Final Project Report

Group 3: Jehan Bugli

Overview

This project tags congressional bills with one of 32 policy area designations[1], replacing an important yet tedious task that is currently completed manually by the Congressional Research Service (CRS).

The CRS assists legislators with policy analysis[2], official bill summaries[3], and more; however, their job is becoming more difficult due to rapidly increasing bill introductions causing significant backlogs for official text releases[4], which in turn affects third parties reliant upon their processing efforts.

Automated policy area assignments appeared as the natural choice for an NLP-based solution. Policy area designations are relatively static, remaining essentially unchanged since their implementation, providing a stable modeling target. In addition, legislative text is uniquely suitable for this purpose; bill text and code citations are frequently re-used across bills covering similar topics, and legislative terminology follows very precise and formally defined specifications.

This experiment's goal is to provide practical utility; as a result, this primarily leverages classical methods. These methods have two key advantages for policy area tagging relative to state-of-the-art neural networks:

- **Interpretability:** results can be traced back with relatively straightforward reasoning, which is critical for real-world adoption; analysts should be able to explain the logic behind these labels to various stakeholders (members of Congress, lobbyists, etc.).
- **Cost:** models are significantly cheaper to train and run for inference, which is especially useful for a resource-constrained organization like the CRS.

This report will:

- Discuss the data collection and processing efforts involved
- Describe the chosen modeling approach and experiment setup
- Outline the experiment results and conclusions

Data collection and processing

Data was sourced from the GovInfo bulk data repository which stores XML-formatted bill data for recent congressional sessions; this dates back to the 113th session (2013-2015).[5]

This includes downloaded text for each bill version in the available sessions. For instance, if a bill received amendments between introduction and enactment, both the amended and original versions would be included. These were saved as a collection of session-specific .zip files.

Apart from bill text, sourcing included a separate collection of zipped XML files with associated data for each bill, including policy area tags and other CRS-processed items, in the same format as noted earlier.

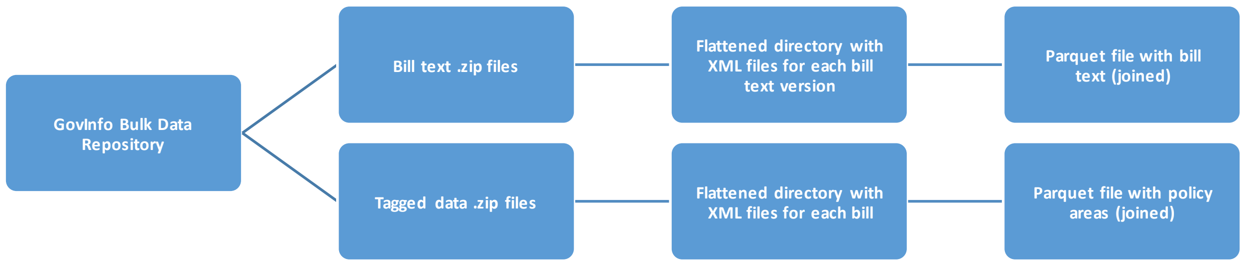
Before further processing, both folders were "flattened", unzipping contents into two main directories.

Each flattened folder was processed to form a tabular dataset. The folder with bill text was processed to generate a parquet file storing each file's name and bill text.

Bill text was cleaned using BeautifulSoup's XML parser; the data collection script searches for valid legislative body tags (e.g. 'legis-body', 'preamble') and discards components such as the title, publishing date, sponsors, sessions, and more. While some of these items certainly may have provided some predictive value, such modeling may step outside of the NLP realm and muddy the results' interpretability.

The folder with additional bill data was processed to generate a parquet file storing policy areas alongside text version names.

These text version names were used to join the two files and construct the finalized input dataset, containing 115,100 tagged bill text samples.


GovInfo Bulk Data Repository
Bill text .zip files
Tagged data .zip files
Flattened directory with XML files for each bill text version
Flattened directory with XML files for each bill
Parquet file with bill text (joined)
Parquet file with policy areas (joined)

Modeling approaches

At its core, the experimental approach uses Term Frequency – Inverse Document Frequency (TF-IDF) to extract useful features. TF-IDF emerged out of pioneering information retrieval research as a method to quantify word importance.[6]

This combines two key metrics:

- **Term Frequency (TF):** How frequently a term appears within a single document
- **Inverse Document Frequency (IDF):** How rare a term is across the entire document collection

The TF-IDF score multiplies these metrics to weight terms that are frequent within a given document but relatively rare across the entire collection of documents. This weighting can prove useful for many tasks, including feature extraction for classification.

