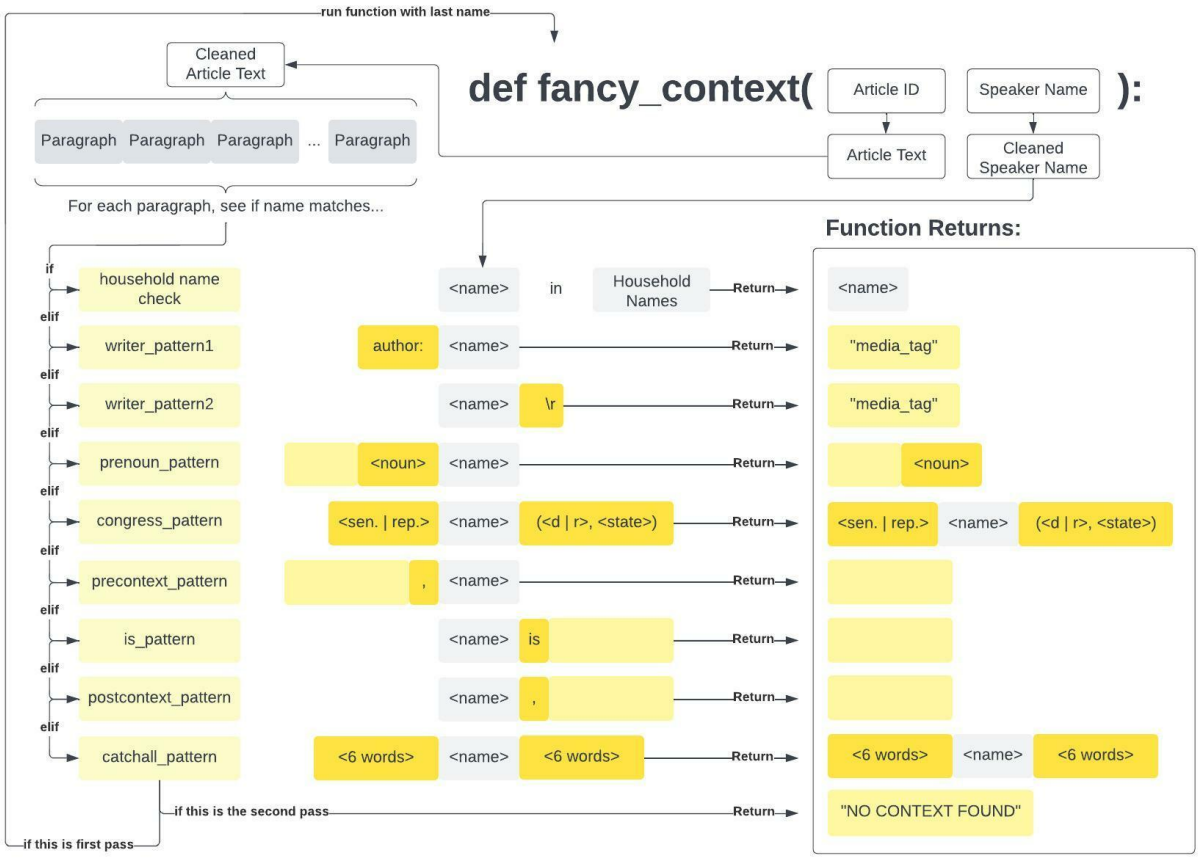
Speaker Classification

Our next goal was to classify a speaker name from an article into its corresponding speaker group (e.g. US government, military, academic, etc.). We used regex and NLTK to obtain the context surrounding the name in the article. We also created lists of indicator features—such as “democrat”, “republican”, and ”senator” for the “US Congress” category—from TD-IDF results and our own observations of the data. We used heatmaps to visualize how frequent these features appeared in the speaker groups and to explore their potential utility in classification.

Speaker types with current categorization scheme



Context extraction process diagram



Wikipedia API

An alternate approach we tried was to use the Wikipedia API to directly “search” the name on Wikipedia and pull information from the Wikipedia page. However, this presented some challenges. First of all, some names had more famous counterparts on Wikipedia, so we would be seeing information about actors, musicians, fictional characters, etc. that clearly would not actually be in new articles about nuclear topics. Secondly, the runtime to use the Wikipedia API was incredibly slow, and we would still have to run additional code on the extracted text to classify the name. Ultimately, we decided it was better to focus on classifying the name from the article context.