After testing parameter combinations, TF-IDF vectorization was tuned by:

- Removing common English stop-words
- Capturing individual words and meaningful two-word phrases
- Ignoring terms that only appear in a single document
- Removing terms that appear in over 85% of documents

Three different models were trained on these extracted features:

- **LinearSVC (Support Vector Classifier):** attempts to find the "best" separating hyperplane that divides the TF-IDF feature space, distinguishing between different policy area categories. It aims to maximize the distance between the closest data points (support vectors) of different classes.[7]
- **Logistic Regression:** predicts the probability of an instance belonging to a particular class using a logistic (sigmoid) function applied to a linear combination of the input features (TF-IDF scores).[8]
- **Multinomial Naive Bayes:** calculates the probability of a document belonging to each policy area based on the frequency of terms within it, making a "naive" assumption that the features are independent of each other given the class.[9]

These were compared using grid search cross-validation; this explores various parameter combinations across the TF-IDF vectorizer and these classifiers, comparing performance on weighted recall to return the best-performing combination.

For each combination, this search performs 3-fold cross-validation, where the data is split into 3 subsets; for each “fold”, a model is trained using 2 subsets and the remaining one serves as a validation set. This prevents overfitting, where the model is unable to generalize effectively from its training data.[10]

The grid search included different hyperparameter configurations, such as:

- **C**: A regularization parameter penalizing misclassified examples; a larger C value prioritizes accurate classification but risks overfitting.[11]
- **Maximum iterations**: The number of allowed iterations, altered to ensure that models are converging

Macro recall was the primary metric. Recall is the ratio of true positives over all actual positive instances for a class (including both true positives and false negatives); this methodology averages recall across all classes without regard for imbalance.[12] This decision is made out of perceived practical utility; for a policy affairs professional, catching every relevant piece of legislation (each “true positive”) for their potentially niche field is of the utmost importance, even if that arrives alongside a number of false positives.

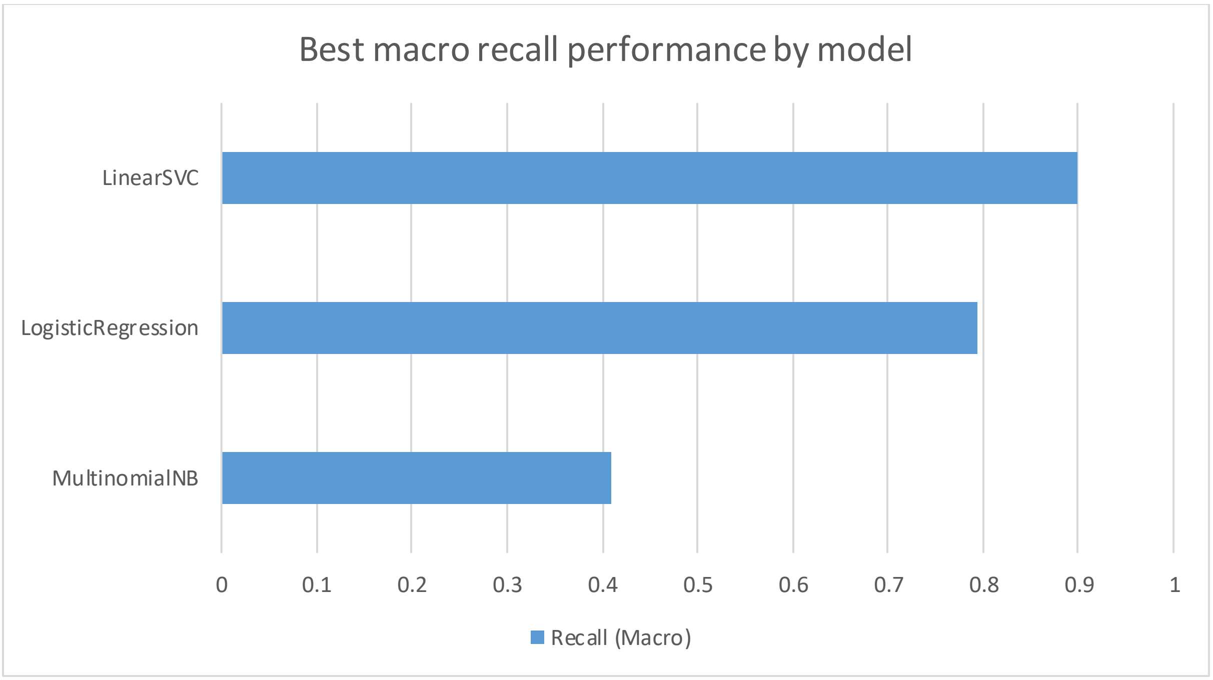
Results

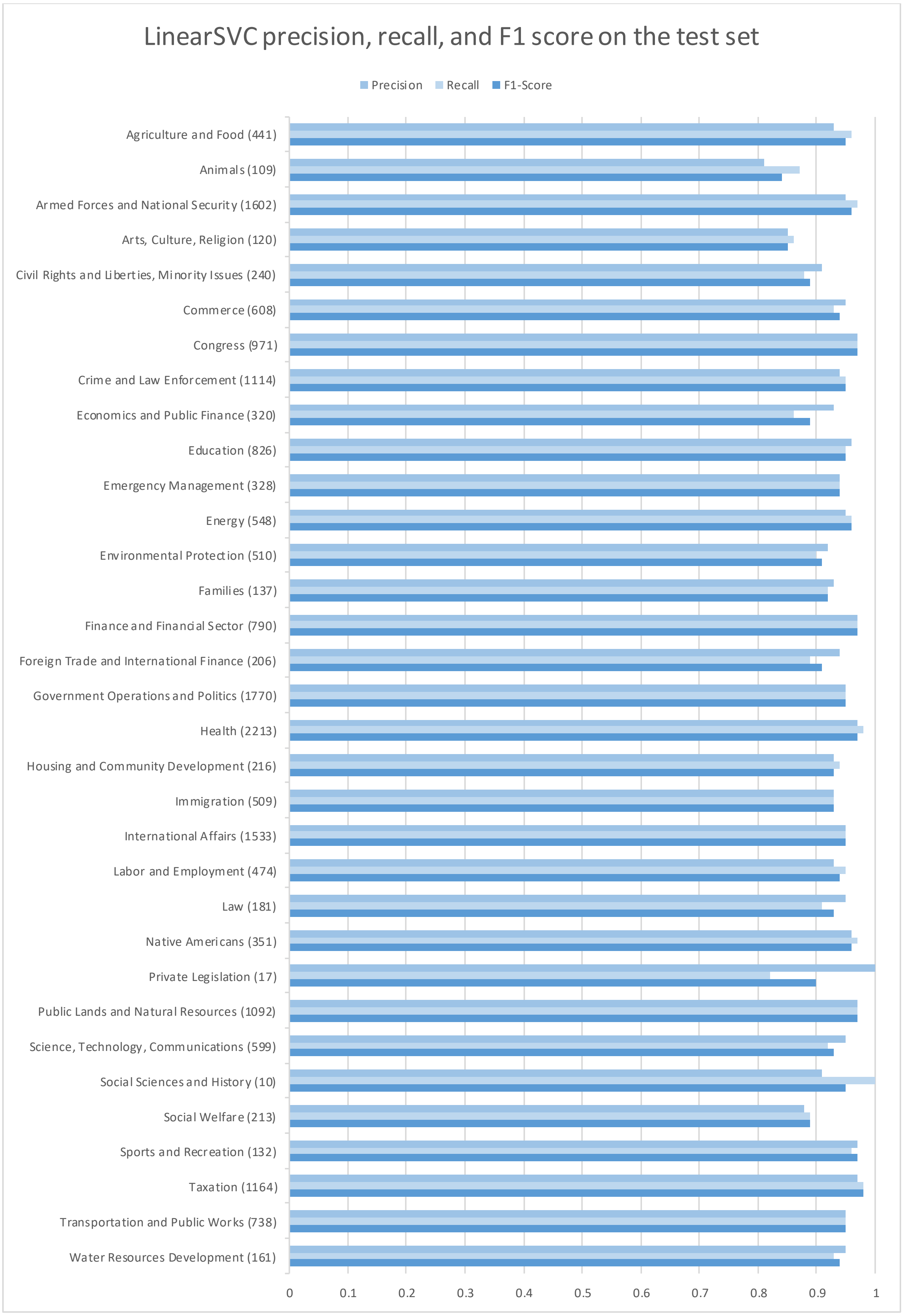
The linear SVC model displayed the strongest performance by far, reaching ~90% macro recall with grid search hyperparameter tuning!