Logistic Regression

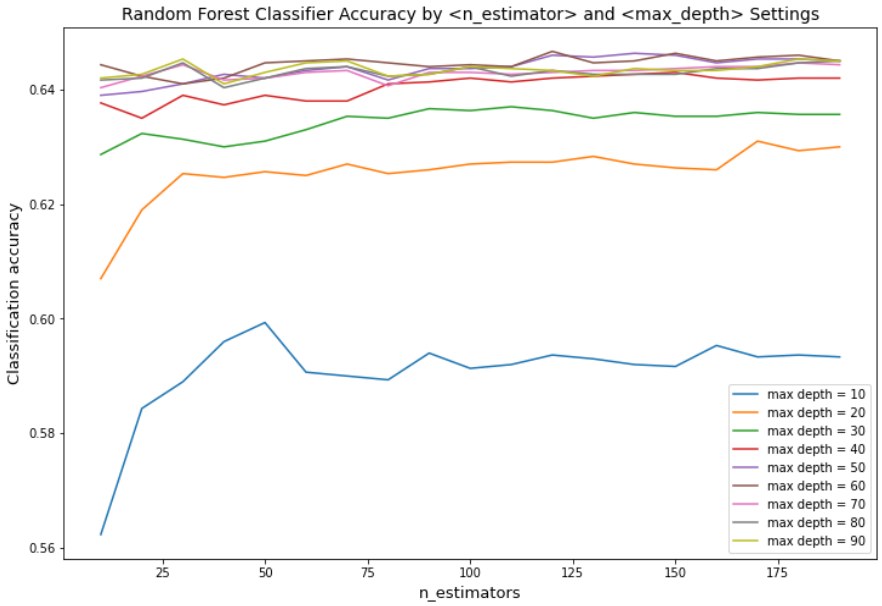
With logistic regression and our selected features, we were able to achieve an accuracy of 67.57%. We believe that this could be improved through a more robust context extraction function (making sure the extract is actually related to the speaker and is comprehensive) and better feature selection (prioritizing ones that are distinct between groups), especially as the model appears to overpredict the foreign government speaker group.

SVM

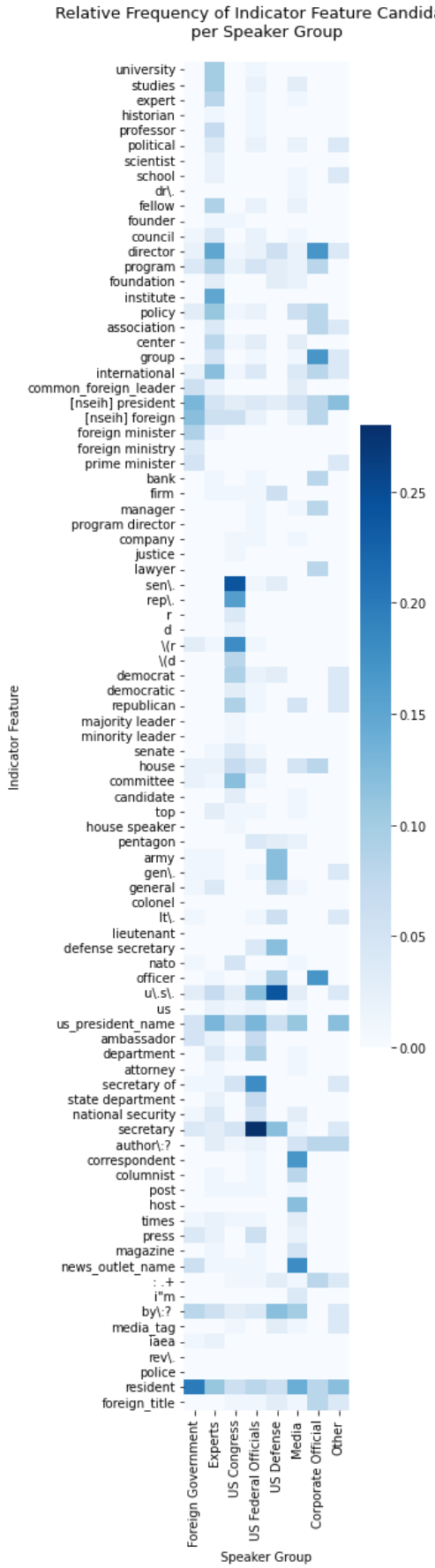
We also ran our indicator features through the support vector machine algorithm and got an accuracy of 66.73%. We then tried hyperparameter tuning using sci-kit learn's GridSearchCV, but we were only able to improve the accuracy by 1-2%.