This served as an encouraging sign for the classical approach. Logistic regression followed close behind, while multinomial naïve Bayes was a distant third.

The graphs below display:

- Macro recall performance across the three tested models, including their best results from grid search tuning
- Area-by-area recall, precision, and F1-score results for the best-performing LinearSVC model




LinearSVC precision, recall, and F1 score on the test set
Precision
Recall
F1-Score
Agriculture and Food (441)
Animals (109)
Armed Forces and National Security (1602)
Arts, Culture, Religion (120)
Civil Rights and Liberties, Minority Issues (240)
Commerce (608)
Congress (971)
Crime and Law Enforcement (1114)
Economics and Public Finance (320)
Education (826)
Emergency Management (328)
Energy (548)
Environmental Protection (510)
Families (137)
Finance and Financial Sector (790)
Foreign Trade and International Finance (206)
Government Operations and Politics (1770)
Health (2213)
Housing and Community Development (216)
Immigration (509)
International Affairs (1533)
Labor and Employment (474)
Law (181)
Native Americans (351)
Private Legislation (17)
Public Lands and Natural Resources (1092)
Science, Technology, Communications (599)
Social Sciences and History (10)
Social Welfare (213)
Sports and Recreation (132)
Taxation (1164)
Transportation and Public Works (738)
Water Resources Development (161)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Discussion

These results support the notion that a classical approach can form a very robust classification mechanism for the federal legislation corpus.

While the LinearSVC model displayed excellent results, however, performance differed across different policy areas as displayed in the prior graph!

For instance, "Private Legislation" saw especially low recall values, which is reasonable given its relative infrequency and lack of topic specificity.[13] There was stability with respect to class imbalance, likely thanks to the macro recall objective; "Social Sciences and History", with only 10 test set instances, saw perfect recall.

While discrepancies are noteworthy and warrant further investigation, the generally robust performance is a positive sign for practical implementation. All class predictions carry a reasonably high degree of certainty, and while policy areas are assigned in a mutually exclusive manner, the bills themselves are not as neatly partitioned! For instance, "Foreign Trade and International Finance" could be viewed as adjacent to "Finance" as well as "International Affairs", and a resulting misclassification (relative to a CRS analyst's decision) may still be highly relevant.

The poor MultinomialNB performance suggests that the "naïve" assumption doesn't hold, such that individual term impacts on categorization cannot be viewed independently!

This makes sense intuitively; compound terms or phrases found jointly in a document could imply classifications that the individual terms would not, such as "pharmacy benefit manager".

Initially, the poor performance led to suspicions regarding TF-IDF suitability, since MultinomialNB is usually used with feature counts rather than these vectors; however, the scikit-learn documentation explicitly notes this as a valid approach for practical use.[14]

Conclusion

Overall, the results ended up as a welcome surprise; while there were initial concerns about a potential trade-off between performance and interpretability, classical methods appear fully capable of supporting real-world tagging applications!

There are a number of avenues for further investigation and application.

- **Representing related classes in scoring:** As noted earlier, while policy areas are mutually exclusive, legislative contents can include relevant impacts across many discrete fields. Judging this model by the CRS-assigned class may not be sufficient to judge practical use; building out a representation of “adjacent” policy areas and their relationships could allow us to evaluate model performance in a more realistic manner.
- **Generalization:** Testing whether this approach could be generalized to other text corpora is also worth considering. For instance, state-level legislation, Federal Register rules, and others are classified fully independently (if at all)! A unified classification system could help the public more easily navigate the mass of available legal materials to identify relevant items.
- **Stop-words:** Results could potentially be improved to a degree by refining stop-words to target common legal phrasing; the current results use the standard TfidfVectorizer English set, which may be a poor fit. Unfortunately, the effort involved in compiling and leveraging a custom list moved that outside of this experiment’s scope.
- **Retroactive or custom classification:** While legislative topics have shifted over time, the labels across the digitized corpus remain static. If the CRS wants to add a new label (e.g. “Artificial Intelligence”), they could potentially re-train a model on an expanded label set (adding the new label manually for a session) and retroactively reclassify older bills to reflect that change. Alternatively, if a public stakeholder (such as a policy affairs team) has an in-house set of labels that they apply to bills, this modeling approach could be used to improve tracking mechanisms.
- **Investigating term representations:** Finally, this method may hold value for investigating legislative attitudes at large across the training corpus. For instance, while the current model associates “China” and “Africa” with “International Affairs”, it associates “Chinese” and “African” with “Armed Forces and National Security”. Understanding such term interactions may reveal subtle trends and biases that a less interpretable modeling approach might not reveal.