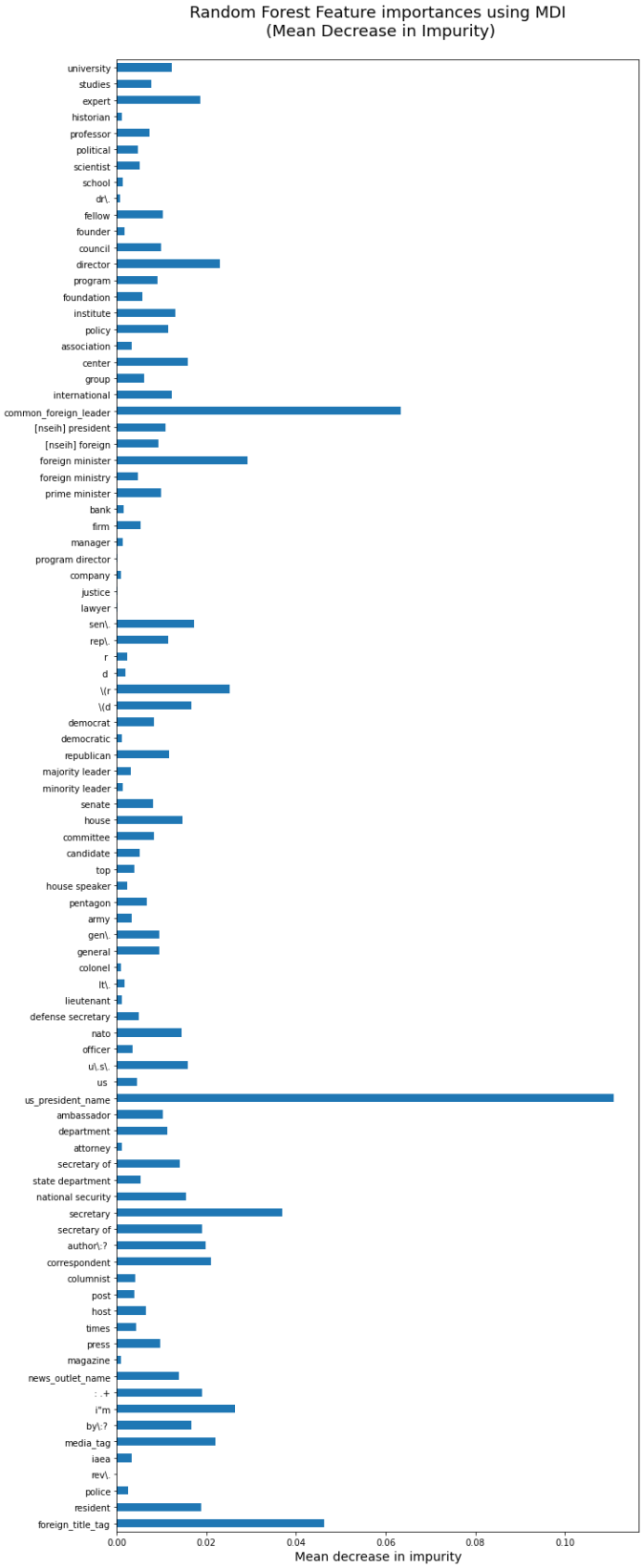
Random Forest

The random forest had 67.37% accuracy but performed the best on average. We tried hyperparameter tuning to achieve a better result but could not find a time-efficient method to run gridsearch. However, this model seemed to be the most promising out of the three we tried.



Relative frequency of indicators per category and their random forest importances





Random forest error analysis subplots