References

Notably, while the experiment did not borrow code directly from any individual website or repository, LLM assistance was used heavily for certain portions of the project, including all tests; see the following page for details.

1. “Policy Areas — Field Values”, *Congress.gov*, accessed April 6, 2025, https://www.congress.gov/help/field-values/policy-area
2. “CRS Products from the Library of Congress”, *Congress.gov*, accessed April 6, 2025, https://www.congress.gov/crs-products
3. “About CRS Bill Summaries”, *Congress.gov*, accessed April 6, 2025, https://www.congress.gov/help/bill-summaries
4. Hunter Savery, “Publishing pileup: Congressional bills are slow to reach the public”, *Roll Call*, March 5, 2025, https://rollcall.com/2025/03/05/publishing-pileup-congressional-bills-slow-to-reach-public/
5. “Bulk Data Repository”, *GovInfo*, accessed April 30, 2025, https://www.govinfo.gov/bulkdata
6. “The origins of TF-IDF (term frequency-inverse document frequency)”, *BytePlus*, April 25, 2025, https://www.byteplus.com/en/topic/400324?title=the-origins-of-tf-idf-term-frequency-inverse-document-frequency
7. “1.4. Support Vector Machines”, *scikit-learn.org*, accessed April 30, 2025, https://scikit-learn.org/stable/modules/svm.html#svm-classification
8. “1.1.11. Logistic regression”, *scikit-learn.org*, accessed April 30, 2025, https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
9. “1.9. Naive Bayes”, *scikit-learn.org*, accessed April 30, 2025, https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes
10. “3.1. Cross-validation: evaluating estimator performance”, *scikit-learn.org*, accessed April 30, 2025, https://scikit-learn.org/stable/modules/cross_validation.html
11. “SVC”, *scikit-learn.org*, accessed April 30, 2025, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
12. “recall_score”, *scikit-learn.org*, accessed April 30, 2025, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html
13. “Federal Legislative History Research”, *University of Minnesota Law School*, accessed April 30, 2025, https://libguides.law.umn.edu/c.php?g=125795&p=823607
14. “MultinomialNB”, *scikit-learn.org*, accessed April 30, 2025, https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

LLM attributions

A significant portion of the code was written with LLM assistance (Gemini Pro 2.5, o4-Mini, and others). Some rough estimates are provided below, broken out by module. Numbers below represent lines of code (including whitespace, excluding unused __name__ == '__main__' blocks).

"Original Code" implies that LLMs were not used at any stage in the drafting process. "LLM code (mod)" includes any slightly altered LLM outputs as well as cases where LLMs were used to wireframe a working implementation to help me explore available methods, test altered code structures, and more without directly borrowed code. "LLM code (unmod)" includes code that was used directly with little to no alterations.

Miscellaneous notes:

- The asynchronous download implementation in bulk_data.py borrows from original code created earlier (outside of this class) for a different dataset
- Testing was entirely LLM-driven due to inexperience in this arena

Module: data_collection				
Script	**Original code**	**LLM code (mod)**	**LLM code (unmod)**	**% borrowed**
bulk_data.py	249	14	0	0
parsing.py	90	6	0	0
policy_areas.py	113	10	0	0
utils/downloads.py	39	0	0	0

Modules: training, app				
Script	**Original code**	**LLM code (mod)**	**LLM code (unmod)**	**% borrowed**
training.py	45	147	24	11
app.py	0	365	74	17

Module: cli				
Script	**Original code**	**LLM code (mod)**	**LLM code (unmod)**	**% borrowed**
get_data.py	99	0	0	0
run_app.py	64	0	0	0
train_model.py	64	0	0	0

External folder: tests				
Script	**Original code**	**LLM code (mod)**	**LLM code (unmod)**	**% borrowed**
data_collection/test_bulk_data.py	0	0	46	100
data_collection/test_policy_areas.py	0	0	26	100
app/test_app.py	0	0	191	100